**Universidade de Aveiro** Departamento de Matemática
**2009**

**Ana Mafalda**
**de Oliveira Martins**

# Optimização Geométrica em Problemas de Visibilidade: Soluções Metaheurísticas e Exactas

# Geometric Optimization on Visibility Problems: Metaheuristic and Exact Solutions

**Universidade de Aveiro** Departamento de Matemática
**2009**

**Ana Mafalda**
**de Oliveira Martins**

# Optimização Geométrica em Problemas de Visibilidade: Soluções Metaheurísticas e Exactas

# Geometric Optimization on Visibility Problems: Metaheuristic and Exact Solutions

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Matemática, realizada sob a orientação científica do Doutor Antonio Leslie Bajuelos Domínguez, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro

Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio.

**o júri**

presidente

Doutor Casimiro Adrião Pio
Professor Catedrático do Departamento de Ambiente e Ordenamento da Universidade de Aveiro

Doutor Jesus Garcia López de Lacalle
Professor Catedrático da Universidade Politécnica de Madrid, Espanha

Doutor Pedro Manuel Rangel Santos Henriques
Professor Associado da Escola de Engenharia da Universidade do Minho

Doutora Maria Rosália Dinis Rodrigues
Professora Associada do Departamento de Matemática da Universidade de Aveiro

Doutor Adriano Martins Lopes
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Doutor Antonio Leslie Bajuelos Dominguez
Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro (Orientador)

**agradecimentos**

Em primeiro lugar ao Professor Doutor Antonio Leslie Bajuelos que permitiu a realização desta tese e a supervisionou. A minha gratidão pela orientação estimulante, conselhos preciosos, ensinamentos e confiança demonstrada. Obrigada!

À Universidade de Aveiro que em tudo facilitou o desenvolvimento desta dissertação.

À Fundação para a Ciência e Tecnologia pela atribuição de uma bolsa de doutoramento que abrangeu todo o tempo da tese que se apresenta.

Aos Professores Gregorio Hernández e Santiago Canales pelo apoio, incentivo e colaboração nos trabalhos desenvolvidos que se tornaram fundamentais na realização deste trabalho.

Agradeço ainda à Inês Pereira e à Bárbara Oliveiros que em tantos momentos me ajudaram e que muito contribuíram para esta tese.

Quero ainda render uma homenagem especial à minha colega e amiga Margarida Corte-Real pela enorme contribuição que deu para esta tese. As suas sugestões e esclarecimentos foram, também, determinantes na elaboração do texto, tornando-o claro e preciso. Muitíssimo obrigada!

Aos meus pais pelos valores e ensinamentos que me deram para enfrentar questões da vida, uma vez que uma tese não é apenas produto de investigação, orientação e apoios.

Às minhas irmãs que apesar de estarem longe estiveram sempre tão perto.

Por fim agradeço a todos os meus amigos que me acompanharam durante todo este tempo.

**palavras-chave**

Geometria Computacional, Problemas de Visibilidade, Problema da Galeria de Arte, Optimização Geométrica, Métodos de Aproximação, Metaheurísticas.

**resumo**

Os problemas de visibilidade têm diversas aplicações a situações reais. Entre os mais conhecidos, e exaustivamente estudados, estão os que envolvem os conceitos de vigilância e ocultação em estruturas geométricas (problemas de vigilância e ocultação). Neste trabalho são estudados problemas de visibilidade em estruturas geométricas conhecidas como polígonos, uma vez que estes podem representar, de forma apropriada, muitos dos objectos reais e são de fácil manipulação computacional. O objectivo dos problemas de vigilância é a determinação do número mínimo de posições para a colocação de dispositivos num dado polígono, de modo a que estes dispositivos consigam “ver” a totalidade do polígono. Por outro lado, o objectivo dos problemas de ocultação é a determinação do número máximo de posições num dado polígono, de modo a que quaisquer duas posições não se consigam “ver”. Infelizmente, a maior parte dos problemas de visibilidade em polígonos são NP-difíceis, o que dá origem a duas linhas de investigação: o desenvolvimento de algoritmos que estabelecem soluções aproximadas e a determinação de soluções exactas para classes especiais de polígonos. Atendendo a estas duas linhas de investigação, o trabalho é dividido em duas partes.

Na primeira parte são propostos algoritmos aproximados, baseados essencialmente em metaheurísticas e metaheurísticas híbridas, para resolver alguns problemas de visibilidade, tanto em polígonos arbitrários como ortogonais. Os problemas estudados são os seguintes: “Maximum Hidden Vertex Set problem”, “Minimum Vertex Guard Set problem”, “Minimum Vertex Floodlight Set problem” e “Minimum Vertex k-Modem Set problem”. São também desenvolvidos métodos que permitem determinar a razão de aproximação dos algoritmos propostos. Para cada problema são implementados os algoritmos apresentados e é realizado um estudo estatístico para estabelecer qual o algoritmo que obtém as melhores soluções num tempo razoável. Este estudo permite concluir que as metaheurísticas híbridas são, em geral, as melhores estratégias para resolver os problemas de visibilidade estudados. Na segunda parte desta dissertação são abordados os problemas “Minimum Vertex Guard Set”, “Maximum Hidden Set” e “Maximum Hidden Vertex Set”, onde são identificadas e estudadas algumas classes de polígonos para as quais são determinadas soluções exactas e/ou limites combinatórios.

**keywords**

Computational Geometry, Visibility Problems, Art Gallery Problem, Geometric Optimization, Approximation Methods, Metaheuristics.

**abstract**

Visibility problems have several applications to real-life problems. Among the most distinguished and exhaustively studied visibility problems are the ones involving concepts of guarding and hiding on geometrical structures (guarding and hiding problems). This work deals with visibility problems on geometrical structures known as polygons, since polygons are appropriate representations of many real-world objects and are easily handled by computers. The objective of the guarding problems studied in this thesis is to find a minimum number of device positions on a given polygon such that these devices collectively "see" the whole polygon. On the other hand, the goal of the hiding problems is to find a maximum number of positions on a given polygon such that no two of these positions can “see" each other. Unfortunately, most of the visibility problems on polygons are NP-hard, which opens two lines of investigation: the development of algorithms that establish approximate solutions and the determination of exact solutions on special classes of polygons. Accordingly, this work is divided in two parts where these two lines of investigation are considered.

The first part of this thesis proposes approximation algorithms, mainly based on metaheuristics and hybrid metaheuristics, to tackle some visibility problems on arbitrary and orthogonal polygons. The addressed problems are the Maximum Hidden Vertex Set problem, the Minimum Vertex Guard Set problem, the Minimum Vertex Floodlight Set problem and the Minimum Vertex k-Modem Set problem. Methods that allow the determination of the performance ratio of the developed algorithms are also proposed. For each problem, the proposed algorithms are implemented and a statistical study is performed to determine which of the developed methods obtains the best solution in a reasonable amount of time. This study allows to conclude that, in general, the hybrid metaheuristics are the best approach to solve the studied visibility problems. The second part of this dissertation addresses the Minimum Vertex Guard Set problem, the Maximum Hidden Set problem and the Maximum Hidden Vertex Set problem, where some classes of polygons are identified and studied and for which are determined exact solutions and/or combinatorial bounds.

# Contents

Contents . . . i
List of Figures . . . v
List of Tables . . . xiii
List of Algorithms . . . xvii

**1 Introduction** **1**
1.1 Visibility Problems . . . 5
1.1.1 Terminology and Definitions . . . 6
1.1.2 Guarding and Hiding Problems . . . 8
1.2 Structure of the Thesis . . . 12

**I Approximation Strategies for Visibility Problems** **17**

**2 Approximation Methods** **21**
2.1 Metaheuristics . . . 21
2.1.1 Simulated Annealing . . . 24
2.1.2 Genetic Algorithms . . . 29
2.2 Hybrid Metaheuristics . . . 35

**3 Maximum Hidden Vertex Set Problem** **39**
3.1 Problem Description . . . 40
3.2 Approximation Methods . . . 41
3.2.1 Greedy Strategies . . . 41
3.2.2 Simulated Annealing Strategy . . . 43
3.2.3 Genetic Algorithms Strategy . . . 45
3.3 Greedy-Sequential Strategy for the Minimum Clique Partition Problem . . . 49
3.4 Experiments and Results . . . 51
3.4.1 Arbitrary Polygons . . . 51
3.4.1.1 Analysis of the SA Parameters . . . 51

3.4.1.2 Comparison of the four strategies . . . 54
3.4.2 Orthogonal Polygons . . . 57
3.4.2.1 Analysis of the SA Parameters . . . 58
3.4.2.2 Comparison of the four strategies . . . 59
3.5 Concluding Remarks . . . 63

**4 Minimum Vertex Guard Set Problem 65**
4.1 Problem Description . . . 66
4.2 Approximation Methods . . . 68
4.2.1 Pre-processing Step . . . 68
4.2.2 Greedy Strategy . . . 68
4.2.3 Simulated Annealing Strategy . . . 70
4.2.4 Genetic Algorithms Strategy . . . 72
4.2.5 Hybrid Strategies . . . 76
4.3 Greedy Strategies for visibility-independent sets . . . 77
4.4 Experiments and Results . . . 79
4.4.1 Arbitrary Polygons . . . 80
4.4.1.1 Analysis of the SA Parameters . . . 80
4.4.1.2 Analysis of the GA Parameters . . . 87
4.4.1.3 Comparison of the five strategies . . . 91
4.4.2 Orthogonal Polygons . . . 95
4.4.2.1 Analysis of the SA Parameters . . . 96
4.4.2.2 Analysis of the GA Parameters . . . 102
4.4.2.3 Comparison of the five strategies . . . 105
4.5 Concluding Remarks . . . 109

**5 Minimum Vertex Floodlight Set Problem 111**
5.1 Problem Description . . . 111
5.2 Approximation Methods . . . 114
5.2.1 Pre-processing Step . . . 115
5.2.2 Simulated Annealing Strategy . . . 116
5.2.3 Genetic Algorithms Strategy . . . 118
5.2.4 Hybrid Strategies . . . 120
5.3 Greedy Strategy for floodlight visibility-independent sets . . . 120
5.4 Experiments and Results . . . 122
5.4.1 Orthogonal Polygons . . . 122
5.4.1.1 Analysis of the SA Parameters . . . 123
5.4.1.2 Comparison of the four strategies . . . 130

5.5 Concluding Remarks . . . . . . . . . . 133

**6 Minimum Vertex $k$-Modem Set Problem** **135**

6.1 Problem Description . . . . . . . . . . 136

6.2 $k$-Modem Visibility Polygon . . . . . . . . . . 139

6.3 Approximation Method . . . . . . . . . . 146

6.4 Experiments and Results . . . . . . . . . . 148

6.4.1 Arbitrary Polygons . . . . . . . . . . 148

6.4.2 Orthogonal Polygons . . . . . . . . . . 151

6.5 Concluding Remarks . . . . . . . . . . 153

**II Visibility Problems on Special Classes of Polygons** **155**

**7 A Subclass of Orthogonal Polygons: the grid $n$-ogons** **159**

7.1 Conventions, Definitions and Results . . . . . . . . . . 159

7.2 More Results on grid $n$-ogons . . . . . . . . . . 164

7.2.1 SPIRAL grid $n$-ogons . . . . . . . . . . 174

7.2.2 Some Problems related to THIN grid $n$-ogons . . . . . . . . . . 179

7.2.2.1 MAX-AREA-THIN grid $n$-ogon . . . . . . . . . . 179

7.2.2.2 Classifying THIN grid $n$-ogons . . . . . . . . . . 180

7.3 Visibility Problems on grid $n$-ogons . . . . . . . . . . 187

7.3.1 MINIMUM VERTEX GUARD SET Problem on grid n-ogons . . . . . . . . . . 188

7.3.1.1 FAT grid $n$-ogons . . . . . . . . . . 188

7.3.1.2 THIN grid $n$-ogons . . . . . . . . . . 188

7.3.2 MAXIMUM HIDDEN VERTEX SET Problem on grid n-ogons . . . . . . . . . . 200

7.3.2.1 THIN grid $n$-ogons . . . . . . . . . . 200

7.4 Concluding Remarks . . . . . . . . . . 203

**8 Spiral and Histogram Polygons** **205**

8.1 MAXIMUM HIDDEN VERTEX SET and MAXIMUM HIDDEN SET Problems . . . 205

8.1.1 Spiral Polygons . . . . . . . . . . 205

8.1.2 Histogram Polygons . . . . . . . . . . 208

8.2 Concluding Remarks . . . . . . . . . . 210

**9 Conclusions** **213**

**Bibliography** **219**

# List of Figures


1.1 A spiral polygon (reflex chain in bold). . . . . . . . . . . . . . . . . . . . . . . . . . 7
1.2 (a) The point $x$ sees $y$ and does not see $z$; (b) The visibility polygon of $x$. . . . 8
1.3 A guarding set of the polygon $P$. . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.1 General scheme of a genetic algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . 29
2.2 Example of the roulette wheel selection method (from [112]). . . . . . . . . . . 32
2.3 (a) SA is an additional genetic operator; (b) SA replaces the mutation operator. 37
2.4 Three different ways to use trajectory methods in population based techniques in a pipeline fashion. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

3.1 (a) *Triangular saw* polygons; (b) *Staircase* polygons. . . . . . . . . . . . . . . . 40
3.2 A 20-vertex polygon and (a) $Vis(v_2, P)$ and $HR_2 = \{HR_2^1, HR_2^2\}$; (b) $Vis(v_5, P)$ and $HR_5 = \{HR_5^1, HR_5^2, HR_5^3, HR_5^4\}$. . . . . . . . . . . . . . . . . . . . . . . . . . 41
3.3 A 10-vertex polygon and its visibility graph. . . . . . . . . . . . . . . . . . . . . . 42
3.4 An element $S_i \in S$ (for a 20-vertex polygon) and its representation. Red dots represent hidden vertices. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
3.5 Solution Validation. Red dots represent hidden vertices. . . . . . . . . . . . . . 44
3.6 An individual $I$ (for a 25-vertex polygon) and its representation. Red dots represent hidden vertices. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
3.7 Polygon with $n = 10$ and its initial population. . . . . . . . . . . . . . . . . . . 47
3.8 Single point crossover. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
3.9 Single point crossover and child validation. . . . . . . . . . . . . . . . . . . . . . 48
3.10 Mutation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
3.11 A clique partition (with four cliques) of $VG(P)$. . . . . . . . . . . . . . . . . . . 50
3.12 Multiple comparison tests, of the six cases, for $n = 50, 100, 150$ and $200$ (arbitrary polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
3.13 Solutions obtained with the strategies $M_1, M_2, M_3$ and $M_4$ (arbitrary polygons). 55
3.14 Multiple comparison tests of the four methods (arbitrary polygons). . . . . . . 55
3.15 Least Squares Method (arbitrary polygons). . . . . . . . . . . . . . . . . . . . . . 56

3.16 Example of a tested arbitrary polygon with $n = 100$. Clique partition obtained with algorithm $A_1$ and hidden vertex sets obtained with: (a) method $M_1$ and (b) method $M_3$. . . . 57
3.17 The $C$ and $H$ sets in a saw polygon, with $n = 20$, obtained with $A_1$ and $M_3$. . 57
3.18 Multiple comparison tests, of the six cases, for $n = 50, 100, 150$ and 200 (orthogonal polygons). . . . 59
3.19 Solutions obtained with the the strategies $M_1, M_2, M_3$ and $M_4$ (orthogonal polygons). . . . 60
3.20 Multiple comparison tests of the four methods (orthogonal polygons). . . . . 61
3.21 Least Squares Method (orthogonal polygons). . . . 62
3.22 Example of a tested orthogonal polygon with $n = 100$. Clique partition obtained with algorithm $A_1$ and hidden vertex sets obtained with: (a) method $M_1$ and (b) method $M_3$. . . . 63
3.23 The $C$ and $H$ sets in a staircase polygon, with $n = 20$, obtained with $A_1$ and $M_3$. . . . 63

4.1 Three 12-vertex arbitrary polygons: (a) requires 3 guards; (b) require 4 guards. 66
4.2 Two 12-vertex orthogonal polygons: (a) requires 3 guards; (b) requires 1 guard. 67
4.3 An element $S_i \in S$ (for a polygon with $n = 20$) and its representation. . . . . 70
4.4 Generation of $S_j$, a neighbour of $S_i$: (a) $S_j$ is a worse solution; (b) $S_j$ is a better solution. . . . 71
4.5 Initial solution. . . . 71
4.6 Polygon with $n = 20$ ($r = 7$) and its initial population. . . . 73
4.7 Single point crossover. . . . 74
4.8 Two-point crossover. . . . 74
4.9 Uniform crossover. . . . 74
4.10 Generation of an invalid child. . . . 75
4.11 Mutation. . . . 75
4.12 First hybrid strategy. . . . 76
4.13 Second hybrid strategy. . . . 77
4.14 Visibility-independent set. Green dots represent visibility-independent points. . 77
4.15 Multiple comparison tests, of SA Cases 1, 2 and 3 (arbitrary polygons). . . . 85
4.16 Multiple comparison tests, of SA Cases 4, 5 and 6 (arbitrary polygons). . . . 85
4.17 Multiple comparison tests, of SA Cases 7, 8 and 9 (arbitrary polygons). . . . 85
4.18 Solutions obtained with strategies $M_1, M_2, M_3$, $M_4$ and $M_5$ (arbitrary polygons). 93
4.19 Multiple comparison tests of the five methods (arbitrary polygons). . . . 93
4.20 Least Squares Method (arbitrary polygons). . . . 94

4.21 *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on arbitrary polygons with: (a) $n = 50$ ; (b) $n = 100$. . . . . . . . . 95
4.22 *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on arbitrary polygons with: (a) $n = 150$ and (b) $n = 200$. . . . . . . 95
4.23 Multiple comparison tests, of SA Cases 1, 2 and 3 (orthogonal polygons). . . . 100
4.24 Multiple comparison tests, of SA Cases 4, 5 and 6, for $n = 30, 50, 70$ and 100 (orthogonal polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
4.25 Multiple comparison tests, of SA Cases 7, 8 and 9 (orthogonal polygons). . . . 101
4.26 Solutions obtained with strategies $M_1, M_2, M_3$, $M_4$ and $M_5$ (orthogonal polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 106
4.27 Multiple comparison tests of the five methods (orthogonal polygons). . . . . . . 107
4.28 Least Squares Method (orthogonal polygons). . . . . . . . . . . . . . . . . . . . 108
4.29 *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on orthogonal polygons with: (a) $n = 50$ ; (b) $n = 100$. . . . . . . . 109
4.30 *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on orthogonal polygons with: (a) $n = 150$ and (b) $n = 200$. . . . . . 109

5.1 Illuminating an orthogonal polygon with the top-left illumination rule. . . . . . 113
5.2 Illuminating an orthogonal polygon with the four illumination rules. . . . . . . 113
5.3 Vertex floodlights: (a) TL-floodlight; (b) TR-floodlight; (c) BL-floodlight and (d) BR-floodlight. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
5.4 Orthogonal polygons that require $\lfloor \frac{3n-4}{8} \rfloor$ floodlights. . . . . . . . . . . . . . . . 113
5.5 Visibility polygons of a BR-floodlight and a TL-floodlight. . . . . . . . . . . . . 115
5.6 An element $S_l \in S$ for a 16-vertex orthogonal and its representation. . . . . . . 116
5.7 Initial Solution. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
5.8 On the left a 20-vertex orthogonal polygon $P$ and all possible floodlights; on the right the initial population for $P$. . . . . . . . . . . . . . . . . . . . . . . . . . 119
5.9 Floodlight visibility-independent set of an orthogonal polygon. . . . . . . . . . 121
5.10 Multiple comparison tests of: (a) Cases 1, 2 and 3; (b) Cases 4, 5 and 6. . . . . 128
5.11 Multiple comparison tests of Cases 7, 8 and 9. . . . . . . . . . . . . . . . . . . . 128
5.12 Solutions obtained with strategies $M_1, M_2, M_3$, and $M_4$. . . . . . . . . . . . . . . 131
5.13 Multiple comparison tests of the five methods. . . . . . . . . . . . . . . . . . . . 132
5.14 Least Squares Method. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
5.15 *FIS* and $F$ sets obtained on a 100-vertex orthogonal polygon with the methods $A_1$ and: (a) $M_2$; (b) $M_4$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133

6.1 (a) The 2-modem placed on $x$ covers $y$ but it does not cover $z$; (b) $Vis_2(x, P)$. 137
6.2 The vertices of a monotone polygon projected onto a line. . . . . . . . . . . . . 138

6.3 A $n$-vertex monotone polygon requiring $\lceil \frac{n}{2k+2} \rceil$ $k$-modems [9]. . . . . . . . . . . 138

6.4 Rays and one of the critical vertices of $P$. . . . . . . . . . . . . . . . . . . . . . . . 140

6.5 (a) Rays and intersection points; (b) Labelled segments. . . . . . . . . . . . . . 140

6.6 Route for the construction of: (a) $Vis_1(x, P)$ and (b) $Vis_2(x, P)$. . . . . . . . . 141

6.7 (a) $Vis_1(x, P)$ and (b) $Vis_2(x, P)$. . . . . . . . . . . . . . . . . . . . . . . . . 141

6.8 Rules to label the critical vertices (shaded zones represent $int(P)$). . . . . . . . 142

6.9 Labelled critical vertices. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

6.10 Rule to label the intersection points, which are relative interior points of edges of $P$ (shaded zones represent $int(P)$). . . . . . . . . . . . . . . . . . . . . . . . 143

6.11 Labelled intersection points. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

6.12 First labelled edge. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

6.13 Labelled segments. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

6.14 Three rays, one ray with two critical vertices and two rays with one critical vertex. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144

6.15 Intersection Points. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144

6.16 Rule to label the intersection points. . . . . . . . . . . . . . . . . . . . . . . . . 145

6.17 Region covered by a k-modem in a orthogonal 30-vertex polygon: (a) $k = 2$ and (b) $k = 4$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145

6.18 A 20-vertex arbitrary polygon $P$ and: (a) $Vis_2(x, P)$; (b) $Vis_4(x, P)$; (c) $Vis_6(x, P)$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146

6.19 A 100-vertex arbitrary polygon $P$ and: (a) $Vis_2(x, P)$; (b) $Vis_4(x, P)$; (c) $Vis_6(x, P)$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146

6.20 A 100-vertex arbitrary polygon $P$ and: (a) $Vis_2(x, P)$; (b) $Vis_4(x, P)$; (c) $Vis_6(x, P)$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146

6.21 Linear adjustment $k = 2$: (a) arbitrary polygons; (b) monotone arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150

6.22 Linear adjustment $k = 4$: (a) arbitrary polygons; (b) monotone arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150

6.23 Linear adjustment $k = 2$: (a) orthogonal polygons; (b) grid monotone orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152

6.24 Linear adjustment $k = 4$: (a) orthogonal polygons; (b) grid monotone orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153

7.1 A $n$-ogon $P$ and its $\Pi(P)$ partition. . . . . . . . . . . . . . . . . . . . . . . . . 160

7.2 A grid $n$-ogon merged into a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid and the free staircase neighborhood for each of its convex vertices [124]. . . . . . . . . . . . . . . . . 161

7.3 The four grid 14-ogon that may be constructed if INFLATE-PASTE is applied to the given 12-ogon, extending the vertical edge that ends at vertex $v_{10}$ [124]. 161
7.4 The three 12-ogons on the left are mapped in the grid 12-ogon on the right. And the three 12-ogons on the left can be obtained from the grid 12-ogon on the right [124]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
7.5 Eight grid$n$-ogons that are symmetrically equivalent. From left to right, we see images by clockwise rotations of 90°, 180° and 270°, by flips wrt horizontal and vertical axes and flips wrt positive and negative diagonals [124]. . . . . . . . . . 162
7.6 The unique FAT $n$-ogons, for $n = 6, 8, 10$ and 12. . . . . . . . . . . . . . . . . . 163
7.7 Three THIN 10-ogons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
7.8 A family of grid $n$-ogons with MAX-AREA, for $r = 2, 3, 4$ and 5. . . . . . . . . . 163
7.9 A sequence of MAX-AREA $n$-ogons, for $r = 6$. . . . . . . . . . . . . . . . . . . . . 163
7.10 The unique grid MIN-AREA grid $n$-ogons, for $r = 1, 2, 3$ and 4. . . . . . . . . . 163
7.11 On the left is the FAT grid 14-ogon, it has area 27. On the right is a 14-ogon with area 28, which is the maximum for $n = 14$. . . . . . . . . . . . . . . . . . . . 163
7.12 THIN grid 12-ogon with area 15, where the area of the MIN-AREA grid 12-ogon is equal to 9. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
7.13 A $n$-ogon $P$ and: (a) partition $\Pi_H(P)$; (b) partition $\Pi(P)$. . . . . . . . . . . . . 164
7.14 A $n$-ogon $P$ and: (a) $G_{\Pi_H(P)}$; (b) $G_{\Pi(P)}$. . . . . . . . . . . . . . . . . . . . . . . . 164
7.15 On the right we can see two grid 14-ogons that can result from the application of INFLATE-PASTE to the 12-ogon on the left, extending the vertical edge that ends at vertex $v_{10}$. In the top-right polygon the number of internal vertices increases in one unit and in the bottom-right polygon this number is maintained. 165
7.16 Three THIN grid 10-ogon and respective dual graphs. . . . . . . . . . . . . . . . . 166
7.17 6-ogon $P$, $\Pi(P)$ and $G_{\Pi(P)}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
7.18 The two possible types of rectangles (shaded) that correspond to leaves in $G_{\Pi_H(P)}$, being $P$ a grid $n$-ogon. . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
7.19 (a) $p \in \overline{vs_v}$; (b) $R_3$ with three boundary edges and one interior edge and (c) Construction of $G_{\Pi(P)}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
7.20 (a) $R = R_1$ is of Type 2; (b) Removal of $R = R_1$ and (c) $v$ is not a vertex of $R_2$. 167
7.21 Four hypotheses to the edges of $R_2$. . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.22 (a) Case b) cannot take place; (b) $R_2$ corresponds to a leaf in $G_{\Pi(Q)}$. . . . . . . 168
7.23 (a) PASTE operation; (b) Construction of $G_{\Pi(P)}$. . . . . . . . . . . . . . . . . . . 169
7.24 A grid 10-ogon and respective dual graph. . . . . . . . . . . . . . . . . . . . . . . 169
7.25 Subgraph of $G_{\Pi(P)}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
7.26 The $r$-pieces of a THIN: (a) *Type 1*; (b) *Type 2*; (c) *Type 3*. . . . . . . . . . . . 170
7.27 The $r$-pieces of a THIN, from left to right: (a) *Type 1*; (b) *Type 2*; (c) *Type 3*. . 171

7.28 $FSN(v_i)$ (free staircase neighborhood of $v_i$). . . . . . . . . . . . . . . . . . . . . . . 171
7.29 The three possibilities for the center of $C$, $c = (x_c, y_c)$: (a) situation (1); (b) situation (2); (c) situation (3). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.30 INFLATE operation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.31 PASTE operation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.32 The only convex vertices that could yield, by INFLATE-PASTE, the illustrated THIN grid 14-ogons are $v_3$, $v_4$, $v_{11}$ and $v_{12}$. . . . . . . . . . . . . . . . . . . . . . 173
7.33 A THIN grid $n$-ogon with $r = 4$; on the left is represented its dual graph $G_{\Pi(P)}$ and on the right its skeleton. . . . . . . . . . . . . . . . . . . . . . . . . . . . 174
7.34 Reflex (in bold) and convex chains. . . . . . . . . . . . . . . . . . . . . . . . . . 175
7.35 Grid $n$-ogon with $r = 1$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175
7.36 On the left it is illustrated *Case 1*, i.e., $e_H(c_1) \equiv \overline{u_r c_1}$; and on the right Case 2, that is, $e_H(c_1) \equiv \overline{c_1 c_2}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.37 A sequence of SPIRAL grid 10-ogons. . . . . . . . . . . . . . . . . . . . . . . . . 176
7.38 Rectangles that might be glued by PASTE to yield $Q$. . . . . . . . . . . . . . . . . 177
7.39 Rectangles glued to $P$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
7.40 From left to right: *Case a)*, *Case b)*, *Case c)* and *Case d)*. . . . . . . . . . . . 179
7.41 Rectangles glued by PASTE to yield $Q$. . . . . . . . . . . . . . . . . . . . . . . . 179
7.42 (a) From left to right $MA_2 = 6, MA_3 = 11, MA_4 = 17, MA_5 = 24$; (b) Two THIN 14-ogons with area 24, $MA_5 = 24$. . . . . . . . . . . . . . . . . . . . . . . . 180
7.43 THIN grid $n$-ogon with $r = 4$ and its skeleton. . . . . . . . . . . . . . . . . . . . 180
7.44 THIN grid $n$-ogons with 4 reflex vertices and respective chains. . . . . . . . . . 183
7.45 Constructing the THIN grid 12-ogon from the chains:(a) $c = 1001$ and (b) $c = 1110$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183
7.46 Constructing a THIN grid 12-ogon from the chains:(a) $c = 1001$ and (b) $c = 1110$.184
7.47 (a) The chain that represents the THIN after the horizontal and vertical reflections is $c = 111011$; (b) The chain that represents the THIN after the vertical reflection is $c = 1001$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.48 Guarded FAT grid 14-ogon. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
7.49 (a) Removing “line 3”; (b)“Constructing” $P$. . . . . . . . . . . . . . . . . . . . . 189
7.50 MIN-AREA grid $n$-ogons with $r = 1, 2, 3$ and 4. . . . . . . . . . . . . . . . . . . 190
7.51 MIN-AREA grid $n$-ogon with $r = 5$. . . . . . . . . . . . . . . . . . . . . . . . . 190
7.52 (a) Polygon $\tilde{P}_2$; (b) Applying induction hypotheses to $\tilde{P}_2$. . . . . . . . . . . . . 191
7.53 (a) Polygon $P_2$ completely covered; (b) Rectangle $R$; (c) Quadrilaterals $Q_1$ and $Q_2$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192
7.54 Min-Area grid 12-ogon. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.55 Visibility Regions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193

7.56 Min-Area grid 12-ogon. . . . . . 193
7.57 Visibility Regions. . . . . . 193
7.58 Construction of the Min-Area grid 18-ogon from two Min-Area grid 12-ogons. . 194
7.59 Polygon $P$ ("merging" $Q$ with the Min-Area grid 12-ogon). . . . . . 194
7.60 Min-Area grid $n$-ogons with $r = 1, 2$ and 3. . . . . . 195
7.61 Min-Area grid $n$-ogons $Q_m$, $Q_{m+2}$ and $Q_{m+4}$. . . . . . 195
7.62 (a) $CR_0$; (b) $CR_k$ and (c) $CR_i$, with $i \neq 0, k$. . . . . . 196
7.63 (a) $CR_0$; (b) $CR_r$ and (c) $CR_i$, $i \in \{1, ..., r-1\}$. . . . . . 197
7.64 (a) $CR_i$, $i \in \{1, ..., r-1\}$; (b) $CR_0$ and (c) $CR_r$. . . . . . 198
7.65 Spiral $n$-ogons with $r$ odd. . . . . . 198
7.66 Spiral $n$-ogons with $r$ even. . . . . . 199
7.67 Two THIN grid $n$-ogons, its skeletons and the chains $C_1$ and $C_2$ ($C_1$ in bold). . 201
7.68 Two THIN grid $n$-ogons and marked hidden vertices ($C_1$ in bold). . . . . . 201
7.69 On the left $v^1_{2k-2}$ is reflex and on the right it is convex. . . . . . 201
7.70 The shaded zones are not visible by the marked vertices. . . . . . 202
7.71 The shaded zones are not visible by the marked vertices. . . . . . 202

8.1 An example of a spiral polygon with its reflex chain. . . . . . 205
8.2 Bounds for $h$. Black dots represent hidden vertices. . . . . . 206
8.3 Placement of hidden vertices (the black dots represent vertices marked as hidden).207
8.4 Decomposition into pieces $A$ and $B$. . . . . . 207
8.5 Hidden points on a spiral polygon. . . . . . 208
8.6 Histogram polygon. . . . . . 209
8.7 Pyramid polygon. . . . . . 209
8.8 Histogram decomposition. . . . . . 210
8.9 Hidden points on histograms polygons. . . . . . 210

# List of Tables


2.1 Example of the roulette wheel selection method. . . . . . . . . . . . . . . . . . . 32

3.1 Studied cases for SA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
3.2 Results obtained with Case 1, Case 2 and Case 3 ($T_0 = n$) on arbitrary polygons. 52
3.3 Results obtained with Case 4, Case 5 and Case 6 ($T_0 = 1000$) on arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
3.4 Results obtained with $M_1$, $M_2$, $M_3$ and $M_4$ (arbitrary polygons). . . . . . . . . 54
3.5 Average number of hidden vertices (arbitrary polygons). . . . . . . . . . . . . . 56
3.6 Results obtained with Cases 1, Case 2 and Case 3 ($T_0 = n$) on orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
3.7 Results obtained with Cases 4, Case 5 and Case 6 ($T_0 = 1000$) on orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
3.8 Results obtained with $M_1$, $M_2$, $M_3$ and $M_4$ (orthogonal polygons). . . . . . . . 60
3.9 Average number of hidden vertices (orthogonal polygons). . . . . . . . . . . . . 62

4.1 Studied cases for SA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
4.2 Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$) on arbitrary polygons. . . 81
4.3 Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$) on arbitrary polygons. . 81
4.4 Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$) on arbitrary polygons.) . . 82
4.5 Multiple comparison tests, of SA Cases, for 30-vertex arbitrary polygons. . . . 82
4.6 Multiple comparison tests, of SA Cases, for 50-vertex arbitrary polygons. . . . 83
4.7 Multiple comparison tests, of SA Cases, for 70-vertex arbitrary polygons. . . . 83
4.8 Multiple comparison tests, of SA Cases, for 100-vertex arbitrary polygons. . . . 83
4.9 Results obtained with SA Case 1, with and without the use of the dominance matrix (arbitrary polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . 87
4.10 Studied cases for GA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
4.11 Results obtained with GA Case 1 (arbitrary polygons). . . . . . . . . . . . . . . 88
4.12 Results obtained with GA Case 2 (arbitrary polygons). . . . . . . . . . . . . . . 88
4.13 Results obtained with GA Case 3 (arbitrary polygons). . . . . . . . . . . . . . . 88

4.14 Results obtained with GA Case 4 (arbitrary polygons). . . . . . . . . . . . . . . . 88
4.15 Results obtained with GA Case 5 (arbitrary polygons). . . . . . . . . . . . . . . . 88
4.16 Results obtained with GA Case 6 (arbitrary polygons). . . . . . . . . . . . . . . . 88
4.17 Results obtained with GA Case 7 (arbitrary polygons). . . . . . . . . . . . . . . . 89
4.18 Results obtained with GA Case 8 (arbitrary polygons). . . . . . . . . . . . . . . . 89
4.19 Results obtained with GA Cases 8, 8.1 and 8.2 (arbitrary polygons). . . . . . . 90
4.20 Results obtained with GA Case 8.2, with and without the use of the dominance matrix (arbitrary polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
4.21 Results obtained with $M_1$, $M_2$ and $M_3$ (arbitrary polygons). . . . . . . . . . . . 92
4.22 Results obtained with $M_4$ and $M_5$ (arbitrary polygons). . . . . . . . . . . . . . 92
4.23 Average of the minimum number of vertex guards (arbitrary polygons). . . . . 94
4.24 Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$) on orthogonal polygons. . 96
4.25 Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$) on orthogonal polygons. 96
4.26 Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$) on orthogonal polygons. . 97
4.27 Multiple comparison tests, of SA Cases, for 30-vertex orthogonal polygons. . . 97
4.28 Multiple comparison tests, of SA Cases, for 50-vertex orthogonal polygons. . . 98
4.29 Multiple comparison tests, of SA Cases, for 70-vertex orthogonal polygons. . . 98
4.30 Multiple comparison tests, of SA Cases, for 100-vertex orthogonal polygons. . . 98
4.31 Results obtained with GA Case 1 (orthogonal polygons). . . . . . . . . . . . . . 102
4.32 Results obtained with GA Case 2 (orthogonal polygons). . . . . . . . . . . . . . 102
4.33 Results obtained with GA Case 3 (orthogonal polygons). . . . . . . . . . . . . . 102
4.34 Results obtained with GA Case 4 (orthogonal polygons). . . . . . . . . . . . . . 102
4.35 Results obtained with GA Case 5 (orthogonal polygons). . . . . . . . . . . . . . 102
4.36 Results obtained with GA Case 6 (orthogonal polygons). . . . . . . . . . . . . . 102
4.37 Results obtained with GA Case 7 (orthogonal polygons). . . . . . . . . . . . . . 103
4.38 Results obtained with GA Case 8 (orthogonal polygons). . . . . . . . . . . . . . 103
4.39 Results obtained with GA Cases 8, 8.1 and 8.2 (orthogonal polygons). . . . . . 104
4.40 Results obtained with GA Case 8.2, with and without the use of the dominance matrix (orthogonal polygons). . . . . . . . . . . . . . . . . . . . . . . . . . . . 105
4.41 Results obtained with $M_1$, $M_2$ and $M_3$ (orthogonal polygons). . . . . . . . . . . 106
4.42 Results obtained with $M_4$ and $M_5$ (orthogonal polygons). . . . . . . . . . . . . 106
4.43 Average of the minimum number of vertex guards (orthogonal polygons). . . . 108

5.1 Studied cases for SA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
5.2 Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$). . . . . . . . . . . . . . . . 123
5.3 Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$). . . . . . . . . . . . . . . 124
5.4 Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$). . . . . . . . . . . . . . . . 125

5.5 Multiple comparison tests, of SA Cases, for 30-vertex orthogonal polygons. . . 125
5.6 Multiple comparison tests, of SA Cases, for 50-vertex arbitrary polygons. . . . 125
5.7 Multiple comparison tests, of SA Cases, for 70-vertex arbitrary polygons. . . . 126
5.8 Multiple comparison tests, of SA Cases, for 100-vertex arbitrary polygons. . . . 126
5.9 Results obtained with SA Case 1, with and without the use of the dominance matrix. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130
5.10 Results obtained with $M_1$ and $M_2$. . . . . . . . . . . . . . . . . . . . . . . . . . 130
5.11 Results obtained with $M_3$ and $M_4$. . . . . . . . . . . . . . . . . . . . . . . . . . 131
5.12 Average of the minimum number of vertex floodlights. . . . . . . . . . . . . . 132

6.1 Results obtained for $k = 2$ on: (a) arbitrary polygons and (b) monotone arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
6.2 Results obtained for $k = 4$ on: (a) arbitrary polygons and (b) monotone arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
6.3 Results obtained for $k = 2$ on: (a) orthogonal polygons and (b) grid monotone orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151
6.4 Results obtained for $k = 4$ on: (a) orthogonal polygons and (b) grid monotone orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151

9.1 Studied problems on arbitrary polygons. . . . . . . . . . . . . . . . . . . . . . . . 215
9.2 Studied problems on orthogonal polygons. . . . . . . . . . . . . . . . . . . . . . . 215
9.3 Studied problems on monotone arbitrary polygons. . . . . . . . . . . . . . . . . . 215
9.4 Studied problems on monotone grid $n$-ogons polygons. . . . . . . . . . . . . . . 216
9.5 Guarding problem on grid $n$-ogons polygons. . . . . . . . . . . . . . . . . . . . . 217
9.6 Hiding problem on grid $n$-ogons polygons. . . . . . . . . . . . . . . . . . . . . . 217
9.7 Hiding problems on spiral and histogram polygons. . . . . . . . . . . . . . . . . 218

# List of Algorithms


2.1 Iterative (minimization) local search algorithm . . . 23
2.2 Simulated Annealing Algorithm (for a minimization problem) . . . 25
2.3 Basic GA . . . 30
3.1 Determining $H$ from the hidden regions (method $M_1$) . . . 42
3.2 Generation of $I_i^0$, $i \in \{0, \ldots, n-1\}$ . . . 46
3.3 Algorithm to determine a clique partition from the vertex $v_k$ . . . 51
4.1 Greedy strategy (method $M_1$) . . . 69
4.2 Removing Redundant Vertices . . . 69
4.3 Computing *IS* (greedy algorithm $A_1$) . . . 79
5.1 Computing *FIS* (greedy algorithm $A_1$) . . . 122
8.1 Algorithm to place hidden vertices . . . 207

# Chapter 1

# Introduction

This dissertation aims at studying a set of problems related to the *visibility subfield*, a central area of the *computational geometry* field. In a restrictive sense, computational geometry can be defined as a branch of computer science devoted to the study of algorithms that can be stated in geometric terms. According to many authors (for instance, [102]) it is common to date the beginning of this research area to the 1970s with the work of Shamos and very particularly with his doctoral dissertation. Since then, many researchers were fascinated by the challenges posed by the geometric problems. There have been many research by the scientific community dedicated to this topic, reflected in many seminars, conferences or university courses, as well as books and journals. Overviews of key concepts and results in computational geometry are provided in the handbooks by Sack and Urrutia [113] and by Goodman and O'Rourke [66]. Some more classical literature in this field can be found in the books by Preparata and Shamos [109], Edelsbrunner [45], Pach [106] and Toussaint [128]. A recent book by Berg *et al.* [39] is seen as a well-accepted introduction to computational geometry.

In a certain sense, the euclidean geometry can be regarded as a historical precedent of the current algorithmic geometry, because Euclides built his geometric objects using a *tool set* (the ruler and the compass) that can perform certain *primitive* or *basic operations*. After Euclides, while some mathematicians had studied the constructive possibilities of diverse tool sets, others were worried about how to reduce the number of steps in the constructions, being, this way, the precursors of the current interest on computational geometry in the design of geometric algorithms and the analysis of its complexity [7, 109, 127].

However, the computational geometry cannot only be considered "daughter" of the euclidean geometry. The strength with which it emerged, and is maintained today, has to do not only with its relation with geometry, and mathematics in general, but especially with its connection with computer science. Computational geometry skilfully combines the use of classic algorithmic paradigms with the deep analysis of the geometric structure of the problems. The metrics and combinatorial characteristics of the geometric problems allow, through the use of the algorithmic techniques and appropriate data structures, to reduce the solution complexity, sometimes in a surprising way. Consider, for instance, the elegance of the solution given by Megiddo [95] for the classical problem of finding the smallest circle enclosing $n$ given points on the plane, for which the brute-force approach has time complexity $\mathcal{O}(n^4)$. Megiddo obtained an algorithm of time complexity $\mathcal{O}(n)$ by applying an interesting algorithmic scheme that cleverly exploits the problem metric properties. In fact, it can be said that computational geometry, in addition to using well-known algorithm design paradigms, such as *divide and conquer* or *dynamic programming*, has created its own paradigms, the most notorious of them being *sweeping* (*line sweep*, *topological sweep*, and so on). Relatively to the data structures used in computational geometry, things are not very different, for example, the *balanced binary search trees* gave rise to specialized structures particularly suited for handling geometric data, such as the *segment tree* or the *doubly connected edge lists* (DCEL) [7].

As we can see, computational geometry contemplates the development of geometric algorithms. Since it is wished to develop efficient algorithms, it is also necessary to study and analyze the performance of these algorithms. In this study it is adopted the real Random Access Machine (RAM) computational model [109]. Note that the analysis of the geometric algorithms performance establishes another relation with computer science, since this analysis is related to the *computational complexity theory*, which is a main subfield of the computer sciences.

Over the past years many efficient algorithms (and data structures) for geometric problems have been developed. Nevertheless, many of these algorithms have no direct effect in practical geometrical computing, because they are more efficient than other solutions for only huge problem instances. Consequently, they are predominantly considered as contributions to the investigation of the complexity of a geometric problem. On the other hand, many other algorithms are efficient for reasonable problem sizes but they are not very useful in practice yet, because the correct implementation of even the simplest of these algorithms can be extremely difficult. There are two main problems that need to be dealt with to close the gap between the theoretical results and the practical implementations. First, there is the dissimilarity between fast floating-point arithmetic, normally used in practice, and exact arithmetic over the real numbers, assumed in theoretical papers (*precision problem*). Second, there is the lack of explicit handling of degenerate cases in theoretical papers (*degeneracy problem*) [39, 57, 114]. Concerning the precision problem, in theoretical papers the proof of the correctness of an algorithm frequently relies on exact computations, however, in general, replacing exact arithmetic by imprecise floating-point arithmetic does not work. Regarding the degeneracy problem, often the theoretical papers exclude degenerate configurations in the input of the algorithms they described. Simple examples of configurations that are considered as degenerate are the existence of duplicate points or the existence of three collinear points in a given point set. In theory, this approach of excluding degeneracies is justified with the argument that degenerate cases are very unlikely if the input set is randomly chosen over the real numbers. However, in practice degenerate inputs occur frequently. For example, the coordinates of geometric objects can not be randomly chosen on $\mathbb{R}$, but lie on a grid, for instance, they can be created by clicking on a window of a graphical interface [57]. In addition to the precision and degeneracy problems, advanced algorithms bring about the difficulty to understanding and to coding them.

For the aforementioned reasons, it is unreasonable for users to implement geometric algorithms from scratch. As a result, computational geometry libraries, providing correct and efficient reusable implementations, are undoubtedly necessary. The development of such libraries represents a large effort by the computational geometry community. Some examples of computational geometry libraries are the *Computational Geometric Algorithms Library* (CGAL) [2], the Library of Efficient Data Types and Algorithms (LEDA) [96,97], the Wykobi Computational Geometry Library (www.wykobi.com), the FastGeo Computational Geometry Library (www.partow.net/projects/fastgeo) and the Graph Drawing Toolkit (GDToolkit) [1, 40]. Among the developed computational geometry libraries, the CGAL library is the most used (see, e.g., [136]). This is reflected, for example, in the incorporation of the 2D and 3D Delaunay triangulations of CGAL in the well-known MATLAB software (version R2009a).

The unstoppable progress of computational geometry can be explained not only by the development of the computer sciences, but also by its applicability in diverse and current fields such as location problems (e.g., [31]), geographic information systems (e.g. [81]) or biological applications (e.g., [46]), in particular molecular modelling (e.g. [79]). These are merely representative examples of the good results arising from the application of computational geometry techniques. In the present, it is notorious the interest in studying and solving applied problems in this area. Within the geometric problems with application, the visibility problems are doubtless of great interest and updated, due to its applicability in areas such as computer graphics (e.g., [42,43]), pattern recognition (e.g., [103]), computer vision or robotics (e.g., [111]), for example, motion planning (e.g., [12, 35, 116]). Another interesting and promising application of visibility problems is its employment in the treatment planning of a radiation therapy in cancer patients [72].

The interest for visibility problems started, according to Honsberg [71], in response to a question posed by Victor Klee in 1973. Klee's question was: *How many guards are always sufficient to cover the interior of a $n$-wall art gallery room?* This problem is today known as the *original Art Gallery Problem*. Soon after that, Vasek Chvátal [34] established what became known as the *Art Gallery Theorem*: $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and occasionally necessary to cover any simple polygon with $n$ edges. The problem and theorem have this designation because a polygon can be seen as the floor plan of an art gallery room and its points can be considered as suitable places for view elements, such as guards, cameras or light sources. Since the publication of this result, many researches on *art gallery problems*, visibility problems on polygons, have been done by mathematicians and computer scientists. A basic reference in this topic is that of O'Rourke [101] who, in 1987, published a book devoted to these problems. This publication gave a further encouragement to the study of these problems, as well as many variations of the original Art Gallery Problem. Two comprehensive survey papers were also written on this subject, one in 1992 by T. Shermer [119] and, a second one, in 2000 by J. Urrutia [129]. Since then, a large number of papers in this area have appeared and some important problems have been solved. Recently, a book completely devoted to visibility algorithms in two dimensions was written by Ghosh [63].

Many variations of the original Art Gallery Problem are $\mathcal{NP}$-hard (see, e.g., [101, 119, 129]), which does not leave "space" to the design of exact algorithms of reasonable complexity and the need of the development of approximate algorithms. Due to this, the computational complexity of these problems usually opens two lines of investigation: (1) the development of algorithms that establish approximate solutions or (2) the determination of optimal solutions on special classes of simple polygons. According to Urrutia [129], the first line of investigation has not been sufficiently undertaken in the study of art gallery problems. The first result on this line of investigation was published in 1987, where it is established a $\mathcal{O}(n^5 \log n)$ time approximation algorithm that finds a vertex guarding set that is at most $\mathcal{O}(\log n)$ times the minimum number of vertex guards needed to cover a polygon [64]. This work was later taken and extended over by Eidenbenz [50] who designed approximation algorithms for several variations of terrain guarding problems. More recent approximation results are due to Efrat *et al.* [48] who devise provable approximation schemes for problems that arise in optimization of sensor networks, Bottino and Laurentini [27] who propose a location incremental technique to approximate the minimum number of sensors able to cover the edges of a polygon (EDGE COVERING problem), Amit *et al.* [11] who analyze heuristics in the number of guards needed to cover a polygon and Packer [107] who present heuristics to compute multiple watchmen routes. Another approach tackled by Erdem and Sclaroff [54] and Tomás, Bajuelos and Marques [125, 126] obtains approximate solutions for the problem of finding a vertex guarding set with minimum cardinality (MINIMUM VERTEX GUARD SET problem), by transforming MINIMUM VERTEX GUARD SET instances into MINIMUM SET COVER instances, using decompositions of the polygon. Following the same approach, Couto, Souza and Rezende [36] proposed an algorithm to find an optimal solution to the Orthogonal Art Gallery Problem refining discretizations of the polygon. The authors say that an upper bound on the maximum number of iterations effected by the algorithm is $\mathcal{O}(n^4)$ and that this establishes the convergence. However, each iteration can take exponential time since it needs to solve an instance of the SET COVER PROBLEM. To our knowledge, there are only two works where metaheuristics strategies are used to solve visibility problems [4, 5], which derive from Canales doctoral dissertation [30]. In these works the metaheuristics techniques proved to behave very well in solving the problem of finding a guarding set with minimum cardinality (MINIMUM GUARD SET problem).

Concerning the second line of investigation, a lot of research has been done. For example, some of the above cited books, such as [101], [129] and [119], present several problems on special classes of polygons. Worman and Keil [134] have also developed an exact algorithm to place the minimum number of guards with *rectangular visibility* on *orthogonal polygons*, whose time complexity is $\mathcal{O}(n^{17})$. In this problem, besides considering a special class of polygons, they also restricted the way the points see each other. Nilsson and Wood [100] gave a linear time algorithm to find the minimum number of guards necessary to guard a *spiral polygon*. Besides these cited works, the majority of the works presented in subsection 1.1.2, where some visibility problems are presented, follows this line of investigation.

Among the most distinguished and exhaustively studied visibility problems are the *guarding* and *hiding* problems, which are the categories of visibility problems studied in this thesis. These problems are $\mathcal{NP}$-hard or it is strongly believed that they are $\mathcal{NP}$-hard. For this reason, the study of the problems follows the two lines of investigation stated above and, in this way, this dissertation is divided in two parts. In the first one it is proposed approximation algorithms. Since the two cited above works, that use metaheuristics strategies to solve visibility problems, have proven to behave very well, the present dissertation directs its efforts in this approach of investigation, using approximation algorithms mainly based on metaheuristics. Besides, only some of the above cited approximation algorithms have been implemented and, in any case, no experimental results comparing the cardinalities of the solution provided by the algorithms with the optimal solution have been presented. In this dissertation all the proposed approximation algorithms were implemented and some techniques for evaluating the quality of the approximate solutions were developed. In the second part it is studied and presented exact solutions of some guarding and hiding problems on special classes of polygons.

In the next section some basic concepts related to polygons and visibility are introduced, some of the known visibility problems and the problems treated in this work are briefly described (section 1.1) and how their study is organized throughout this thesis (subsection 1.2).

## 1.1 Visibility Problems

As stated before, the interest on visibility problems started when Victor Klee, in 1973, stated the original Art Gallery Problem. In the abstract version of this problem, the floor plan of the art gallery room is modelled by a *simple polygon* $P$ and a *guard* is considered a fixed point on $P$ with $2\pi$ range of visibility and cannot see through the walls. Therefore, in the following subsection some general terminology and definitions related to *polygons* and *visibility*, necessary for the comprehension of this thesis, are given. More specific terminology and definitions will be introduced throughout the thesis.

### 1.1.1 Terminology and Definitions

**Definition 1.1** *A* ***polygonal chain*** *is defined as an ordered sequence of points* $v_0, v_1, \ldots, v_{n-1}$*, with* $n \geq 3$*, called* ***vertices****, together with the set of line segments* $e_0 = \overline{v_0v_1}$*,* $e_1 = \overline{v_1v_2}$*, ...,* $e_{n-2} = \overline{v_{n-2}v_{n-1}}$*, designated by* ***edges****. The polygonal chain is* ***closed*** *if the first and the last vertices are also connected by a line segment, i.e., if the line segment* $e_{n-1} = \overline{v_{n-1}v_0}$ *exists. The polygonal chain is* ***simple*** *if the only intersection of edges are those at common endpoints of consecutive edges.*

A simple closed polygonal chain divides the plane in two regions, an unbounded one, called the exterior region, and a bounded one, the interior. In this area, a simple polygon is usually defined as follows:

**Definition 1.2** *A* ***simple polygon*** $P$ *is defined as the simple closed polygonal chain together with its interior.*

The points $v_i$, with $i = 0, 1, \ldots, n-1$, are designated by vertices of $P$. The set of these vertices is denoted by $V_P$, that is, $V_P = \{v_0, v_1, \ldots, v_{n-1}\}$. The line segments $e_i$, with $i = 0, 1, \ldots, n-1$, are named the edges of $P$ and form the boundary of $P$. The interior of the polygon $P$, the boundary and the exterior are denoted by $int(P)$, $\partial P$ and $ext(P)$, respectively. Thus, $P = int(P) \cup \partial P$. In the sequel, a polygon with $n$ vertices is also called a $n$-vertex polygon or a $n$-gon, for short. Since this work only deals with simple polygons, the term *"polygon"* is used instead of *"simple polygon"*. Unless stated otherwise, the vertices of a polygon $P$ are assumed to be in counterclockwise (CCW) order so that the $int(P)$ lies to the left as one goes through $\partial P$. All index arithmetic is mod $n$, implying a cyclic order of the points, with $v_0$ following $v_{n-1}$ since $(n-1)+1 \equiv n \equiv 0 \pmod{n}$.

**Definition 1.3** *A vertex* $v_i$ *is called* ***reflex*** *if the interior angle between its two incident edges is greater than* $\pi$*. Otherwise, it is called* ***convex****.*

A polygonal chain is called *reflex chain* if its vertices are all reflex (all except the vertices at the end of the chain). A polygonal chain is called *convex chain* if its vertices are all convex.

**Definition 1.4** *A polygon is a* ***spiral*** *if its boundary can be divided in a* ***reflex chain*** *and a* ***convex chain****.*

Note that, a spiral polygon can be expressed as an ordered sequence of vertices $u_1, u_2, ...,$ $u_r, c_1, c_2, ..., c_{n-r}$ where the $u_i$'s are reflex vertices and the $c_i$'s are convex vertices. Thus, the reflex chain is the polygonal chain $c_{n-r}, u_1, ..., u_r, c_1$ and the convex chain is the polygonal chain $c_1, c_2, ..., c_{n-r}$ (see Figure 1.1).

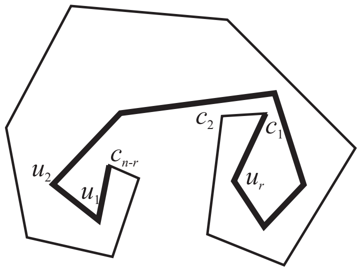



Figure 1.1: A spiral polygon (reflex chain in bold).

**Definition 1.5** *A polygon is called* ***orthogonal*** *if its internal angles have range of* $\frac{\pi}{2}$ *or* $\frac{3\pi}{2}$.

An orthogonal polygon can also be defined as a polygon whose edges are all parallel to a pair of orthogonal axes in the plane. O'Rourke [101] showed that $n = 2r + 4$ for every orthogonal polygon with $n$ vertices, where $r$ is the number of reflex vertices. Consequently, orthogonal polygons have an even number of vertices.

Orthogonal polygons are of great interest. Indeed, most "real life" buildings and galleries are "orthogonal" [129]. Moreover, this kind of polygons arises naturally in certain applications, such as Very Large Scale Integration (VLSI) design and computer graphics. Due to its interest in real applications, all the problems studied in this dissertation contemplate this class of polygons. In the sequel, the terms "*arbitrary polygon*" and "*orthogonal polygon*" are used to mean a polygon (without restrictions) and a polygon belonging to the class of orthogonal polygons, respectively. Throughout this thesis, an orthogonal polygon with $n$ vertices is also called $n$-ogon, for short.

Since the problems studied in this thesis are visibility problems, some visibility concepts will follow.

**Definition 1.6** *Let $P$ be a polygon and $x, y \in P$. The point $x$* ***sees*** *the point $y$ (or $y$ is visible from $x$) if the segment $\overline{xy}$ does not intersect the exterior of $P$, that is, if $\overline{xy} \cap P = \overline{xy}$.*

**Definition 1.7** *Let $P$ be a polygon and $x \in P$. The set of all points of $P$ visible from $x$, that is, the set $Vis(x, P) = \{y \in P : x \text{ sees } y\}$ is called* ***visibility polygon*** *of $x$.*

Figure 1.2 illustrates the last two concepts.

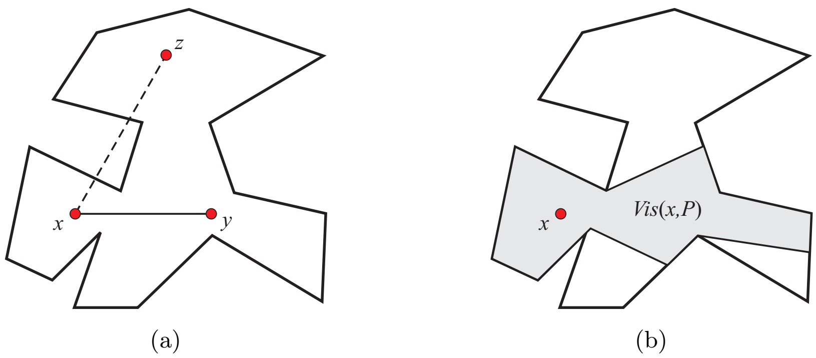



Figure 1.2: (a) The point $x$ sees $y$ and does not see $z$; (b) The visibility polygon of $x$.

**Definition 1.8** *A set of guards $G \subset P$ **covers** $P$ if each point of $P$ is visible by at least one guard, that is, if $\bigcup_{x \in G} Vis(x, P) = P$. In this case, $G$ is called a **guarding set** of $P$.*

Figure 1.3 illustrates a polygon $P$ and a guarding set of $P$. If, in a set of guards, the guards are replaced by light sources (that emit light in all directions) the term visibility is usually replaced by *illumination*. The use of the terms *illuminates* and *guards/covers* follows the notion that the view element is a light source or a guard, respectively [129]. The guarding problems are also known as *illumination problems*.

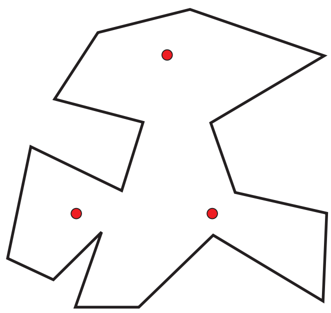


Figure 1.3: A guarding set of the polygon $P$.

The Klees's question can then be reformulated as: *How many guards/lights are always sufficient to guard/illuminate the interior of a $n$-vertex polygon?*

### 1.1.2 Guarding and Hiding Problems

As said before, in 1975, Chvátal [34] established the next result: $\lfloor \frac{n}{3} \rfloor$ *guards/lights are always sufficient and occasionally necessary to cover/illuminate a simple polygon with $n$ vertices.* Avis and Toussaint [15] developed an efficient algorithm to cover $n$-vertex polygons with $\lfloor \frac{n}{3} \rfloor$ guards. But while the established number of guards is always sufficient to cover any $n$-vertex polygon, for many polygons this number is clearly too large (for example, on convex polygons).

This reasoning justifies the following variation of the original problem: *given a polygon $P$ with $n$ vertices, determine the minimum number of guards necessary to cover it* (MINIMUM GUARD SET problem). Aggarwal [8] showed that this problem is $\mathcal{NP}$-hard.

After the first result of Chvátal, the study of several variations of the problem started start multiply. They are known as *guarding problems*. Another category of visibility problems, also studied in the literature, are the *hiding problems* that, besides being theoretical attractive, also have practical applications (e.g. [51, 117]). These problems concern with hiding a maximum number of objects from each other in a given geometric configuration. The problems of hiding objects have application, for example, in computer games, where each player needs to find and collect, or destroy, as many objects as possible. Another application, described by Eidenbenz in [49], is the partition of an area into lots for house building, which uses the three-dimensional version of the problem: hiding points in polyhedra terrains. The hiding problems can also have computer graphics applications, for example, on the hidden surface determination and hidden line removal.

The *guarding* and *hiding* problems are among the most distinguished and exhaustively studied visibility problems. In these problems a geometric configuration is given as input, such as a polygon, a line arrangement, a set of polygons in three-dimensional space or a polyhedra. As, in this dissertation, the input of the studied problems is a polygon, we will not focus on geometric configurations rather than polygons. In fact, the studied guarding problems deal with finding a minimum number of guards/lights positions on a given polygon, such that these guards/lights collectively see the whole polygon and the hiding problems deal with finding a maximum number of positions on a given polygon, such that no two of these positions can see each other.

To a comprehensive study of the guarding problems the reader is referred to the books and surveys that were mentioned earlier in this chapter, namely [101, 119, 129]. Here only some of them, considered significant in the scope of this thesis, will be pointed out.

One of the variations of the original guarding problem was to change the assumptions on the polygons under study. Kahn et al. [77], for example, have proven the *Orthogonal Art Gallery Theorem*. It states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with $n$ vertices, while Edelsbrunner *et al.* [47] showed that this number of guards could be placed in linear time. Rather than change the assumptions on the input polygons, a different variant was to restrict the positions of the guards, for instance, a *vertex guard* is defined to be a guard located at a vertex; in contrast, guards who have no restriction on their location are named *point guards*. A different variant is to change the part of the polygon that must be guarded. Laurentini [83], for instance, has introduced and explored the *edge covering problem*, where the guards are required to observe the edges of the polygon (metaphorically, the paintings on the walls of the art gallery and not necessarily every point of the gallery). According to O'Rourke [101], J. Malkelvitch and D. Wood independently suggested other two variants. In the first one they focused their attention on the visibility outside a polygon and, in the second one, the visibility is considered both inside and outside a polygon. The art gallery problem for the exterior of a polygon has been called the *fortress problem* (metaphorically, the polygon represents a fortress, and it is wished to see any enemy that comes closer). Likewise, when one wishes to see simultaneously both the inside and the outside of a polygon the problem is called the *prison-yard problem*, because one must watch both people outside, trying to break in, and people inside, trying to break out.

Another major research line derived from the original Art Gallery Problem has arisen when regions (inside a polygon), rather than just points, are considered as elements of guard sets. A point is called visible from a region if it is visible from a point in that region. This notion of visibility is known as *weak visibility*, to contrast with *strong visibility*, where a point is called visible from a region if it is visible from every point of that region [14]. Toussaint [14] was the first to propose these types of guards when he introduced the concept of *edge guards*, followed by the *mobile guards* of O'Rourke [104] and the *diagonal guards* of Shermer [118]. These three types of guards are not in stationary positions, they are allowed to patrol line segments on a polygon $P$: the edge guards, the individual edges of $P$; the mobile guards can patrol either an edge or a *diagonal* (a line segment between nonadjacent vertices) of the polygon and the diagonal guards are allowed to patrol diagonals.

Throughout the years, the developments in the study of visibility problems denote a progressive interest in addressing more realistic problems. In the works of Bose *et al.* [26], Estivill-Castro *et al.* [55], Abello *et al.* [6] and Steiger and Streinu [121] it is no longer considered that the light sources emit in all directions, or that the guards can patrol around themselves in all directions. They present illumination problems, in which the light sources have a restricted angle of illumination, which are called *floodlights*, and the respectively problems are known as *floodlight problems*. These ones are quite natural and they capture scenarios involving guards or security cameras with restricted angle of vision (for a comprehensive survey see [129]). More recent works on floodlights problems are due to Dietel et al. [41] and Cary et al. [32].

A recent variant of the art gallery problem was introduced by Aichholzer *et al.* [9] and Fabila-Monroy, Vargas and Urrutia [56]. In this new variant, the view element, instead of being modelled as a light source, is modelled as a wireless device whose signal can be transmitted through a given number $k$ of walls. The authors designated these wireless devices by $k$-modems. The parameter $k$ represents the strength of the signal emitted by the modem. This generalization of the art gallery problem can have a great potential in wireless networking applications.

Regarding the hiding problems, the initial problem can be viewed as the "inverse" problem of the original guarding problem: *given a polygon $P$, the problem is to find a hidden set of points on $P$ that is of maximum cardinality.* This problem was introduced by Shermer [117] in 1989. In the same work, he showed that this problem is $\mathcal{NP}$-hard, regardless of whether the hidden set is or not restricted to vertices, both on arbitrary and orthogonal polygons. He also obtained some combinatorial bounds for $h$, the maximum cardinality of any set of hidden points. In particular, he showed that: if $P$ is a polygon with $r$ reflex vertices then $h \leq r+1$. Shermer also showed that the maximum size of a hidden set of vertices on a polygon of $n$ vertices is at most $\lfloor \frac{n}{2} \rfloor$, being this bound tight as the previous one. On orthogonal polygons the upper bound is $\frac{n-2}{2}$, both for hidden points and hidden vertices, and this bound is reached on a special class of orthogonal polygons, designated by *staircase polygons*.

Later on, Hurtado [73] extended the notion of hiding points on other geometric configurations. If $F$ is a family of disjoint sets on the plane, a set of points $H$ is called a *hidden set for $F$* if it is contained in the complement of the union of the elements of $F$ and if the segment, joining any two points of $H$, intersects some set of $F$. If $F$ is a family of $n$ disjoint segments on the plane, Hurtado, Serra and Urrutia [74] proved that $F$ admits a hidden set of points of size at least $\sqrt{n}$ and that there are sets of segments that do not admit sets of hidden points of size greater than $2\sqrt{n}$. Hurtado [73] also obtained results on hiding points in families of triangles, rectangles and hexagons.

Eidenbenz [49] studied the three-dimensional version of the initial hiding problem: hiding points in polyhedra terrains. Eidenbenz was interested in finding the precise number and an optimal placement of people to be hidden, being given a terrain with $n$ vertices. He proved that this new variation is also $\mathcal{NP}$-hard, regardless of whether the hidden set is or is not restricted to vertices. Eidenbenz also studied the variation where it is allowed, as input, polygons with holes, which is also $\mathcal{NP}$-hard (independently of knowing if the set is limited to the vertices).

A combination of the two classic problems (original Art Gallery Problem and initial hiding problem) was also introduced and studied by Shermer [117]. This problem consists on finding a *hidden guarding set* of minimum cardinality, where a hidden guarding set is defined as a set of guard positions in a given polygon such that no two guards see each other and such that every point of the polygon is seen by at least one guard. The point and vertex variants of this problem gave rise to very different results. Shermer proved that every polygon admits a hidden guard set. However, if the hidden set is restricted to vertices, not every polygon admits such a set. Moreover, the determination that such a hidden guard set exists is $\mathcal{NP}$-hard. He also showed that finding a hidden guard set of minimum cardinality is $\mathcal{NP}$-hard. Later on, Eidenbenz [51] defined two variations of this problem by allowing polygons with or without holes or by letting the input be a polyhedra terrain.

## 1.2 Structure of the Thesis

As we can see, the original art galley problem gave rise to a huge amount of variants. Most of these guarding problems are either $\mathcal{NP}$-hard or it is strongly believed that they are $\mathcal{NP}$-hard (see e.g. [129]). On the other hand, as far as we know, the hiding problems, studied so far, are $\mathcal{NP}$-hard. As said before, for such problems there are usually two lines of investigation: (1) the development of algorithms that find approximate solutions or (2) the determination of optimal solutions on special classes of polygons.

In this dissertation, some of the visibility problems are studied following the first line of investigation and/or the second line of investigation. Although each one of these problems is formalized and explained in detail in the beginning of the respective chapter, a general summary is presented now. The studied guarding problems are: the MINIMUM VERTEX GUARD SET problem; the MINIMUM VERTEX FLOODLIGHT SET problem and the MINIMUM VERTEX $k$-MODEM SET problem. The studied hiding problems are: the MAXIMUM HIDDEN SET and the MINIMUM VERTEX GUARD SET problems.

**Minimum Vertex Guard Set problem.** The MINIMUM VERTEX GUARD SET problem asks for a minimum number of vertex guards needed to cover a given polygon $P$. A set $G \subset V_P$ of vertices of $P$ is called a *vertex guarding set* for $P$ if the vertices cover $P$. The MINIMUM VERTEX GUARD SET problem is denoted by MVGS($P$) and it is stated as follows:

**MVGS($P$)**
**Input**: A polygon $P$ with $n$ vertices.
**Question**: What is the minimum number of vertex guards necessary to cover $P$?

This problem is $\mathcal{NP}$-hard both for arbitrary polygons [84] and for orthogonal polygons [115].

**Minimum Vertex Floodlight Set problem.** Given an orthogonal polygon $P$, the MINIMUM VERTEX FLOODLIGHT SET problem consists on finding the minimum number of orthogonal floodlights (light sources with $\frac{\pi}{2}$ angle of illumination) placed on vertices of $P$ necessary to illuminate it. A given set $F \subset V_P$ of vertices of $P$ is called a *vertex orthogonal floodlighting set* (*vertex floodlighting set*, for short) for $P$ if the floodlights illuminate $P$. The MINIMUM VERTEX FLOODLIGHT SET problem is denoted by MVFS($P$) and it is formally defined as:

**MVFS($P$)**
**Input**: An orthogonal polygon $P$ with $n$ vertices.

**Question**: What is the minimum number of vertex orthogonal floodlights necessary to illuminate $P$?

It is strongly believed that the MVFS($P$) problem is $\mathcal{NP}$-hard [129].

**Minimum Vertex $k$-Modem Set problem.** A $k$-modem is a wireless modem whose signal strength is $k$, that is, its signal is strong enough to transmit a stable signal through at most $k$ walls along a straight line. Being $P$ a polygon, *a covering vertex $k$-modem set* is a set of $k$-modems, placed on vertices of $P$, such that the $k$-modems cover $P$. The MINIMUM VERTEX $k$-MODEM SET problem asks for a vertex $k$-modem set of minimum cardinality. This problem is denoted by MVkMS($P, k$) and it is stated as follows:

**MVkMS($P, k$)**
**Input**: A polygon $P$ with $n$ vertices and a number $k$ of walls.
**Question**: What is the minimum number of vertex $k$-modems necessary to cover $P$?

As for the MVFS($P$) problem, it is strongly believed that this problem is $\mathcal{NP}$-hard [9].

**Maximum Hidden Set problem:** Given a polygon $P$, two points on $P$ are called hidden points if they do not see each other. A set $H \subset P$ of points of $P$ is called a *hidden set* for $P$ if any two points of $H$ do not see each other. The MAXIMUM HIDDEN SET problem asks for a maximum number of hidden points on a given polygon $P$. The problem as denoted by MHS($P$) and it is stated as follows:

**MHS($P$)**
**Input**: A polygon $P$ with $n$ vertices.
**Question**: What is the maximum number of hidden points on $P$?

This problem is $\mathcal{NP}$-hard both for arbitrary and orthogonal polygons [117].

**Maximum Hidden Vertex Set problem:** If a hidden set $H$ is restricted to the vertices of $P$, $H$ is called a *hidden vertex set*. The MAXIMUM HIDDEN VERTEX SET problem is denoted by MHVS($P$) and it is formalized as follows:

**MHVS($P$)**
**Input**: A polygon $P$ with $n$ vertices.
**Question**: What is the maximum number of hidden vertices on $P$?

Like the MHS($P$) problem, this problem is also $\mathcal{NP}$-hard both for arbitrary and orthogonal polygons [117].

These problems are studied following the two lines of investigation stated above and, in this way, this dissertation is divided in two parts. In the first one, which is called *"Approximation Strategies for Visibility Problems"*, it is proposed approximation algorithms, based mainly on metaheuristics. In the second part, which is called "*Visibility Problems on Special Classes of Polygons*", it is studied and presented exact solutions on special classes of polygons.

The first part is divided in five chapters, from Chapter 2 to Chapter 6, and the second one consists of Chapters 7 and 8.

In Chapter 2 a brief introduction to approximation methods is presented. The general metaheuristics *simulated annealing* and *genetic algorithms*, which are the metaheuristic techniques used to tackle the problems under study, are succinctly described in section 2.1. This chapter ends with a short introduction to hybrid metaheuristics (section 2.2).

Chapters 3, 4, 5 and 6 deal with the MHVS($P$), MVGS($P$), MVFS($P$) and MVkMS($P$) problems, respectively. In each one of these chapters the corresponding problem is formally defined, approximation algorithms are proposed and experimental results are described.

In Chapter 3, the MHVS($P$) problem is described (section 3.1) and four approximation algorithms to determine a hidden vertex set, whose cardinality approximates the maximal number of hidden vertices on a given polygon $P$, are presented (section 3.2). The first two are greedy strategies designed specifically to solve the MHVS($P$) problem (subsection 3.2.1) and the other two are based on the simulated annealing and genetic algorithms metaheuristics (subsections 3.2.2 and 3.2.3, respectively). As the optimal solution for the MHVS($P$) problem is unknown, in section 3.3 it is developed a method (based on the determination of an approximate solution of the MINIMUM CLIQUE PARTITION problem) to determine an upper bound for it. This method allows to get the performance ratio of the developed approximation algorithms. Section 3.4 describes the experiments made over a large set of randomly generated polygons (arbitrary and orthogonal). In this section it is also presented a statistical comparative study of the obtained experimental results to select the best strategy. Then, using the selected method, more experiments were made in order to get some conclusions about its performance.

In Chapter 4 the MVGS($P$) problem is presented (section 4.1) and five approximation algorithms to tackle it are developed (section 4.2). The first is a greedy algorithm (subsection 4.2.2), the second is based on the simulated annealing metaheuristic (subsection 4.2.3), the third is based on the genetic algorithms metaheuristic (subsection 4.2.4) and the last two are hybrid algorithms, based on the simulated annealing and genetic algorithms metaheuristics (subsection 4.2.5). As the optimal solution for the MVGS($P$) problem is not known, in section 4.3 it is developed a method that allows to determine a lower bound for it. This method permits to get the performance ratio of the developed approximation strategies. Section 4.4 describes the experiments made over a large set of randomly generated polygons (arbitrary and orthogonal). In this section it is also presented a statistical comparative study of the obtained experimental results to select the best strategy. As before, the performance of this strategy is analyzed with more experiments.

In Chapter 5 the MVFS($P$) problem is described (section 5.1) and four approximation algorithms are developed (section 5.2). The first is based on the simulated annealing metaheuristic (subsection 5.2.2), the second is based on the genetic algorithms metaheuristic (subsection 5.2.3) and the last two are hybrid algorithms based on used metaheuristics (subsection 5.2.4). In section 5.3 it is presented a method to determine a lower bound for the unknown optimal solution. This method permits to get the performance ratio of the approximation algorithms. In section 5.4 the experiments made over a large set of randomly generated orthogonal polygons are described. After a statistical comparative study of the obtained experimental results, the best strategy is selected. With this strategy, approximate solutions to the MVFS($P$) problem are determined.

Finally, the last chapter of the first part deals with the problem of minimizing the number of $k$-modems necessary to cover a given polygon $P$, which is described in subsection 6.1. Note that, to tackle this problem, a necessary and fundamental step is to determine the *region covered by a $k$-modem located at a point $x$ on a polygon $P$*, denoted by $Vis_k(x, P)$. Unlike the determination of $Vis(x, P)$, and as far as we know, no algorithm has been developed so far to determine this region. Therefore, in first place, it is presented an algorithm to calculate the $k$-modem visibility region of a $k$-modem placed on $x \in P$, for any $k \in \mathbb{N}_0$ (section 6.2). Then, in section 6.3, an approximation algorithm is developed in order to determine a vertex $k$-modem set, whose cardinality approximates the minimal number of vertex $k$-modems needed to cover a given polygon $P$. This approximation algorithm is a hybrid metaheuristic technique that combines the genetic algorithms and simulated annealing metaheuristics. For this problem, the approximation solutions were obtained for 2-modems and 4-modems on monotone and non-monotone arbitrary polygons and orthogonal polygons (section 6.4).

The second part, Part II, is almost all devoted to the study of problems on orthogonal polygons. In Chapter 7 a subclass of orthogonal polygons - the *grid $n$-ogons* - is addressed. These polygons were defined by Tomás and Bajuelos [24, 124] and they appear to exhibit interesting characteristics. Besides, they were used experimentally to evaluate approximation methods for the resolution of some guarding problems [36, 37, 126]. In section 7.1 some definitions and known results related to them are briefly presented. Section 7.2 is devoted to the study of new results related to this kind of polygons, mainly related to structural properties. In section 7.3, for some special classes of polygons, the optimal solution of the following problems is determined: MVGS($P$), where $P$ is a FAT grid $n$-ogon, a MIN-AREA grid $n$-ogon and a SPIRAL grid $n$-ogon and MHVS($P$), where $P$ is a THIN grid $n$-ogon. In Chapter 8 the MHVS($P$) and MHS($P$) problems, where $P$ is a spiral or a histogram polygon, are studied.

Finally, general conclusions and future work are discussed in Chapter 9.

Some of the presented results were obtained in collaboration with other authors and they have led to several publications and conference presentations. In particular, the material of Chapter 3 appeared in [17]. The content of Chapter 4 was presented in [21, 22]. The subject of Chapter 6 was presented in [23]. The content of Chapter 7 appeared in [16, 18, 19, 91–94] and [20] is related to Chapter 8.

# Part I

# Approximation Strategies for Visibility Problems

# Introduction

This part of the dissertation is divided in five chapters. In the first chapter it is made a short introduction to approximation methods. Each of the other chapters is dedicated to the study of a visibility problem. The addressed problems are: the Maximum Hidden Vertex Set problem, MHVS($P$) (Chapter 3); the Minimum Vertex Guard Set problem, MVGS($P$) (Chapter 4); the Minimum Vertex Floodlight Set problem, MVFS($P$) (Chapter 5); and Minimum Vertex $k$-Router Set problem, MVkMS($P, k$) (Chapter 6), where $P$ is an arbitrary or an orthogonal polygon. These four problems are $\mathcal{NP}$-hard or it is strongly believed that they are $\mathcal{NP}$-hard. This means that finding exact and efficient methods to solve them is very unlikely. Thus, this part of the dissertation is devoted to the study of approximation algorithms, mainly based on the general metaheuristics simulated annealing and genetic algorithms, to tackle these problems. In this way, following the first line of investigation proposed for problems with this computational complexity (see Chapter 1).

The proposed algorithms were implemented in C/C++, using the MS Visual Studio 2005, on top of the Computational Geometric Algorithms Library (CGAL) 3.2.1 [2]. The CGAL package mostly used was the $2D$ Regularized Boolean Set-Operations package [60] and it was decided to use exact arithmetic (which is extremely slow), since the floating-point arithmetic in spite of being faster than the exact arithmetic, leads to precision problems (see Chapter 1). In the performed implementations, a special attention was given to the treatment of degenerate cases that often are not treated in the theoretical papers. As an example, a particular attention was given to the existence of three or more collinear points in a set of points, since it is a situation that arises in orthogonal polygons very often.

The computational tests were performed on a PC featuring a Intel(R) Core (TM)2 CPU 6400 at 2.66 GHz and 1 GB of RAM. For each problem, extensive experiments were conducted with the algorithms developed to tackle it. These experiments were performed on a large set of randomly generated polygons, arbitrary and orthogonal. The arbitrary polygons were generated using the CGAL's function *random_polygon_2*. According to [69], the implementation of this function is based on the method of eliminating self-intersections in a polygon by using the so-called "2-opt" moves. Such a move eliminates an intersection between two edges by reversing the order of the vertices between the edges. No more than $\mathcal{O}(n^3)$ of these moves are required to simplify a polygon defined on $n$ points [86]. Intersecting edges are detected using a simple sweep through the vertices and then one intersection is chosen at random to eliminate after each sweep. Therefore, the worse-case running time is $\mathcal{O}(n^4 \log n)$. To generate orthogonal polygons it was used the polygon generator developed by Joseph O'Rourke (personal communication 2002).

For each problem (and consequently, in each chapter) there is a section where it is studied which of the developed approximation methods obtains the best solutions in a reasonable time. However, the simple comparison of two or more values (e.g., averages, medians) might be different according to the statistical distributions. So, it is necessary to perform a general statistical study to ensure the maximum statistical power of the results, i.e., determining whether the conclusions are meaningful and not just noise [10]. In order to obtain that, in first place, it was applied the Kolmogorov-Smirnof test to check the data normality. Since, in all performed studies the distribution of the data sets was found not to be normality distributed, the statistical tests that were used then were the non-parametric. Since it was intended to perform several independent group comparisons, it was used the Kruskal-Wallis test. This test permits to ensure statistical difference in the results, with a higher statistical power than ANOVA when data do not come from a normal population [90, 132]. When at least a data sample is significantly different than the others, multiple comparison tests were used to determine which pairs of results were significantly different, and which were not. A significance level of 0.05 was used for all tests.

# Chapter 2

# Approximation Methods

In the forthcoming chapters approximate solutions are presented, metaheuristic solutions included, for geometric optimization on visibility problems for which it is either known or strongly believed they are $\mathcal{NP}$-hard. Therefore, in this section it is presented a brief introduction to metaheuristics and hybrid metaheuristics techniques.

## 2.1 Metaheuristics

A *combinatorial optimization problem* consists of [1]: given a discrete, finite or countably infinite, *solution* (*or search*) *space* $S$, where each element, called *candidate solution*, is an admissible solution to the problem and an *objective function* $f : S \to \mathbb{R}$, determine $x^* \in S$ such that $f(x^*) \leq f(x)\ \forall x \in S$ (minimization problem) or $f(x^*) \geq f(x)\ \forall x \in S$ (maximization problem). In minimization problems, $x^*$ is called a *globally minimal solution* (or *global minimum*) and in maximization problems, $x^*$ is called a *globally maximal solution* (or *global maximum*). A global minimum/maximum is also called a *globally optimal solution* or simply an *optimal solution*. The set of all globally optimal solutions is denoted by $S^*$.

Frequently, these combinatorial problems are easy to state but very difficult to solve. Due to its practical importance, many algorithms have been developed to tackle them, which can be classified as either *exact algorithms* or *approximate algorithms*. Exact algorithms are guaranteed to find an optimal solution for every finite size instance of a combinatorial optimization problem. Yet, many of the interesting and important combinatorial optimization problems are $\mathcal{NP}$-hard, that is, it is strongly believed that no polynomial time algorithm exists to solve it to optimality. For a detailed study of computational complexity see Garey and Johnson [61]. In the case of $\mathcal{NP}$-hard problems, and in the worst case, exponential time is necessary to find the optimal solution. Thus, for $\mathcal{NP}$-hard problems, the performance of exact algorithms is not satisfactory. In this way, for practical purposes, if optimal solutions cannot

[1]Other definitions are possible, which are equivalent to the one described here.

be obtained in a reasonable computational time, one of the possibilities to tackle the problem is to sacrifice the guarantee of finding optimal solutions in exchange for getting good solutions in a reasonable computational time. Consequently, the use of approximate algorithms to solve combinatorial optimization problems has received more and more attention in the last years. Approximate algorithms, often also called *heuristic methods* or simply *heuristics*, seek to obtain "good" solutions, that is, near-optimal solutions at relatively low computational cost.

According to Blum and Roli [25], among the basic heuristic methods, it is usual to distinguish between *constructive methods* and *local search methods*. Constructive algorithms generate solutions by adding to an initially empty partial solution, components until a solution is complete. The *greedy heuristics* are a well-known class of this type of heuristics, characterized by taking always the best immediate, or local, solution while finding an answer. Constructive algorithms are usually faster than local search heuristics, but frequently they return worse solutions. Local search algorithms start from some initial solution and iteratively they try to replace the current solution by a better solution in a properly defined *neighbourhood* of the current solution. A neighbourhood is formally defined as [29]:

**Definition 2.1** *A* ***neighbourhood structure*** *is a function $\mathcal{N} : S \rightarrow 2^{|S|}$ that assigns to every $s \in S$ a set of neighbours $\mathcal{N}(s) \subset S$. $\mathcal{N}(s)$ is called the neighbourhood of $s$.*

Often, neighbourhood structures are implicitly defined by specifying the changes that must be applied to a solution $s$ in order to generate all its neighbours. The application of such an operator that produces a neighbour $s' \in \mathcal{N}(s)$ of a solution $s$ is commonly called a *move*. After defining a neighbour structure it is possible to define *locally solutions*.

A *locally minimal solution* (or *local minimum*) with respect to a neighbourhood structure $\mathcal{N}$ is a solution $\hat{x}$ such that $f(\hat{x}) \leq f(y)\ \forall y \in \mathcal{N}(\hat{x})$. If $f(\hat{x}) < f(y)\ \forall y \in \mathcal{N}(\hat{x})$, $\hat{x}$ is called a *strict locally minimal solution*. A *locally maximal solution* (or *local maximum*) with respect to a neighbourhood structure $\mathcal{N}$ is a solution $\hat{x}$ such that $f(\hat{x}) \geq f(y)\ \forall y \in \mathcal{N}(\hat{x})$. In a similar way, if $f(\hat{x}) > f(y)\ \forall y \in \mathcal{N}(\hat{x})$, $\hat{x}$ is called a *strict locally maximal solution*. A local minimum or a local maximum is also called a *locally optimal solution* or a *local optimum*.

The most simple local search method works by iteratively improving a given solution $x$ by choosing a solution $y$ from $\mathcal{N}(x)$, as long as possible. A general scheme of an iterative minimization local search algorithm is illustrated in Algorithm 2.1. The weak point of this method is that it strongly depends on the initial generated solution and it stops as soon as it finds a local optimum. Thus, it often cannot find good locally optimal solutions. As a consequence, its performance is usually quite unsatisfactory. A deterministic iterative minimization local search algorithm partitions the search space $S$ into *basins of attraction* of local minima. The basins of attraction of local minimum $\hat{x} \in S$ is the set of all solutions $S^* \subset S$

for which the search terminates in $\hat{x}$ when it starts in an element $x \in S^*$.

**Algorithm 2.1** Iterative (minimization) local search algorithm

Generate an initial solution $x \in S$
**repeat**
  Generate $y \in \mathcal{N}(x) \subset S$
  Evaluate $\delta = f(y) - f(x)$
  **if** $\delta < 0$ **then**
    $x \leftarrow y$
  **end if**
**until** $f(x) \leq f(y), \forall y \in \mathcal{N}(x)$

A different type of approximation methods, which attempts to bypass these problems was proposed. These methods try to combine basic heuristics in higher level framework, in order to efficiently and effectively explore a solution space, and are usually called *metaheuristics* [29]. The term metaheuristic derives from the two Greek words *heuristic* and *meta*: *heuristic* derives from the verb *heuriskein*, which means "to find", while the suffix *meta* means "beyond, in an upper level". Since the metaheuristic concept is very general, it is very difficult to give a precise definition of what a metaheuristic exactly is. However, an usually and accepted definition is: "*A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. In other words, a metaheuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem specific heuristic (e.g., local search algorithm or constructive heuristic) toward promising regions of the solution space containing high-quality solutions. A metaheuristic is therefore a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem*" [44]. Examples of metaheuristics include *Ant Colony Optimization* (ACO), *Genetic Algorithms* (GAs), *Iterated Local Search* (ILS), *Simulated Annealing* (SA), and *Tabu Search* (TS).

There are two very important concepts in metaheuristics, called *intensification* and *diversification*. Intensification usually refers to the carefully and intensively search around a good solution (that is, the exploitation of good solutions), while diversification refers to the ability to guide the search to unvisited regions (that is, the exploration of the search space). A good balance between these two goals is important because a search should intensively explore areas of the search space with high quality solutions and move to unexplored areas of the search space when necessary (see, e.g., [29,87]).

It is usual to classify metaheuristics according to the number of solutions used at the same time (for other classifications see, e.g., [25]). This classification divides metaheuristics into *trajectory methods* and *population-based methods*. Trajectory methods are algorithms that work on a single solution at any time; they share the property that the search process describes a trajectory in the solution space. These methods include all metaheuristics that are based on local search, such as simulated annealing search, tabu search and iterated local search. Population-based metaheuristics, on the contrary, perform search processes which can be described as the evolution of a set of solutions (as for example in genetic algorithms). This classification allows a clear description of the algorithms. Currently, there is a trend to the hybridization of methods in the direction of the integration of single point search algorithms in population-based ones.

In this dissertation two metaheuristics are used: the simulated annealing metaheuristic and the genetic algorithms metaheuristic; the first one is a trajectory method, while the second one is a population-based method. Besides, the combination of these two metaheuristics is performed resulting in hybrid metaheuristics. In subsections 2.1.1 and 2.1.2 the main principles of simulated annealing and genetic algorithms metaheuristics are described (for a detailed introduction to these and other metaheuristics see, for instance, [65]). In subsection 2.2 it is presented a brief overview on hybridization of metaheuristics.

### 2.1.1 Simulated Annealing

Simulated Annealing (SA) is usually known as being the oldest among the metaheuristics. This strategy is based on the analogy between simulation of the annealing of solids and the problem of solving large combinatorial optimization problems. Metropolis et al. [98] developed a method, which simulates the evolution to thermal equilibrium of a solid for a fixed value of the temperature. This inspired Kirkpatrick et al. [80] and, independently, Černý [130] to develop the SA method as a (trajectory) local search algorithm to solve combinatorial optimization problems. Without loss of generality, the description of the method is going to be done assuming minimization problems. It is commonly said that SA is one of the first algorithms that has an explicit strategy to escape from locally optimal solutions (see, e.g. [25]). For that, SA introduces a control parameter $T$, designated by *temperature*, whose initial value should be high and should decrease during the search process. The search process is done according to the execution of several iterations of the algorithm until a *termination condition* is achieved. The control parameter $T$ allows, with a certain *probability*, moves to solutions $y \in S$ whose objective function (or cost function) values are worse than the objective function value of the current solution $x \in S$, called *uphill moves*. This probability is usually called *acceptance function* and it is evaluated according to (see, e.g., [29]):

$$p(T, x, y) = e^{-\frac{\delta}{T}}, \quad \text{where} \quad \delta = f(y) - f(x). \tag{2.1}$$

By equation 2.1, we can see that the probability of accepting uphill moves is controlled by two factors: $T$ and $\delta = f(y) - f(x)$. On one hand, at a fixed $\delta$, the lower $T$ is, the lower the probability of uphill moves. Thus, along the search process (i.e., along the algorithm iterations) it is more difficult to accept uphill moves. On the other hand, at a fixed temperature, the higher $\delta$ is, the lower the probability to accept a move from $x$ to $y$. Note that, to equal values of $\delta$, the probability of accepting uphill moves is greater for high values of $T$. In the limit case $T = \infty$, any worsening in the objective function is accepted and, in the limit case $T = 0$, any increase in the objective function is rejected and the algorithm would have the normal scheme of an iterative local search algorithm. The basic outline of the SA algorithm is illustrated in Algorithm 2.2.

**Algorithm 2.2** Simulated Annealing Algorithm (for a minimization problem)

Generate an initial solution $x \in S$
Set the initial temperature $T_0$
$k \leftarrow 0$
**repeat**
    **for** $i = 1$ to $N(T_k)$ **do**
        Generate $y \in \mathcal{N}(x) \subset S$
        Evaluate $\delta = f(y) - f(x)$
        **if** $\delta < 0$ **then**
            $x \leftarrow y$ // $y$ replaces $x$
        **else**
            $x \leftarrow y$ with probability $p(T_k, x, y, )$ // see Equation 2.1
        **end if**
    **end for**
    $k \leftarrow k + 1$
    Decrease temperature $T_k$
**until** termination condition met

The algorithm starts by generating an initial solution $x \in S$, which can be randomly or heuristically constructed, and by initializing the temperature value $T_0$. Being $N(T_k)$ the number of iterations for temperature $T_k$, at each one a new solution $y \in \mathcal{N}(x)$ is randomly generated. If $y$ is better than $x$, then $y$ is accepted as the current solution. Otherwise (the move from $x$ to $y$ is an uphill move), $y$ is accepted with a probability computed according to Equation 2.1. Usually, the implementation of the acceptance function is done as follows: a real number $u$ is randomly generated, following the $U(0, 1)$ distribution; if $u \leq p(T, x, y)$, then the new solution $y$ is accepted as the current solution, otherwise $y$ is rejected. Finally, the value of $T_k$ is decreased at each algorithm iteration $k$. The algorithm continues this way until the termination condition is met.

Regarding the search process, we can see that the algorithm is the result of two combined strategies: random walk and iterative improvement [25]. In the first phase of the search (random walk), the bias toward improvements is low and it is permitted the exploration of the search space. However, this behaviour is slowly decreased, leading the search to converge to a (local) minimum (iterative improvement).

Therefore, given an optimization problem it is necessary to adapt it to the SA scheme, which is obtained by specifying the following parameters [30]:

1. **Specific Parameters** (of the problem):
   - *representation of the solution space*;
   - *objective function*;
   - *neighbourhood of a solution*;
   - *initial solution.*
2. **Generic Parameters** (of the SA strategy):
   - *initial temperature* ($T_0$);
   - *temperature decrement rule*;
   - *number of iterations at each temperature* ($N(T_k)$);
   - *termination condition.*

It is presented, next, some practical issues concerning the aforementioned parameters.

**Specific Parameters**

For a combinatorial optimization problem the *solution space* and the *objective function* are already defined. Thus, depending on the problem, the representation of its admissible solutions should be chosen in a way that it is easy to apply the SA strategy. Concerning the objective function, the SA needs to calculate this function for each new generated solution. In the interest of the overall computational efficiency, it is important that the evaluations of the objective function should be performed efficiently, especially because in many applications this evaluation is computationally intensive [110]. Even when the objective function is computed efficiently, the large number of new solutions required by the SA method can consume a vast amount of CPU time. According to [67], there are various techniques to speed-up this evaluation. One of these techniques, called *surrogate function swindle*, approximates the difference in the objective functions $\delta$ instead of calculating both $f(x)$ and $f(y)$. Sometimes it is also possible to calculate $f(y)$ from the cost of a current solution $f(x)$ or, if it is very expensive to determine the cost of a solution, a estimation of that cost can be obtained.

The *neighbourhood of a solution* must be done carefully because the solution generator must, on one hand, introduce small random changes and, on the other hand, allow all admissible solutions to be achieved. But, the generation mechanism is necessarily problem-specific and it must be compatible with the chosen representation of the solutions. For combinatorial problems, it is common to permute a small, randomly chosen, part of the solution [110].

The construction of the *initial solution* should be fast and, if possible, the initial solution should be a "good" starting point. Often, the fastest way of producing an initial solution is to generate it randomly. Another possibility is to adopt constructive heuristics such as greedy heuristics (specific to the problem) [25, 29].

### Generic Parameters

This set of parameters is called, by some authors, *annealing schedule* or *cooling schedule* (see, e.g., [3, 110]).

"*The annealing schedule determines the degree of uphill movement permitted during the search and is, thus, critical to the algorithm's performance. The principle underlying the choice of a suitable annealing schedule is easily stated - the initial temperature should be high enough to "melt" the system completely and should be reduced towards its "freezing point" as the search progresses - "but choosing an annealing schedule for practical purposes is still something of a black art"*" [110].

Annealing schedules can be grouped into two broad classes: static and dynamic schedules (see, e.g., [3]). In a static cooling schedule the parameters are fixed, that is, they cannot be changed during execution of the algorithm. In a dynamic cooling schedule the parameters are adaptively changed during execution of the algorithm. As mentioned before, the generic parameters are the initial temperature ($T_0$), the temperature decrement rule, the number of iterations at each temperature ($N(T_k)$) and the termination condition.

*Initial temperature* ($T_0$)

In general, it might be said that one of the properties that must be verified by any search method is that it should not be dependent of the initial solution. In this way, it would be advisable that simulated annealing starts from a high initial temperature, with which it will go through solutions different from the initial one (random walk). Nevertheless, it does not seem suitable to consider, for $T_0$, fixed values regardless of the problem. In this sense, the literature advises to carry on different analysis to choose $T_0$, since its value may depend, to a large extent, on the problem to solve [30]. For instance, for the MINIMUM VERTEX GUARD SET problem where the input is a $n$-vertex polygon, it could be considered:

1. an initial temperature that is dependent on the number of vertices of the polygon: $T_0 = f(n)$, where $n$ is the number of vertices of the polygon; or
2. a constant initial temperature: $T_0 = c$, where $c \in \mathbb{R}^+$ is chosen from some previous experimental tests.

*Temperature decrement rule*

The choice of an appropriate temperature decrement rule is crucial for the performance of the algorithm. The temperature decrement rule defines the value of $T$ at each algorithm iteration $k$, $T_{k+1} = Q(T_k, k)$, where $Q(T_k, k)$ is a function of the temperature and of the algorithm iteration number [25].

One temperature decrement rule of great theoretical interest is the one that follows a logarithmic law: $T_{k+1} = \frac{T_0}{\log(1+k)}$, usually designated by Classical Simulated Annealing (CSA). This logarithmic law guarantees the converge to an optimal solution [62]. Unfortunately, it is not feasible in applications, because it is too slow for practical purposes. Its convergence time is likely to exceed the time needed to find an optimal solution by exhaustive search. Therefore, faster temperature decreases are adopted in applications. For instance, Szu and Hartley [122] proposed a fast convergence, called Fast Simulated Annealing (FSA), using the temperature decrement rule $T_{k+1} = \frac{T_0}{1+k}$, which was improved by Ingber [75] using the temperature decrement rule $T_{k+1} = \frac{T_0}{e^k}$, called Very Fast Simulated Annealing (VFSA).

Yao [135] proposed a new temperature decrement rule $T_{k+1} = \frac{T_0}{\exp(e^k)}$, designated by New Simulate Annealing (NSA). However, this temperature decrease has the opposite drawback of the CSA decrease, since it converges so fast that it does not do an "exhaustive" search of the optimal solution.

Finally, according to many authors, among the different temperature decreases that appear in the literature, one of the most used follows a geometric law: $T_{k+1} = \alpha T_k$, where $\alpha \in [0, 1]$.

*Number of iterations at each temperature* $(N(T_k))$

According to the literature, a constant number of iterations at each temperature is the most used. Alternatively, the number of iterations can be dynamically changed. For example, at high temperatures a large value of iterations can be used to explore the search space and, at low temperatures, a smaller value of iterations can be used.

*Termination condition*

Based on various SA experiments for solving optimization problems, different types of termination conditions are proposed in the literature. Some of them regard the value of the temperature. Theoretically, the search process should stop when a *frozen state* is reached, i.e., when $T_k = 0$. However, much before reaching this value, the probability $e^{-\frac{\delta}{T}}$ to accept a move to a worse solution is practically null. As a result it is possible, in general, to finish the search with a final temperature, $T_f$, greater than zero without quality loss in the solution. Clearly, the lower the final temperature is, the closer the final solution will be to the optimum. In this case, however, the response time of the algorithm increases considerably [30].

Another type of termination conditions considers the final temperature by fixing the number of temperatures values to be used or the total number of solutions to be generated. Another possible termination criteria is to halt the search when it ceases making progress. Lack of progress can be defined in very different ways but a frequently basic definition is: no improvement (i.e., no new best solution) registered in the last $l$ consecutive values of temperatures combined with the acceptance ratio of solutions falling below a given (small) value $\varepsilon$ [28, 110].

### 2.1.2 Genetic Algorithms

Genetic Algorithms (GAs) were proposed by Holland [70] and are population-based search methods. They are a particular class of *evolutionary algorithms* that use techniques/concepts inspired by evolutionary biology, such as, *chromosomes*, *genes*, *selection* and *crossover* also known by *recombination*, to solve optimization problems. GAs are implemented as a computer simulation in which a population of abstract representations of candidate solutions of an optimization problem evolves toward better solutions. Each representation is called *individual*, *chromosome* or *genotype* and a solution is called *phenotype* (see, e.g., [82]). The evolution usually starts from an initial population of randomly generated individuals. These individuals are evaluated with a function that indicates the adaptation degree of the individual to the environment (in biological terms) or it is an indicator of the solution quality (in optimization terms). From this initial situation several iterations are made in each of those a new population is generated from the previous one, by applying the genetic operators *selection*, *crossover* and *mutation* on the individuals (described later). Commonly, the algorithm terminates when either a given number of generations was produced or a satisfactory fitness level was reached for the population. The final population, if the algorithm converges properly, will be composed of good individuals, the best of these is the solution achieved by the algorithm (see Figure 2.1). As in the previous subsection, the description of the method is done, without loss of generality, assuming minimization problems.

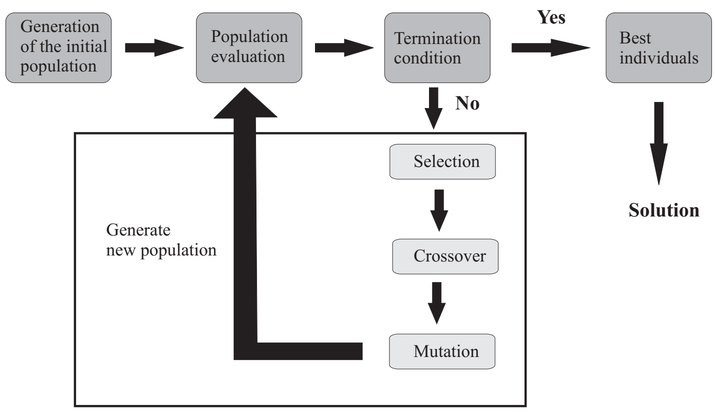



Figure 2.1: General scheme of a genetic algorithm.

GAs have numerous variants due to different parametrization settings and implementations. The structure of a basic Genetic Algorithm (GA) is illustrated in Algorithm 2.3.

**Algorithm 2.3** Basic GA

1. $t \leftarrow 0$
2. Initialize $P(t)$
3. **while** termination condition not met **do**
4. Evaluate $P(t)$
5. Selection on $P(t)$
6. Recombine $P(t)$
7. Mutate $P(t)$
8. $t \leftarrow t + 1$
9. Generate $P(t)$ from $P(t-1)$
10. **end while**

Taking into account the above, a genetic algorithm has the following components/parameters [30]:

1. Genetic representation of the candidate solutions to the problem - *Encoding*
2. Creation of an initial population of admissible solutions - *Initial population.*
3. Definition of a function to evaluate the individuals and make the effect of natural selection, by sorting the solutions according to their "strength" - *Fitness function.*
4. Genetic operators to change the composition of the solutions - *Selection*, *Crossover*, and *Mutation.*
5. Definition of various parameters used by the genetic algorithm (e.g., population size, probability of the genetic operators, population evaluation, population generation, termination condition).

Some practical issues concerning these components will follow.

*Encoding*

It is necessary to define an abstract genetic representation to the admissible solutions of the problem. As said before, these representations are called individuals, chromosomes or genotypes, whereas the solutions that are encoded by individuals are called phenotypes. This is to differentiate between the representation of solutions and solutions themselves, what is one of the distinctive features of the GA approach. The most used representations of solutions are *strings* made up from an *alphabet* $\mathcal{A}$. The elements of these strings are called *genes.* According to [112], the representation of the solutions by binary strings ($\mathcal{A} = \{0, 1\}$) is in some sense the best one and, although this idea has been changed, it is still frequently convenient from a mathematical point of view. Although in many applications other representations are possible and used, it is the binary case which is assumed in the sequel.

*Initial population*

The population for a given generation consists of a set of individuals. The major questions to consider are firstly the size of the population and, secondly, the method by which the individuals are chosen [112]. Concerning the size of the population, the idea is to get a compromise between efficiency and effectiveness. The total number of individuals in each population has to be large enough to ensure sufficient room for exploring the search space effectively, but not so much that might damage the efficiency of the algorithm in a way that no solution is achieved in a reasonable amount of time. Most GAs algorithms work with populations of fixed size having in consideration this compromise. Concerning how the population is chosen, according to [112], it is nearly almost assumed that initialization should be done randomly. However, some works showed that including high quality solutions in the initial population can help a GA to find better solutions faster than a randomly generated initial population.

*Fitness Function*

The fitness function should help to make the selection of the best individuals to be reproduced (i.e., to be recombined) and it should assign higher values to the solutions closer to the optimal one(s). The value of the fitness function can be related to the value of the objective function or some other kind of quality measure [25]. The evaluation of the fitness function for complex problems is often the most expensive part of GAs. In real world problems one single function evaluation may require several hours, or even several days, of complete simulation. In this case, it may be necessary to replace an exact evaluation for an approximated fitness that is computationally efficient.

*Genetic operators* (*selection*, *crossover*, and *mutation*)

The *selection* operator should choose the best individuals to be reproduced. This operator does not produce new individuals. It determines which are the individuals that will leave offspring, and in what amount, for the next generation [30]. Many selection methods have been proposed to accomplish this idea, including *roulette-wheel selection*, *stochastic universal selection*, *ranking selection* and *tournament selection*. Since the roulette-wheel selection and the tournament selection are the most used, they are described below. For details about other selection methods the reader must refer to [76, 112, 133], for instance.

The basic idea of selection is that it should base their decisions on the fitness of the individuals. The original scheme, and the most commonly known, is the roulette-wheel method. In this method the individuals are given a probability of being selected that is proportional to their fitness. Then, based on these probabilities, $k$ individuals are randomly chosen to be the reproduction candidates (i.e, parents in crossover) [112]. The roulette wheel selection scheme can be implemented as follows [76].

1. Evaluate the fitness, $f_i$ , of each individual $i$ in the population.
2. Compute the probability (slot size), $p_i$ , of selecting each member of the population $p_i = \frac{f_i}{\sum_{j=1}^{M} f_j}$, where $M$ is the population size.
3. Calculate the cumulative probability, $q_i$ , for each individual: $q_i = \sum_{j=1}^{i} p_j$ .
4. Generate a uniform random number, $r \in (0, 1]$. If $r < q_1$ then select the first individual, $x_1$; else select the individual $x_i$ such that $q_i - 1 < r \leq q_i$.
5. Repeat step 4 $k$ times to create $k$ reproduction candidates (i.e., $k$ parents to be used in crossover).

To illustrate the method, imagine a roulette wheel where all the individuals of the population are placed. To each individual, a roulette wheel section is assigned, whose size is proportional to the individual fitness. The bigger the fitness value, the larger the section size. To select an individual, the wheel is spined and it stops in a section. The individual that is assigned to that section is chosen to be a parent in the crossover. This procedure is repeated until $k$ individuals have been selected [76, 133].

Table 2.1 and Figure 2.2 provide a simple example of the roulette wheel selection method. In this example, there is a population with five individuals ($M = 5$) with fitness values $\{32, 9, 17, 17, 25\}$, respectively. If, for example, the numbers 0.13 and 0.68 are randomly generated, the individuals 1 and 4 are selected.

| Individual # | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Fitness, $f_i$ | 32 | 9 | 17 | 17 | 25 |
| Probability, $p_i$ | 0.32 | 0.09 | 0.17 | 0.17 | 0.25 |
| Cumulative probability, $q_i$ | 0.32 | 0.41 | 0.58 | 0.75 | 1.00 |

Table 2.1: Example of the roulette wheel selection method.

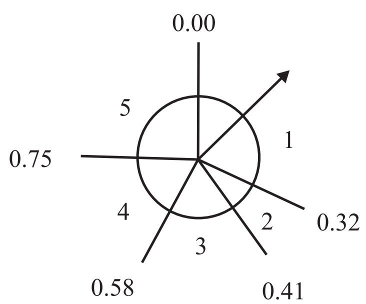



Figure 2.2: Example of the roulette wheel selection method (from [112]).

Another used selection scheme is the tournament selection (according to [133] this is one of the most popular and effective selection schemes). In tournament selection, $m$ individuals are randomly picked from the population and compared with each other in a tournament. The individual with the highest fitness value in the group of the $m$ individuals is selected as the parent. The most widely used value of $m$ is 2 [76]. Using this selection scheme, $k$ tournaments are required to choose $k$ individuals.

After selection, the selected individuals are recombined (or crossed over) to create new, hopefully better individuals, called *children*. *Crossover* is the recombination of two parents to produce new individuals (it is also possible to recombine more than two parents, but is more unusual) [25]. It operates on selected genes from parent individuals and creates new child individuals.

The crossover of two selected individuals is simply a matter of replacing some of the genes in one parent for the corresponding genes of the other. In the GAs literature, many crossover methods have been proposed, but several of them are problem-specific. In this section a few generic (problem independent) crossover operators, such as *single point crossover* or *one-point crossover*, *two-point crossover* and *uniform crossover* are introduced.

Suppose that there are 2 parents $A$ and $B$, each one having 6 genes, that is, $A = a_0a_1a_2a_3a_4a_5$ and $B = b_0b_1b_2b_3b_4b_5$, with $a_i, b_i \in \{0, 1\}$. In single point crossover, a randomly selected position (gene) of the two parents, called *crossover point*, is chosen and the parents are split at that crossover point. Finally, two children are created by exchanging the parents tails. Assuming that the crossover point is 3, the children are the strings $C_1 = a_0a_1a_2a_3b_4b_5$ and $C_2 = b_0b_1b_2b_3a_4a_5$. In two-point crossover, two crossover points are randomly selected and the fragment between these two points is exchanged with the corresponding fragment of the second individual. Assuming that the crossover points are 1 and 3, the children are $C_1 = a_0a_1b_2b_3a_4a_5$ and $C_2 = b_0b_1a_2a_3b_4b_5$. A $m$-point crossover, with $m > 1$, can be defined in a similar way. Notice that, from the two generated children we can consider only one.

Uniform crossover is an operator that decides (with a given probability) which parent will contribute to each of the gene values of the child chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as in the case of single and two-point crossover). If the probability is 50%, approximately half of the genes of the child are inherited from one parent and the other half from the other.

Finally, according to [112], one of the keys to good performance is to maintain the diversity of the population as long as possible. A popular technique to achieve this goal is not to allow children that are merely clones of the parents (i.e., copies of the parents). The disadvantage of this technique is the need to compare each parent with the new children, what has computational efforts. However, some steps can be performed to reduce the chance of cloning before the children are generated. For example, with single point crossover, the two strings 1101001 and 1100010 will only generate clones if the crossover point is any of the first three positions. So, before applying crossover, the selected parents should be examined to find suitable crossover points. This requires to compute an "exclusive-OR" (XOR) string between the parents so that only positions between the outermost 1s of the XOR string should be considered as crossover points. In the example above, the XOR string is 0001011 and, as previously stated, only the crossover points 3, 4 and 5 will give rise to a different string.

To conclude, it is necessary to say that the crossover does not occur always, it only occurs with a certain probability, $p_c$. The value of $p_c$ usually is set experimentally and it has, in general, a high value.

Finally, The *mutation operator* causes self-adaptation of individuals. In a binary representation, the action of mutation is relatively simple. It merely flips a randomly selected binary digit (or each) from zero to one or vice versa, with a certain probability $p_m$ [112]. The value of $p_m$ is usually set experimentally and should generally be a fairly low value.

It is often suggested that in GAs the simplest mechanism to diversify the population, and to ensure that it is possible to explore the entire search space, is the mutation operator. As stated above, the simple form of a mutation operator just performs a small random perturbation of an individual, introducing a kind of "noise" (see, e.g., [25]).

Several authors have suggested some adaptive mutations, for example, the use of different $p_m$ values at different gene positions and the variation of $p_m$ values according to the diversity in the population (measured in terms of the fitness variation).

*Population evaluation*

The evaluation of a population, i.e., the *population fitness*, measures the improvement of the genetic algorithm [78]. There are several ways to calculate it, an usual one is to consider the minimum value of the fitness function when applied to all individuals of the population [30], that is, designating the population fitness by $F(P(t))$,

$$F(P(t)) = \min\{f(x_1), f(x_2), \ldots, f(x_M)\},$$

where $f$ denotes the fitness function and $M$ the population size.

*Population generation*

After the new offspring solutions (children) are created, using crossover and mutation, there is the need to place them into the parental population. It is expected that these children are among the fittest ones in the population, since the parent individuals have already been selected according to their fitness. In this way, it is expected that the population will gradually, on average, increase its fitness. Some of the most common replacement or population generation techniques are outlined below (it was adopted the overview given by Reeves [112], which is, in our opinion, a good overview).

The original population generation is a *generational* approach, in which selection, crossover and mutation are applied to a population with $M$ individuals until a new set of $M$ individuals are generated. This set became the new population. Subsequently, the *elitist* and *population overlaps* approaches were introduced. The first one ensures the survival of the best individual obtained so far, by preserving it and replacing the remaining $(M-1)$ individuals with new strings. In the second one, a fraction $G$, called the *generation gap*, of the population is replaced at each generation. Finally, the well-known *steady-state* reproduction has been proposed, in which only one new (or sometimes two) individual is generated at each iteration. In this case it is necessary to select the individuals of the population to be deleted. Some GAs assume that parents are replaced by their children, others delete the worst member of the population, and, finally, another approach deletes according to the chromosome "age".

*Termination condition*

Like the SA metaheuristic, GAs do not stop as soon as they find a local optimum. Theoretically these algorithms can run for ever. In this way, a termination condition is needed in practice. Common termination conditions are to set a limit on the number of generations or on the computer clock time. Another criteria, related to the first one, is to track the population fitness and to stop, for example, if, in a sufficiently large number of generations, this fitness has not changed, since it can be assumed that the solution is close to optimal [30].

Finally, to conclude about GAs, it is necessary to say that they often have to deal with inadmissible individuals. For example, admissible individuals can be recombined and the new generated individual(s) might be inadmissible. There are basically three different ways to handle with inadmissible individuals. The first one is to reject them, the second one is to penalize them in the fitness function and the third possibility consists of trying to repair them (see, e.g., [25]).

## 2.2 Hybrid Metaheuristics

Along the last years it became evident that the concentration on a single metaheuristic is very restrictive for advancing when tackling both academic and practical optimization problems. Many examples have showed that different *hybridid metaheuristics* have provided powerful search algorithms and successful applications, see e.g. [25, 29, 123]. Although there is not a precise definition of the term hybrid metaheuristic, in this work it was adopted the definition in the broad sense of integration of a metaheuristic related concept with other techniques (possibly other metaheuristics). According to Blum and Roli [25] three different types of hybrid metaheuristics can be distinguished. In the first one there is exchange of information among several optimization techniques, the second one consists of integrating metaheuristic with exact methods and in the third one components from one metaheuristic are included into another metaheuristic. The first type of hybridization is related to cooperative and parallel search, which typically consists of parallel implementations of metaheuristics, the second type is related to the combination of metaheuristics with techniques which are typical of other fields, such as operations research and artificial intelligence. Since the main goal of this chapter is to provide an overview of the core ideas and strategies of metaheuristics, the third type of hybridization will be briefly described. Readers interested in this and in the other types of hybridizations are referred to, e.g., [25, 29, 87, 123].

Concerning the third type of hybridization, a well-known way of hybridization is the use of trajectory methods into population-based techniques. Indeed, the most successful applications of population-based methods make use of local search procedures. The reason for that becomes clear when the strong points of population-based methods and the trajectory methods are analyzed. Remember that there are two main, complementary, forces (concepts) that determine the behaviour of a metaheuristic, intensification (exploitation) and diversification (exploration). Diversification ensures that many and different regions of the search space are "visited", whereas intensification guarantees a carefully and intensively search within those regions, allowing to obtain high quality solutions. To guarantee an efficiently exploration of a search space there must be an appropriate balance between these two concepts. While all metaheuristics are driven by these two forces, some of them have a clear tendency to intensification and others to diversification.

A natural way to ensure the exploration of the search space is to start the search from multiple solutions. Since population-based metaheuristics deal with a set of solutions rather than with a single solution, they provide an intrinsic and natural way for exploring the search space. Another way to provide diversification is to generate new individuals in "unvisited" regions, which is also done by this type of metaheuristics by means of the application of certain operators to the population. For instance, recombining solutions to obtain new ones enable guided steps in the search space. These steps are generally larger than the ones made by trajectory methods, because a solution resulting from a recombination usually differs more from the parents than a solution resulting from a predecessor solution, obtained by applying a move in a trajectory method. Although trajectory methods can also perform large steps, unlike the previous ones, these steps are not usually guided. It can be said that they are "blind". From this brief overview it can be said that the main driven force of population-based methods is diversification, which allows to find promising areas in the search space. Concerning the trajectory methods, their strength is found in the way they exploit a promising region in the search space. With these methods a promising area is searched in a more structured way than with population-based strategies, because their driving component is the local search. Thus, the danger of being close to good solutions and "lose" them is lower than in the population-based methods.

In summary, population-based methods are better in identifying promising areas on vast and complex search spaces, whereas trajectory methods are better in exploiting those promising areas. In other words, population-based metaheuristics are mainly guided by diversification, while intensification guides trajectory methods. Thus, the idea of combining these two complementary forces, diversification and intensification, is a good reason to hybridize population-based and trajectory metaheuristics and seek to incorporate the strengths and eliminate weaknesses of both types of methods [89].

There are many ways to use trajectory methods in population based techniques. For example, when GAs are used to solve an optimization problem, instead of using blind genetic operators acting regardless the fitness of the original individual, it can be used a trajectory method that considers an individual as its initial solution and then replaces the original individual by the improved one. This trajectory method can be, for instance, "placed" between two standard genetic operators or can replace a standard genetic operator (see Figure 2.3).

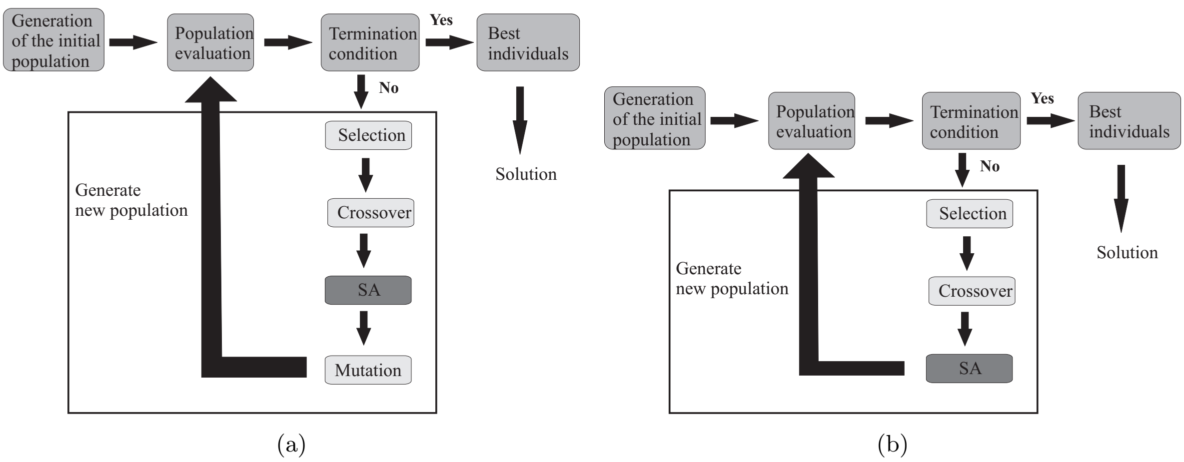



Figure 2.3: (a) SA is an additional genetic operator; (b) SA replaces the mutation operator.

Another popular way to use trajectory methods in population based techniques is to apply these methods one after another, each one using the output of the previous one as its own input, acting in a pipeline fashion. For example, the typical evolution of a GA is that, after a certain amount of time, the population is quite uniform, the fitness of the population is no longer decreasing and the chances to produce better individuals is very low. In other words, the search process has been trapped in a basin of attraction from which the probability to escape is very low. It is interesting to exploit this basin of attraction in order to find its optimal point. As stated before, exploitation is the main strength of trajectory methods, so it is more efficient to use these methods, for example, iterative local search or SA, to perform this exploitation (see Figure 2.4 (a)). It is also possible to use a trajectory method, such as SA, to generate a "good" initial population for GAs (see Figure 2.4 (b)). Another possibility is to use SA to improve a population obtained by a GA (see Figure 2.4 (c)).

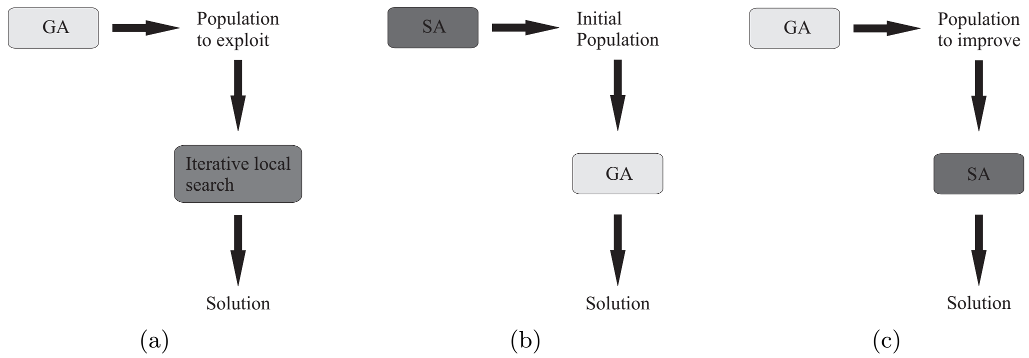



Figure 2.4: Three different ways to use trajectory methods in population based techniques in a pipeline fashion.

Here only some examples of combinations of population-based and trajectory methods were presented. The interested readers on further combinations are referred to 0sman and Laporte review [105], which contains more than 50 references, or to [38] that not only present hybridizations of SA and GAs metaheuristics, but also have various references to other works that deal with the hybridization of SA and GAs metaheuristics.

Note that, as examples for the hybridizations of population-based and trajectory methods, we considered the the GAs and SA metaheuristics because these are the ones used in this work and whose parameters were described in section 2.1.

# Chapter 3

# Maximum Hidden Vertex Set Problem

This chapter focuses on hidden problems. These problems deal with finding the maximum number of points on a given geometric configuration, such that no two of these points see each other. In this class of visibility problems, if the input is a polygon $P$, there are two problems: the Maximum Hidden Set problem and the Maximum Hidden Vertex Set problem [50]. The Maximum Hidden Set problem asks for a set $S$ of maximum cardinality of points in the interior or on the boundary of a given polygon, such that no two points of $S$ see each other. The Maximum Hidden Vertex Set problem asks for a set $S$ of maximum cardinality of vertices of a given polygon, such that no two vertices of $S$ see each other. Shermer [117] proved that both problems are $\mathcal{NP}$-hard both on arbitrary and orthogonal polygons. In this chapter, the Maximum Hidden Vertex Set problem, MHVS($P$), is studied.

The chapter is divided in five sections. In the first one, section 3.1, the problem is described and formalized. In section 3.2 four approximation algorithms to determine approximate solutions to the problem are presented. The first two are greedy strategies designed specifically to solve the MHVS($P$) problem (subsection 3.2.1) and the other two algorithms are based on the simulated annealing and genetic algorithms metaheuristics (subsections 3.2.2 and 3.2.3, respectively). Since the optimal solution for the MHVS($P$) problem is unknown, in section 3.3 it is developed a method to determine an upper bound for it. This method allows to get the performance ratio of the developed approximation algorithms. In section 3.4 are described the computational experiments made over a large set of randomly generated polygons (arbitrary and orthogonal). In section 3.5 some conclusions are presented.

Let, finally, mention that some of the results appearing in this chapter have been published in [17].

## 3.1 Problem Description

**Definition 3.1** *Let $P$ be a polygon. A* ***hidden set*** *for $P$ is set of points on $P$ such that no two points of the set are visible to each other. That is, a hidden set for $P$ is a set $S$ such that*

$$S \subset P \text{ and } \forall p, q \in S, p \neq q \Rightarrow [pq] \cap P \neq [pq].$$

*An element of a hidden set is designated by* ***hidden point****.*

**Definition 3.2** *Let $P$ be a polygon. A* ***hidden vertex set*** *for $P$ is a hidden set $S$ for $P$ such that each element of $S$ is a vertex of $P$. A hidden vertex set for $P$ is denoted by $H$ and an element of $H$ is designated by* ***hidden vertex****.*

Shermer [117] proved that on arbitrary and orthogonal polygons, with $n$ vertices, the size of a hidden vertex set of maximum cardinality can be as large as, but not exceed, $\lceil \frac{n}{2} \rceil$ and $\frac{n-2}{2}$, respectively. These tight bounds can be achieved on *triangular saw* polygons and on *staircase* polygons, respectively (see Figure 3.1).


Figure 3.1: (a) *Triangular saw* polygons; (b) *Staircase* polygons.

But although it is possible to hide the above established number of vertices on some polygons, for many others these numbers are clearly too large. This reasoning justifies the algorithmic MAXIMUM HIDDEN VERTEX SET problem, which formally is denoted by MHVS($P$) and can be stated as follows:

| |
|---|
| **MHVS($P$)** |
| **Input**: A polygon $P$ with $n$ vertices. |
| **Question**: What is the maximum number of hidden vertices on $P$? |

Shermer proved that this problem is $\mathcal{NP}$-hard both on arbitrary and orthogonal polygons [117]. Thus, in this work, approximation methods were developed to tackle it. These methods are described in the next section.

## 3.2 Approximation Methods

Four approximation algorithms were designed to determine a hidden vertex set $H$, whose cardinality approximates the maximal number of hidden vertices on given polygon $P$. The first two, designated by $M_1$ and $M_2$, are greedy algorithms. The other two, called $M_3$ and $M_4$, are based on the general metaheuristics Simulated Annealing (SA) and Genetic Algorithms (GAs), respectively.

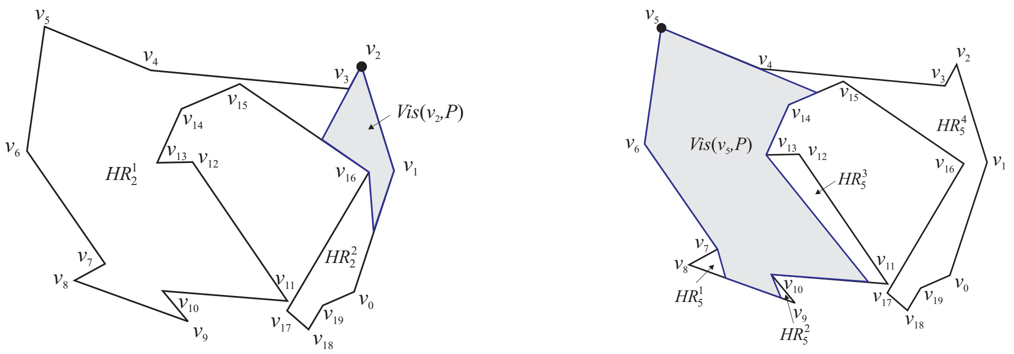
### 3.2.1 Greedy Strategies

A natural approach to find a hidden vertex set $H$ is to do so in a greedy way, that is: add hidden vertices one by one until $H$ (which initially is empty) is achieved, selecting at each step a hidden vertex from the set of vertices of $P$, according to some rule. Two different rules were applied to select the hidden vertices. Therefore, there are two different greedy algorithms: $M_1$ and $M_2$. Below are defined some concepts which will be necessary to explain these algorithms.

**Definition 3.3** *Being $v_i$ a vertex of a polygon $P$, a sub-polygon (region) of $P$ whose points are not seen by $v_i$ is called* ***hidden region*** *of $v_i$. The set of all hidden regions of $v_i$ is denoted by $HR_i$. That is,*

$$HR_i = P \setminus Vis(v_i, P) = \{HR_i^1, \ldots, HR_i^k\},$$

*where $HR_i^j$, with $j \in \{1, \ldots, k\}$ is a hidden region of $v_i$ (see Figure 3.2).*




(a) (b)

Figure 3.2: A 20-vertex polygon and (a) $Vis(v_2, P)$ and $HR_2 = \{HR_2^1, HR_2^2\}$; (b) $Vis(v_5, P)$ and $HR_5 = \{HR_5^1, HR_5^2, HR_5^3, HR_5^4\}$.

A *graph* $G = (V, E)$ consists of a finite, nonempty, set $V$ and a set $E$ whose elements are subsets of $V$ of size 2, that is, $\{u, v\}$, where $u, v \in V$. The elements of $V$ are called *vertices* or *nodes* and the elements of $E$ are called *edges*. An edge $\{u, v\}$ can also be denoted by $uv$. Two nodes $u$ and $v$ are called adjacent if $uv \in E$.

**Definition 3.4** *The **visibility graph** of a polygon $P$ is defined as:* $VG(P) = (V, E)$*, where* $V = \{v \mid v$ *is a vertex of* $P\}$ *and* $E = \{uv \mid$ *the vertices* $u$ *and* $v$ *are visible on* $P\}$ [119]. *Figure 3.3 illustrates a polygon and its visibility graph.*

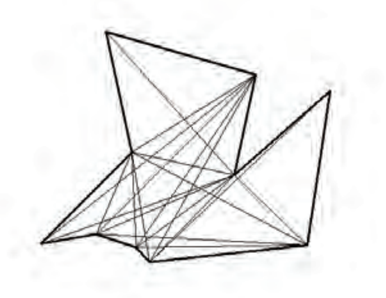


Figure 3.3: A 10-vertex polygon and its visibility graph.

Now, the description of the two methods, $M_1$ and $M_2$, will follow.

**Method $M_1$.** The rule that gives rise to method $M_1$ is based on the hidden region concept (definition 3.3). Based on some experiments, it was observed that, in most cases, the convex vertices are those that have more hidden regions. Therefore, convex vertices are selected one by one, according to the cardinality of its hidden region set and the area of its visibility polygon. In this way, the set $H$ is built. See Algorithm 3.1 for a complete description of method $M_1$.

**Algorithm 3.1** Determining $H$ from the hidden regions (method $M_1$)

**Input:** A polygon $P$ with $n$ vertices and $VG(P)$

**Output:** A hidden vertex set, $H$

$H \leftarrow \emptyset$
$V_{conv} \leftarrow \{v_i \in V_P \mid v_i$ is convex$\}$
**for** each $v_i \in V_{conv}$ **do**
  determine $Vis(v_i, P)$ and $|HR_i|$
**end for**
**while** $V_{conv} \neq \emptyset$ **do**
  Choose $v_i$ from $V_{conv}$ with more hidden regions; and, in the event of a tie, choose the one whose the area of $Vis(v_i)$ is smaller
  $H \leftarrow H \cup \{v_i\}$
  Delete $v_i$ (and all vertices seen by $v_i$) from $V_{conv}$
**end while**
**return** $H$

In the algorithm described above, the calculating of $VG(P)$ is done as a pre-processing step and is made according to the algorithm developed by Hershberger [68] of time complexity $\mathcal{O}(n^2)$. The calculating of $Vis(v_i, P)$ is made according to the linear algorithm of Lee [85].

**Method $M_2$.** The second rule is based on the number of vertices seen by each vertex. Thus, the method $M_2$ is similar to $M_1$. The main differences are: in step 2, where it is considered the set of all vertices of $P$, $V_P$, instead of the set of convex vertices; and in step 7, where instead of selecting the convex vertex that has more hidden regions, is chosen the vertex that sees less vertices.

### 3.2.2 Simulated Annealing Strategy

As stated in Chapter 2, subsection 2.1.1, to solve an optimization problem with the SA metaheuristic it is necessary to identify the following parameters: 1. **Specific Parameters** (of the problem): *solution space*, *objective function*, *neighbourhood of each solution* and *initial solution*; 2. **Generic Parameters** (of the annealing strategy): *initial temperature* ($T_0$), *temperature decrement rule*, *number of iterations at each temperature* ($N(T_k)$) and *termination condition*. Below, are described how these parameters were defined to suit the MHVS($P$) problem.

#### 1. Specific Parameters

**Solution space.** The solution space, set $S$, to the MHVS($P$) problem is the set of all hidden vertex sets for $P$. Therefore, $S$ is finite and can be represented by $S = \{S_1, S_2, \ldots, S_m\}$, where $S_i = v_0^i v_1^i \ldots v_{n-1}^i$ for $i = 1, \ldots, m$. In this way, each element of $S$ (i.e., each candidate solution), $S_i$, is represented by a chain with length $n$, where $v_j^i$, with $j \in \{0, \ldots, n-1\}$, represents the vertex $v_j \in V_P$ and its value is 0 or 1. If $v_j^i = 1$ then the vertex $v_j$ is marked as a hidden vertex; otherwise ($v_j^i = 0$) the vertex $v_j$ is marked as not hidden. In order to illustrate these notions an example is presented in Figure 3.4.

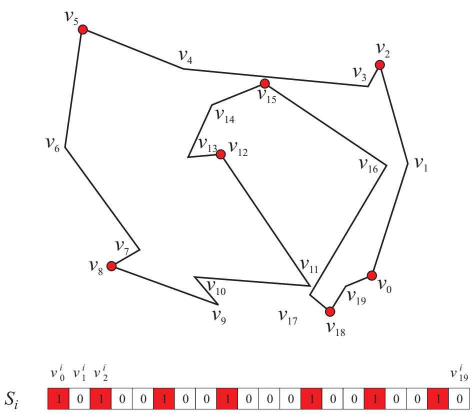



Figure 3.4: An element $S_i \in S$ (for a 20-vertex polygon) and its representation. Red dots represent hidden vertices.

**Objective function.** The objective function $f : S \to \mathbb{N}$ assigns to each element of $S$ a natural value. For each $S_i \in S$, $f$ is defined by $f(S_i) = \sum_{j=0}^{n-1} v_j^i$. In this way, $f(S_i)$ is equal to the cardinality of the hidden vertex set represented by the chain $S_i$.

**Neighbourhood of each solution.** According to SA, for each candidate solution $S_i \in S$, an element $S_j \in S$, called neighbour of $S_i$, must be obtained in order to be analyzed in the next iteration. In our case, given $S_i = v_0^i, \ldots, v_{n-1}^i$, a natural number $t \in [0, n-1]$ is randomly generated (following a uniformly distribution) and then if:

- $v_t^i = 1$ then $v_t^j$ is set to 0 (that is, $v_t^j = 0$) and this new solution is accepted with a probability because it is a worse solution.
- $v_t^i = 0$ then $v_t^j$ is set to 1 (that is, $v_t^j = 1$). If this new solution is an admissible (or *valid*) solution, then it is accepted since the solution was improved; else, in the developed algorithm, the obtained solution is repaired (or *validated*) and it is accepted with a probability.

The *validation process* is done in the following way: all hidden vertices are marked as not hidden if $v_t$ sees them, in other words, if $v_k^j = 1$ and $v_t$ sees $v_k$ then the value of $v_k^j$ is changed to 0. Note that, this validation process always worsen the solution that is why the new solution is accepted with a probability. For example, suppose that we want to obtain a neighbour of the solution $S_i$ illustrated in Figure 3.4. Suppose, also, that the randomly generate number is $t = 16$. The resulting neighbour $S_j$ is invalid because the vertex $v_{16}$ is seen by the vertices $v_0, v_2$ and $v_{15}$. Thus, in the validation process the vertices $v_0, v_2$ and $v_{15}$ are marked as not hidden (see Figure 3.5, for illustration).

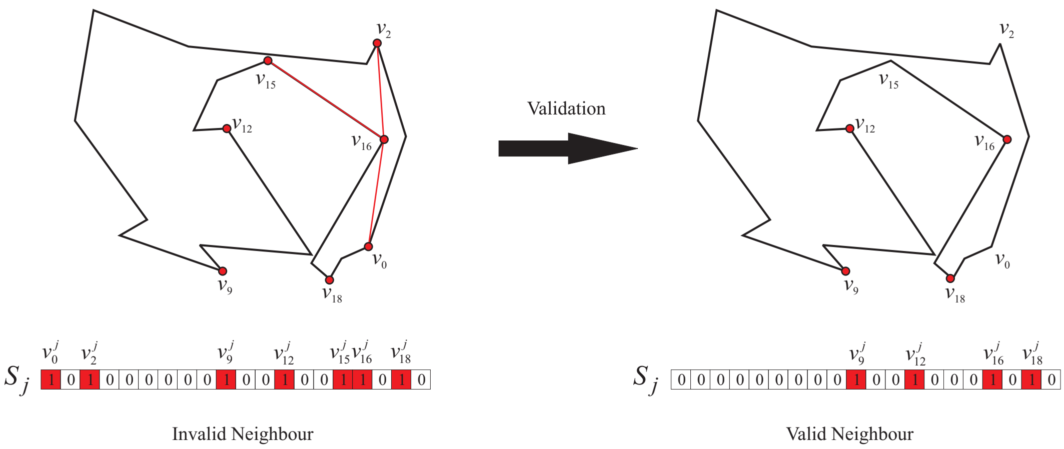



Figure 3.5: Solution Validation. Red dots represent hidden vertices.

**Initial solution.** The initial solution that the SA strategy needs to tackle the MHVS($P$) problem is an element of $S$, which is designated by $S_0$, and is the first solution to be analyzed.

In the developed algorithm, it was considered $S_0 = 10\ldots0$, that is, the vertex $v_0$ is marked as hidden and the remainder are labelled as not hidden.

**2. Generic Parameters**

**Initial temperature ($T_0$).** Following the idea described in Chapter 2, subsection 2.1.1, a comparative study has been performed taking into account two different types of $T_0$:

1. An initial temperature that depends on the number of vertices of the polygon $P$, i.e., $T_0 = f(n)$. In the conducted study it was considered $T_0 = n$;
2. A constant initial temperature, i.e., $T_0 = c$, with $c \in \mathbb{R}^+$. In the performed study it was chosen $T_0 = 1000$.

**Temperature decrement rule.** Remember that the value of the temperature at each iteration $k$, $T_k$, is established by the temperature decrement rule. An analysis was made on three different types of rules (see Chapter 2, subsection 2.1.1):

1. $T_{k+1} = \frac{T_0}{1+k}$ (Fast Simulated Annealing (FSA) decrease);
2. $T_{k+1} = \frac{T_0}{e^k}$ (Very Fast Simulated Annealing (VFSA) decrease) and
3. $T_{k+1} = \alpha T_k$, where $0 < \alpha < 1$ (Geometric decrease). In the performed study it was chosen $\alpha = 0.9$.

**Number of iterations at each temperature ($N(T_k)$).** In the developed algorithm it was considered $N(T_k) = \lceil T_k \rceil$, this ensures that there are more iterations while the temperatures are high, when the solutions are still far from optimal.

**Termination condition.** The chosen termination condition consists of finishing the search when the temperature is less than or equal to 0.005, i.e, $T_f = 0.005$. Clearly for lower temperatures the obtained solution will be closer to the optimum, but the response time of the algorithm will increase considerably.

### 3.2.3 Genetic Algorithms Strategy

As stated in Chapter 2, subsection 2.1.2, to solve an optimization problem with the GA metaheuristic it is necessary to specify the following parameters: *encoding*, *initial population*, *fitness function*, *genetic operators* (*selection*, *crossover* and *mutation*) and *the value of various parameters used by the genetic algorithm* (e.g., population size, probability of the genetic operators, population evaluation, population generation, termination condition). Next, it is explained how these parameters were delineated to suit the MHVS($P$) problem.

**Encoding.** In the developed algorithm an individual $I$ is represented by a chain of 0's and 1's, with length $n$, i.e., $I = g_0g_1 \ldots g_{n-1}$. Each gene represents a vertex of the polygon, that is, the $i^{th}$ gene $g_i$ represents the vertex $v_i \in V_P$, and its value is 0 or 1. If $g_i = 1$ then the vertex $v_i$ is marked as a hidden vertex; otherwise ($g_i = 0$) the vertex $v_i$ is marked as not hidden (see Figure 3.6). Note that, the genetic representation of an individual $I$ is similar to the representation of a candidate solution $S_i$ in the SA strategy.

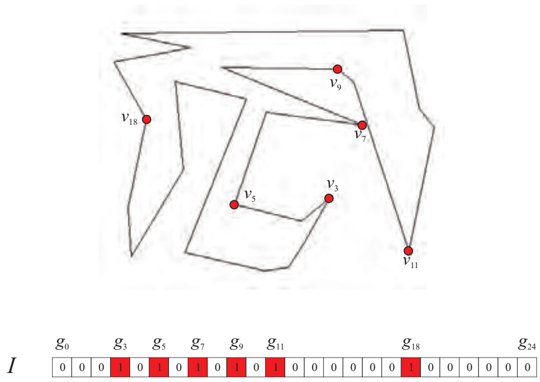



Figure 3.6: An individual $I$ (for a 25-vertex polygon) and its representation. Red dots represent hidden vertices.

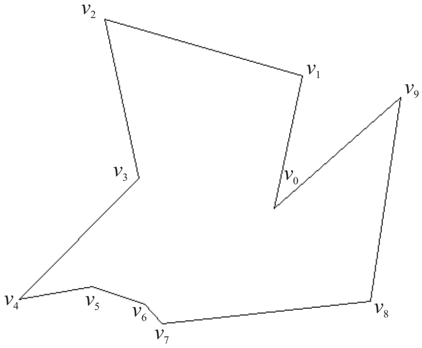
**Initial population.** It has been taken as the population size the number of vertices of the polygon, linking in this way the problem input with the metaheuristic components. Thus, the population for the generation $t$ is represented by: $P(t) = \{I_0^t, I_1^t, \ldots, I_{n-1}^t\}$, where each $I_i^t$ represents an individual belonging to the population $P(t)$ and $n$ is the number of vertices of the polygon $P$.

Remember that an individual represents a candidate solution to the MHVS($P$) problem, i.e., each individual must be a hidden vertex set. Thus, to create the initial population, $P(0)$, each of the $n$ individuals is generated in the following way: $\forall i \in \{0, \ldots, n-1\}$, the vertex $v_i$ and all the vertices on $P$ that form with $v_i$ a hidden vertex set are marked as hidden. See Algorithm 3.2, for illustration.

**Algorithm 3.2** Generation of $I_i^0$, $i \in \{0, \ldots, n-1\}$

1. $g_i \leftarrow 1$ and $g_j \leftarrow 0, \forall j \neq i$ // $H = \{v_i\}$
2. **for** $j = 0$ to $n-1$ **do**
3. **if** $v_j \cup H$ is a hidden vertex set **then**
4. $g_j \leftarrow 1$ // $H = H \cup \{v_j\}$
5. **end if**
6. **end for**

For example, in Figure 3.7 it is illustrated a polygon with 10 vertices and its initial population, $P(0) = \{I_0^0, I_1^0, \ldots, I_9^0\}$.




| | |
|---|---|
| $I_0^0 = 1000000000$ | $I_5^0 = 0010010100$ |
| $I_1^0 = 0100000010$ | $I_6^0 = 0010101001$ |
| $I_2^0 = 0010101001$ | $I_7^0 = 0010100100$ |
| $I_3^0 = 0001000001$ | $I_8^0 = 0100000010$ |
| $I_4^0 = 0010101001$ | $I_9^0 = 0100000001$ |

Individuals of *Pop*(0)

Figure 3.7: Polygon with $n = 10$ and its initial population.

**Fitness function.** Remember that the fitness function should help to make the selection of the best individuals to be reproduced, so that it should assign higher values to the solutions closer to the optimal one(s). This function was defined in a similar way to the objective function defined for the SA strategy. For each $I$, $f$ is defined by $f(I) = \sum_{i=0}^{n-1} g_i$, that is, $f(I)$ represents the cardinality of the hidden vertex set represented by $I$.

**Selection.** The selection method should choose the best individuals to be reproduced. Although there are various different types (see Chapter 2, subsection 2.1.2) of selection, in the developed algorithm it was used the roulette wheel selection to choose the two best individuals to be parents in crossover.

**Crossover.** Crossover operates on selected genes from parent individuals and creates new individuals (children). In the developed algorithm was used the single point crossover (see Chapter 2, subsection 2.1.2). In this type of crossover two children are often created and inserted into the new population, but sometimes only one child is created. It was used the single point crossover to generate only *one* child (see Figure 3.8, for illustration).

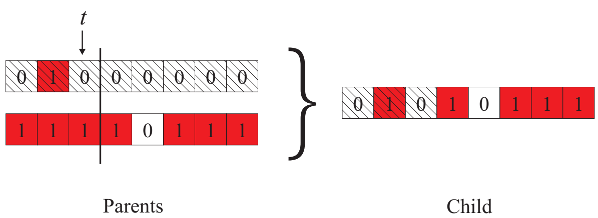



Figure 3.8: Single point crossover.

Remember that crossover does not always occur, it occurs with a certain probability, $p_c$. It was used $p_c = 0.8$ (this value was experimentally chosen). Note that the child resulting from this type of crossover may not be valid, that is, it may not correspond to a hidden vertex set. In the developed algorithm it was chosen to repair or validate it. For that, the best tail, which is the tail that has more 1's, is fixed and the other tail is changed as follows. Suppose that the selected gene is $g_t$, $t \in \{0, \ldots, n-1\}$, and that the worst tail has $m$ genes. Two cases may take place: ($i$) the worst tail is the first one ($ii$) the worst tail is the second one. In the case:

($i$) for each $i \in \{0, \ldots, m-1\}$, if $g_i = 1$ and $v_i$ is not seen by any vertex represented in the second tail, then the value 1 is maintained, otherwise its value is changed to 0.

($ii$) for each $i \in \{0, \ldots, m-1\}$, if $g_{t+i+1} = 1$ and $v_{t+i+1}$ is not seen by any vertex represented in the first tail, then the value 1 is maintained, otherwise its value is changed to 0.

Figure 3.9 exemplifies case (2). In this example, $t = 8$ and we can see that the generated child is not valid, since $v_9$ and $v_{11}$ are seen by $v_8$. The worst tail is the second one and it has $m = 11$ elements. Since $g_9 = 1$ and $v_9$ is seen by the vertex $v_8$, its value is changed to 0; $g_{11} = 1$ and $v_{11}$ is seen by the vertex $v_8$, so its value is changed to 0; and $g_{18} = 1$ and $v_{18}$ is not seen by any vertex represented in the first tail, so its value remains 1. Thus, the new child is now 10100100100000000010, which is a valid individual.

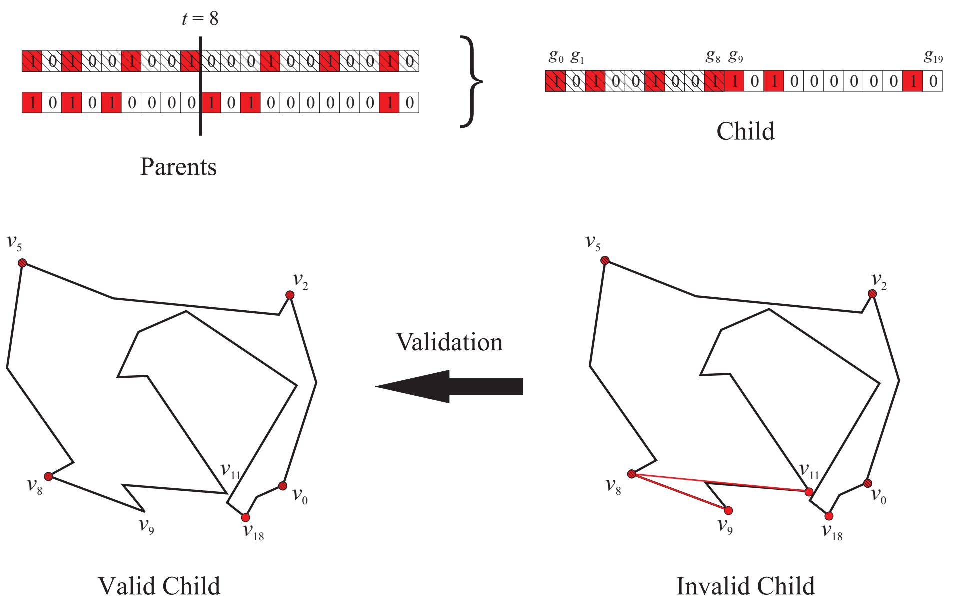



Figure 3.9: Single point crossover and child validation.

**Mutation.** In the developed algorithm the mutation operator merely flips a randomly selected binary digit from zero to one or vice versa, as shown in Figure 3.10. Such as the crossover, the mutation does not always occur, but it occurs with a certain probability, $p_m$. In the developed algorithm the mutation is applied to the child obtained on the crossover operation, with $p_m = 0.05$ (this value was experimentally chosen) as follows: a natural number $t \in [0, n-1]$ is randomly generated (following a uniformly distribution). If $g_t = 1$ then its value is changed to 0; otherwise ($g_t = 0$) its value is changed to 1 only if the resultant individual is valid (i.e., if the resultant individual represents a hidden vertex set).

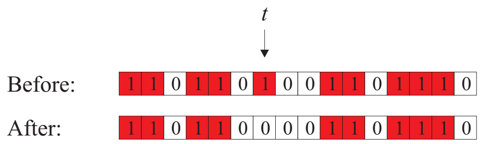



Figure 3.10: Mutation.

**Population generation.** To generate a new population it was used a steady-state reproduction (see Chapter 2, subsection 2.1.2), where the worst individual of the population is replaced by the child obtained on the crossover operation.

**Population evaluation.** The fitness of a population was considered as the maximum value of the fitness function when applied to all individuals of the population, i.e., $F(P(t)) = \max\{f(I_0^t), \ldots, f(I_{n-1}^t)\}$.

**Termination condition.** If in a sufficiently large number of generations the fitness has not changed, it can be assumed that the solution is close to optimal (see Chapter 2, subsection 2.1.2). Thus, for the termination condition it was considered that if the fitness of the population $F(P(t))$ remains unchanged for a number of generations $h$, the search should stop. To define this parameter, several tests were made varying the value of $h$. It was observed that from $h = 5000$ the quality of the solution does not improve much. So, it was chosen $h = 5000$ in the developed algorithm.

In the sequel, the GA strategy is, sometimes, designated by $M_4$.

## 3.3 Greedy-Sequential Strategy for the Minimum Clique Partition Problem

Since the MHVS($P$) problem is $\mathcal{NP}$-hard, its optimal solution is unknown. So, if one can not compute the optimal value, how can one expect to prove that the output of the approximation algorithms are near it? As Amit, Mitchell and Packer [11], in this work it was conducted an experimental analysis of their performance. For that, it was developed a method to compute an upper bound on the optimal number of hidden vertices for each instance in our experiments.

Remember that the visibility graph $VG(P)$ of a polygon $P$ is the graph of the visibility relation of the vertices of $P$ (see Definition 3.4). A clique partition of $VG(P)$ is defined as follows.

**Definition 3.5** *A set $C$ is a **clique partition** of $VG(P) = (V, E)$ if its elements are disjoint subsets $V_i$ of $V$, where each vertex on $V_i$ sees all vertices on $V_i$ and $\bigcup V_i = V$. The elements of $C$ are called cliques. Figure 3.11 illustrates a polygon $P$ and a clique partition of $VG(P)$.*

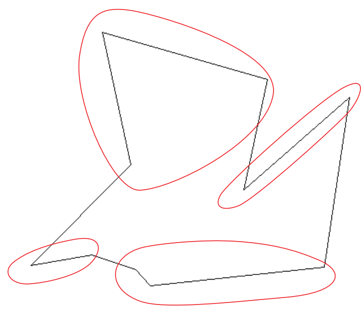


Figure 3.11: A clique partition (with four cliques) of $VG(P)$.

It is easy to see that, for each element of $C$ at most one vertex can be hidden on $P$, so $|C| \geq |H|$, $\forall C, H$. Easily it can be concluded that $c(P) \geq h(P)$, where $h(P)$ is the number of hidden vertices in a maximum-cardinality hidden vertex set of $P$ and $c(P)$ is the number of cliques in a minimum-cardinality clique partition of $VG(P)$. Thus, $c(P)$ is an upper bound on the optimal number of hidden vertices on $P$.

However, the problem of determining this upper bound, MINIMUM CLIQUE PARTITION (MCP) problem, is also $\mathcal{NP}$-hard [52]. So, it was developed an algorithm to determine approximate solutions (which will be described below).

Let $|C|$ be the cardinality of an approximate solution of the MCP problem

$$|H| \leq h(P) \leq c(P) \leq |C|, \forall P. \tag{3.1}$$

If there is a constant $c \in \mathbb{R}^+$ such that $|H| \geq \frac{1}{c} \times |C|$, for any polygon $P$, it can be said that the approximation algorithm used to obtain $H$ has an approximation ratio of $c$ [13]. In other words, the approximate solution $|H|$ is at least $\frac{1}{c}$ times the optimal solution $h(P)$. In fact,

$$|H| \geq \frac{1}{c} \times |C| \Rightarrow |H| \geq \frac{1}{c} \times h(P). \tag{3.2}$$

The developed approximation algorithm is a greedy-sequential constructive strategy, which is designated by $A_1$. In this strategy, first $n$ clique partitions of $VG(P)$ are determined, each one from each of the vertices of $P$ (see Algorithm 3.3). In the end, the partition with fewer elements is the solution returned by the algorithm.

**Algorithm 3.3** Algorithm to determine a clique partition from the vertex $v_k$

1. $C \leftarrow \emptyset$
2. $j = k$
3. **repeat**
4. Determine a clique from $v_j$, $C_j = \{v_j, v_{j+1}, \ldots, v_i\}$
5. $C \leftarrow C \cup C_j$
6. $j \leftarrow (i+1) \bmod n$
7. **until** $j \neq k$

In this way, the application of the algorithms described in section 3.2 (greedy and meta-heuristics based strategies) together with this method $A_1$, to each instance in our experiments, gives provable performance bounds in terms of approximation ratios.

Remark that in the performed experiments, given a polygon $P$, the main objective is to find a large hidden vertex set $H$ and a small clique partition $C$. The obtained set $H$ approximates the optimal number of hidden vertices, $h(P)$, with approximation ratio $\frac{|C|}{|H|}$. Note that if a clique partition set $C$ and a hidden vertex set $H$ are found, such that $|C| = |H|$, then $H$ is an optimal hidden vertex set.

## 3.4 Experiments and Results

To understand which of the described approximation strategies yields the best approximate solutions in a reasonable time, they were implemented and their behavior was tested over a large set of randomly generated polygons. In the next two subsections, subsections 3.4.1 and 3.4.2, it will be reported the results and conclusions from the accomplished experiments on arbitrary and orthogonal polygons, respectively.

### 3.4.1 Arbitrary Polygons

The experiments described in this section were performed over four sets of randomly generated arbitrary polygons, each one with 50 polygons of 50, 100, 150 and 200-vertex polygons.

#### 3.4.1.1 Analysis of the SA Parameters

According to section 3.2.2, there are several choices for two of the SA parameters: the initial temperature ($T_0$) and the temperature decrement rule. The different combinations of their values give rise to six cases (see Table 3.1), which are going to be analyzed to find the combination that best fits into the MHVS($P$) problem.

| SA Cases | |
|---|---|
| **Case 1** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 2** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 3** | $T_0 = n$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |
| **Case 4** | $T_0 = 1000$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 5** | $T_0 = 1000$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 6** | $T_0 = 1000$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |

Table 3.1: Studied cases for SA.

These six cases were analyzed by comparing the number of hidden vertices, the runtime and the number of iterations performed by each of them. Table 3.2 presents the results obtained with the first three cases. In this table it is exhibited the average time of pre-processing in seconds, *PP*, the average number of hidden vertices, $|H|$, the average runtime in seconds, *Time*, and the average number of algorithm iterations, *Iterations*.

| $n$ | PP (sec.) | Case 1 (FSA dec.) | | | Case 2 (VFSA dec.) | | | Case 3 (Geometric dec.) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations |
| 50 | 0.34 | 13.96 | 0.02 | 9999.00 | 6.72 | $< 0.001$ | 10.00 | 9.60 | $< 0.001$ | 88.00 |
| 100 | 1.84 | 27.40 | 0.06 | 19999.00 | 11.68 | $< 0.001$ | 10.00 | 16.56 | $< 0.001$ | 94.00 |
| 150 | 5.32 | 40.50 | 0.18 | 29999.00 | 17.68 | $< 0.001$ | 11.00 | 22.28 | 0.02 | 98.00 |
| 200 | 11.98 | 53.86 | 0.26 | 39999.00 | 22.76 | $< 0.001$ | 11.00 | 28.14 | $< 0.001$ | 101.00 |

Table 3.2: Results obtained with Case 1, Case 2 and Case 3 ($T_0 = n$) on arbitrary polygons.

In these first three cases the best solution, concerning the average number of hidden vertices, seems to be obtained with Case 1. So, the best solution seems to correspond to the FSA temperature reduction (which is the slowest temperature reduction) with a larger number of iterations and a greater response time.

In the following three cases, it is going to be analyzed how the different types of temperature decreasing behave, being $T_0$ constant (see Table 3.3).

| $n$ | PP (sec.) | Case 4 (FSA dec.) | | | Case 5 (VFSA dec.) | | | Case 6 (Geometric dec.) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations |
| 50 | 0.34 | 13.96 | 0.68 | 199999.00 | 6.52 | 0.02 | 13.00 | 9.60 | 0.02 | 116.00 |
| 100 | 1.84 | 27.40 | 0.88 | 199999.00 | 11.92 | $< 0.001$ | 13.00 | 16.66 | 0.02 | 116.00 |
| 150 | 5.32 | 40.48 | 1.08 | 199999.00 | 17.48 | $< 0.001$ | 13.00 | 22.28 | 0.04 | 116.00 |
| 200 | 11.98 | 53.84 | 1.36 | 199999.00 | 22.30 | $< 0.001$ | 13.00 | 28.54 | 0.04 | 116.00 |

Table 3.3: Results obtained with Case 4, Case 5 and Case 6 ($T_0 = 1000$) on arbitrary polygons.

As the number of vertices of the analyzed polygons is 50, 100, 150 and 200, it has been chosen a constant value $T_0 = 1000$ (this value was experimentally chosen). This way, we have a constant value much greater than any $n$ value and considered large enough so that we can see how the algorithm behaves with a high initial temperature. As can be seen, in these three cases the best solution seems to be obtained with Case 4. Thus, the best solution apparently corresponds, again, to the FSA temperature reduction, with a larger number of iterations and a greater response time.

Comparing the six cases we can notice that the obtained average of $|H|$ is almost equal for the Cases 1 and 4, Cases 2 and 5 and Cases 3 and 6. That is, no matter the type of $T_0$, the averages of $|H|$ are identical for FSA, VFSA and geometric temperature reductions. Besides, the Cases 1 and 4 seem to be the strategies that obtain the best solutions, followed by Cases 3 and 6, and finally it looks like Cases 2 and 5 obtain the worst solutions.

However, as stated before, the comparison between the results obtained with the six cases only makes sense if a statistical study is made to ensure its statistically significance. First of all the results concerning to the number of hidden vertices have been studied.

The analysis of the data normality showed that the data obtained with Case 1 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 50, 100, 150$ and 200). So, it was applied the Kruskal-Wallis test, which showed that there was a significant difference between the six cases with respect to $|H|$, for $n = 50, 100, 150$ and 200 ($p$-value $< 0.001 < 0.05$). So, multiple comparison tests were performed to determine which pairs of results were significantly different, and which were not. The multiple comparison tests allowed to sort the six cases, in ascending order on $|H|$, as follows (see Figure 3.12, in this figure Venn diagrams are illustrated where each ellipse represents the cases that are not significantly different):

- for $n = 50, 100, 150$ and 200:
  - Cases 2 and 5, with no significant differences between them;
  - Case 3 and 6, with no significant differences between them;
  - Cases 1 and 4, with no significant differences between them.

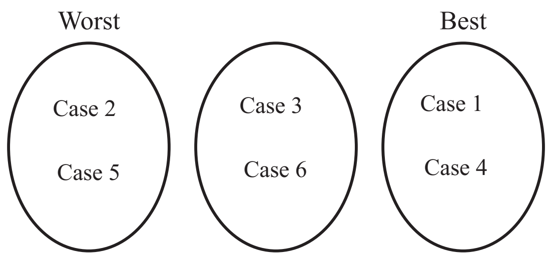



Figure 3.12: Multiple comparison tests, of the six cases, for $n = 50, 100, 150$ and 200 (arbitrary polygons).

As can be noticed, using the multiple comparison tests, a significant difference was not found between the number of hidden vertices obtained by the best cases (Cases 1 and 4). So, the statistical analysis proceeded regarding the runtime.

This analysis was made in a similar way and allowed to conclude that Case 1 is significantly faster than Case 4, for $n = 50, 100, 150$ and 200. Given that it is desired a compromise between the goodness of the solution obtained and the algorithm runtime, Case 1 was chosen to be method $M_3$.

#### 3.4.1.2 Comparison of the four strategies

In this section are going to be analyzed and compared the results obtained with the four developed approximation techniques: the greedy strategies $M_1$ and $M_2$, the SA strategy $M_3$ and the GA strategy $M_4$. Table 3.4 presents the results obtained with these strategies. These results, as in Tables 3.2 and 3.3, are: the average time of pre-processing in seconds, *PP*; the average number of hidden vertices, $|H|$; the average runtime in seconds, *Time*; and the average number of algorithm iterations, *Iterations*.

| $n$ | PP | $M_1$ | | | $M_2$ | | | $M_3$ | | | $M_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. |
| 50 | 0.34 | 12.10 | 0.04 | 12.10 | 13.58 | $< 0.001$ | 13.58 | 13.96 | 0.02 | 9999 | 13.48 | 0.12 | 5135.90 |
| 100 | 1.84 | 24.18 | 0.18 | 24.18 | 27.12 | $< 0.001$ | 27.12 | 27.40 | 0.06 | 19999 | 26.32 | 0.22 | 5724.80 |
| 150 | 5.32 | 35.12 | 0.52 | 35.12 | 39.88 | $< 0.001$ | 39.88 | 40.50 | 0.18 | 29999 | 38.46 | 0.22 | 6629.30 |
| 200 | 11.98 | 46.62 | 0.66 | 46.62 | 52.68 | $< 0.001$ | 56.68 | 53.86 | 0.26 | 39999 | 50.32 | 0.24 | 7585.60 |

Table 3.4: Results obtained with $M_1$, $M_2$, $M_3$ and $M_4$ (arbitrary polygons).

Comparing the results obtained with the greedy strategies, $M_1$ and $M_2$, it can be noticed that $M_2$ is faster and seems to obtain better solutions. Concerning the metaheuristic strategies, $M_3$ and $M_4$, it can be seen that $M_3$ appears to be faster (except for 200-vertex polygons) and it seems to obtain better solutions (except for 50-vertex polygons, where the obtained solutions look to be similar).

So, relatively to greedy strategies the method $M_2$, seems to be the best one. Concerning the metaheuristic strategies, the method $M_3$ appear to be better than $M_4$. Comparing, now, the results obtained with $M_2$ and $M_3$, we can see that although $M_3$ is slower, the average of hidden vertices obtained with it looks to be slightly higher (especially for $n = 150$ and $n = 200$).

As a conclusion, $M_3$ seems to be the best technique, since the obtained average of hidden vertices is the best and $M_2$ is the only method that is faster than it. If only the obtained solutions are contemplated it seems that the best approximation technique is $M_3$, the second best is $M_2$, followed by $M_4$, and finally the worst is $M_1$ (see Figure 3.13).

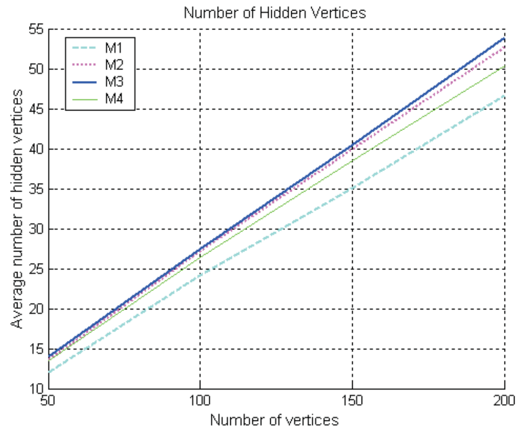
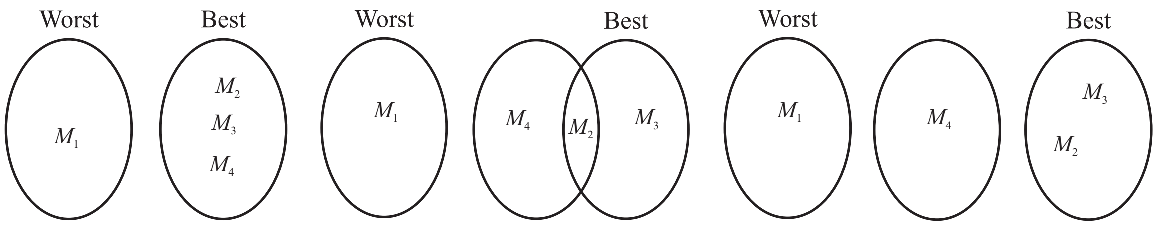



Figure 3.13: Solutions obtained with the strategies $M_1, M_2, M_3$ and $M_4$ (arbitrary polygons).

As in the analysis of the SA Cases, a statistical study was conducted to ensure the statistically significance of the results. In this way, first of all the results related to $|H|$ have been studied. As we already know, the data obtained with $M_3$ is non-normally distributed, for $n = 50, 100, 150$ and 200 (see the previous subsection). Thus, the Kruskal-Wallis test has been used, which showed that there was a significant difference between the four methods concerning $|H|$, for $n = 50, 100, 150$ and 200 ($p$-value $< 0.001 < 0.05$, for $n = 50, 100, 150$ and 200). Therefore, multiple comparison tests were performed, whose results allowed to conclude that (see Figure 3.14, in this figure Venn diagrams are illustrated where each ellipse represents the methods that are not significantly different):

- for $n = 50$, the best method is $M_3$ with no significant differences from methods $M_2$ and $M_4$ and the worst method is $M_1$.
- for $n = 100, 150$ and 200, the best method is $M_3$ with no significant differences from method $M_2$ and the worst method is $M_1$.




(a) $n = 50$ (b) $n = 100$ and $n = 150$ (c) $n = 200$

Figure 3.14: Multiple comparison tests of the four methods (arbitrary polygons).

Thus, it can be concluded that no significant differences were found between methods $M_3$ and $M_2$, although we observed, in Table 3.4, that $M_3$ seems to get slightly better results in the average of the number of hidden vertices. For this reason, a statistical study has been made concerning the runtime. This study was made in a similar way and it allowed to conclude that the method $M_2$ only is significantly faster than $M_3$ for 200-vertex polygons. Once again, and since it is desired a compromise between the goodness of the solution obtained and the algorithm runtime, the method $M_3$ was considered the best method.

Now, to conclude about the average of the maximum number of vertices that can be hidden in a $n$-vertex arbitrary polygon, it was applied the selected method (method $M_3$) to eight sets of arbitrary polygons, each one with 50 polygons of 30, 50, 70, 100, 110, 130, 150 and 200 vertex polygons. The average of the obtained results, concerning $|H|$, is exposed in Table 3.5.

| Vertices | 30 | 50 | 70 | 100 | 110 | 130 | 150 | 200 |
|---|---|---|---|---|---|---|---|---|
| $\|H\|$ | 8.54 | 13.96 | 19.12 | 27.40 | 29.50 | 35.64 | 40.50 | 53.86 |

Table 3.5: Average number of hidden vertices (arbitrary polygons).

Then, using the least squares method, the following linear adjustment was obtained, with a correlation factor of 0.9997 (see Figure 3.15):

$$f(x) = 0.2668x + 0.5494 \approx \frac{x}{3.74} + 0.5494 \approx \frac{x}{3.74}.$$

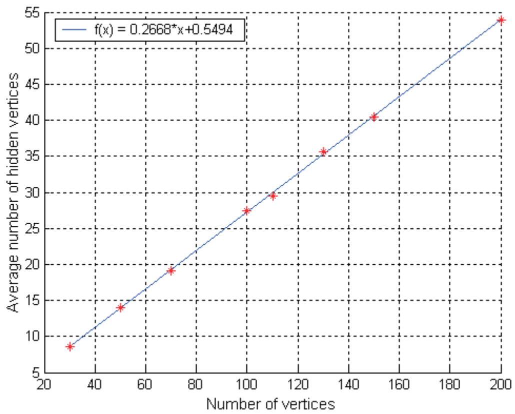
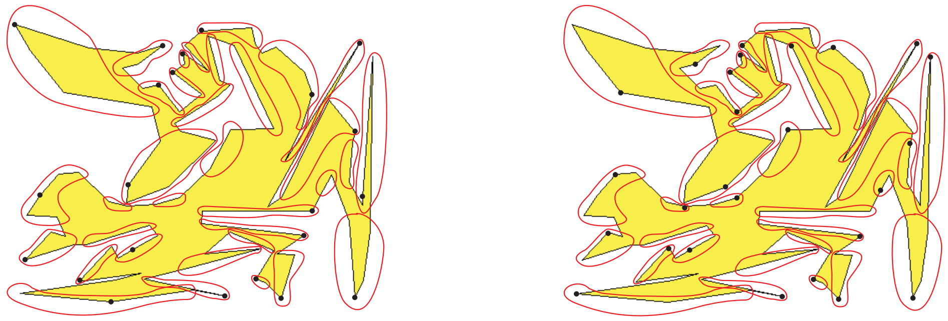



Figure 3.15: Least Squares Method (arbitrary polygons).

Thus, it can be concluded that on average, and approximately, the maximum number of hidden vertices on an arbitrary polygon $P$ with $n$ vertices was observed to be $\lceil \frac{n}{3.74} \rceil$. In order to get a quantitative measure on the quality of the calculated $|H|$, the minimum clique partitions were computed, to our instances (the eight sets of polygons described above). The ratio between the minimum clique partition $C$ and the larger hidden vertex-set $H$ never exceeded 1.62 (with an average of 1.33 for the universe of 320 polygons). This implies that the algorithm $M_3$ has an approximation ratio less than or equal to 1.62.

Figure 3.16 shows a snapshot of one of the 100-vertex arbitrary polygons that has been tested with our software (the black dots represent the obtained hidden vertices).



(a) (b)

Figure 3.16: Example of a tested arbitrary polygon with $n = 100$. Clique partition obtained with algorithm $A_1$ and hidden vertex sets obtained with: (a) method $M_1$ and (b) method $M_3$.

In Figure 3.16(a) it is illustrated the minimum clique partition obtained with $A_1$ (see section 3.3), $|C| = 35$, and the solution obtained with the worst method ($M_1$), $|H| = 23$. In Figure 3.16(b)) it illustrated the minimum clique partition obtained with $A_1$ and the solution obtained with the best strategy ($M_3$), $|H| = 27$. Figure 3.17 shows a 20-vertex saw polygon that has been tested with our software. In this figure, it is illustrated the clique partition obtained with $A_1$, $|C| = 10$, and the solution obtained with $M_3$, $|H| = 10$. Note that in this example, $\frac{|C|}{|H|} = 1$, that is, $M_3$ obtained the optimal solution (the black dots represent the obtained hidden vertices).

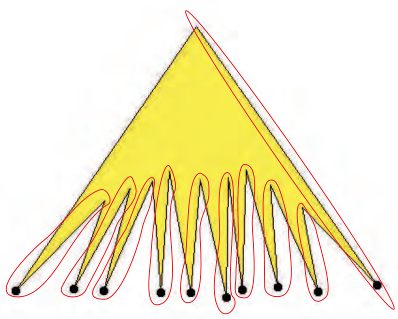


Figure 3.17: The $C$ and $H$ sets in a saw polygon, with $n = 20$, obtained with $A_1$ and $M_3$.

### 3.4.2 Orthogonal Polygons

In this subsection, as in subsection 3.4.1, it will be analyzed the six cases, resulting from the choice of the SA parameters (see Table 3.1), to select the one that best fits on the MHVS($P$)

problem for orthogonal polygons. After that, the results obtained with the four developed strategies will be studied. Like for arbitrary polygons, the experiments for randomly generated orthogonal polygons were performed with four sets of orthogonal polygons, each one with 50 polygons of 50, 100, 150 and 200 vertex polygons.

#### 3.4.2.1 Analysis of the SA Parameters

Table 3.6 shows the results obtained with the first three cases. As we can see, such as on arbitrary polygons, in these three cases the best solutions seem to be obtained in Case 1. So, the best solution appears to correspond to the slowest temperature reduction, with a larger number of iterations and a greater response time. In table 3.7 are presented the results obtained with Cases 4, 5 and 6. As we can observe, in these three cases the best solution seems to correspond, once more, to the slowest temperature reduction, with a larger number of iterations and a greater response time.

| $n$ | PP (sec.) | Case 1 | | | Case 2 | | | Case 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations |
| 50 | 0.62 | 13.52 | 0.08 | 9999.00 | 5.86 | $< 0.001$ | 10.00 | 8.84 | $< 0.001$ | 88.00 |
| 100 | 3.3 | 26.86 | 0.08 | 19999.00 | 10.64 | $< 0.001$ | 10.00 | 15.16 | $< 0.001$ | 94.00 |
| 150 | 8.68 | 39.70 | 0.18 | 29999.00 | 15.08 | $< 0.001$ | 11.00 | 20.12 | 0.02 | 98.00 |
| 200 | 17.80 | 53.16 | 0.22 | 39999.00 | 19.64 | $< 0.001$ | 11.00 | 25.60 | 0.02 | 101.00 |

Table 3.6: Results obtained with Cases 1, Case 2 and Case 3 ($T_0 = n$) on orthogonal polygons.

| $n$ | PP (sec.) | Case 4 | | | Case 5 | | | Case 6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations | $\|H\|$ | Time (sec.) | Iterations |
| 50 | 0.62 | 13.52 | 0.68 | 199999.00 | 6.04 | $< 0.001$ | 13.00 | 9.32 | $< 0.001$ | 116.00 |
| 100 | 3.30 | 26.88 | 1.00 | 199999.00 | 10.36 | $< 0.001$ | 13.00 | 15.22 | $< 0.001$ | 116.00 |
| 150 | 8.68 | 39.74 | 1.08 | 199999.00 | 15.10 | 0.04 | 13.00 | 21.00 | 0.02 | 116.00 |
| 200 | 17.80 | 53.24 | 1.30 | 199999.00 | 19.34 | 0.03 | 13.00 | 26.28 | 0.02 | 116.00 |

Table 3.7: Results obtained with Cases 4, Case 5 and Case 6 ($T_0 = 1000$) on orthogonal polygons.

Comparing the six cases, we can see that the solutions are almost equal for Cases 1 and 4 and for Cases 2 and 5 (except for $n = 50$), that is, independently of the type of $T_0$, the results are identical for FSA and VFSA decreases. We can also notice that Case 6 seems to be better than Case 3. As on arbitrary polygons, it appears that, in spite of the algorithm response time being higher, a slow reduction of the temperature improves the solution. As a conclusion, if we are looking for a solution closer to the optimum, it looks like to be more suitable to choose a slow decreasing of the temperature.

Next it is going to be presented the performed statistical study to ensure the statistically significance of the results. First of all it was performed a study concerning $|H|$. For Case 1, the underlying distribution of the number of hidden vertices was found not to be normally distributed (the $p$-value returned by the Kolmogorov-Smirnof test was less than 0.001, for $n = 50, 100, 150$ and 200). So, the Kruskal-Wallis test has been used, whose results allowed to conclude that there was a significant difference between the six cases concerning $|H|$ ($p$-value $< 0.001 < 0.05$, for $n = 50, 100, 150$ and 200).

Then, multiple comparison tests were performed, whose results showed that the cases arranged in ascending order, on $|H|$, are (see Figure 3.18, in this figure are illustrated Venn diagrams where each ellipse represents the cases that are not significantly different):

- for $n = 50, 100, 150$ and 200:
  - Cases 2 and 5, with no significant differences between them;
  - Case 3 and 6, with no significant differences between them;
  - Cases 1 and 4, with no significant differences between them.

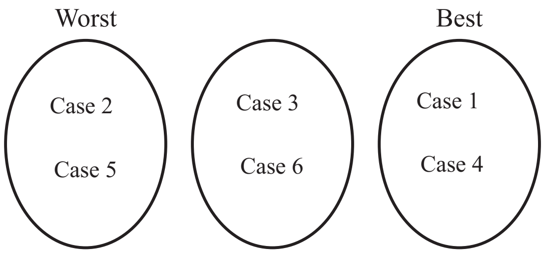



Figure 3.18: Multiple comparison tests, of the six cases, for $n = 50, 100, 150$ and 200 (orthogonal polygons).

From the multiple comparison tests, it can be concluded that Cases 3 and 6 do not obtain significantly different solutions, in spite of our observing that Case 6 gets better results in the average of the number of hidden vertices (see Tables 3.6 and 3.7). It can, also, be concluded that the best cases (Cases 1 and 4), concerning $|H|$, do not obtain significantly differences. So, the statistical study proceeds regarding the runtime. This study was made in a similar way and allowed to conclude that Case 1 is always significantly faster than Case 4 for $n = 50, 100, 150$ and 200. For this reason Case 1 was chosen to be the method $M_3$.

#### 3.4.2.2 Comparison of the four strategies

Table 3.8 presents the results obtained with the four approximation methods on orthogonal polygons.

| $n$ | PP | $M_1$ | | | $M_2$ | | | $M_3$ | | | $M_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. | $\|H\|$ | Time | Iter. |
| 50 | 0.62 | 12.46 | 0.06 | 12.46 | 12.88 | $< 0.001$ | 12.88 | 13.52 | 0.08 | 9999 | 13.12 | 0.06 | 5288.00 |
| 100 | 3.30 | 25.10 | 0.20 | 25.10 | 25.40 | $< 0.001$ | 25.40 | 26.86 | 0.08 | 19999 | 25.32 | 0.08 | 6061.50 |
| 150 | 8.68 | 36.52 | 0.28 | 36.52 | 37.88 | $< 0.001$ | 37.88 | 39.70 | 0.18 | 29999 | 37.30 | 0.14 | 6703.60 |
| 200 | 17.80 | 48.40 | 0.62 | 48.40 | 50.08 | $< 0.001$ | 50.08 | 53.16 | 0.22 | 39999 | 48.88 | 0.40 | 7945.90 |

Table 3.8: Results obtained with $M_1$, $M_2$, $M_3$ and $M_4$ (orthogonal polygons).

Comparing the results obtained with the greedy strategies, $M_1$ and $M_2$, it can be noted that $M_2$ is faster and the obtained solutions are apparently similar for $n = 50$ and $n = 100$ and they are slightly better for higher values of $n$, namely $n = 150$ and $n = 200$. Concerning the metaheuristic strategies, $M_3$ and $M_4$, we can see that $M_3$ is faster for $n = 200$ and the average of hidden vertices obtained with it seems to be better (except for 50-vertex polygons, where the obtained solutions look to be similar).

So, the method $M_2$ seems to be the best of the greedy strategies. Concerning the metaheuristic strategies the method $M_3$ appears to be better than $M_4$. Comparing, now, the results obtained with $M_2$ and $M_3$, we can see that, although the $M_3$ is slower, the average of hidden vertices obtained with it looks to be higher (especially for $n = 150$ and $n = 200$).

Summing up, it seems that $M_3$ is the method for which the obtained average number of hidden vertices is the best and the only method, that is faster than it, is $M_2$. In terms of the obtained solutions, the second best method seems to be $M_2$, followed by $M_4$, and, finally, the worst looks to be $M_1$. In Figure 3.19, we can see that $M_3$ stands out among the other strategies, mainly for $n \geq 100$.

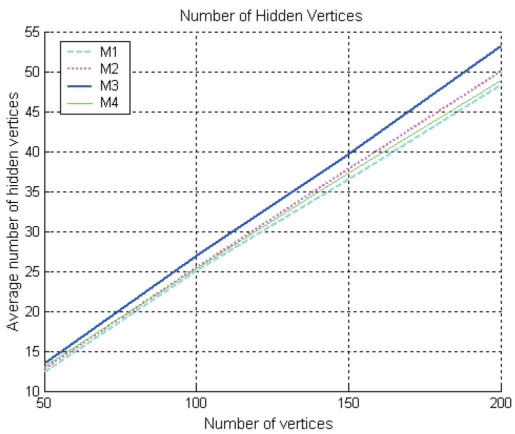
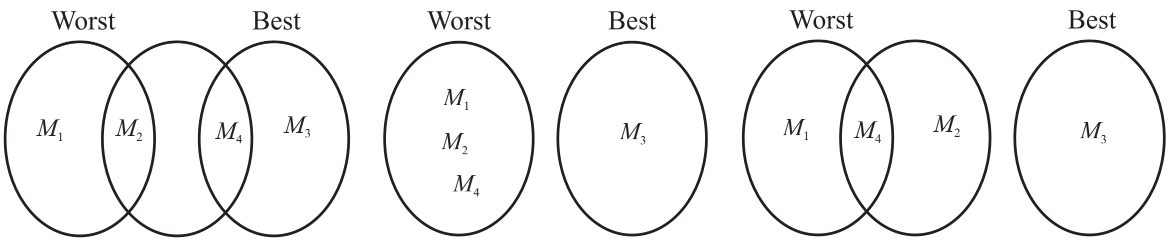



Figure 3.19: Solutions obtained with the the strategies $M_1, M_2, M_3$ and $M_4$ (orthogonal polygons).

The performed statistical study to ensure the statistically significance of the results follows. First of all it was performed a study concerning $|H|$. As we already know, the data obtained with method $M_3$ is non-normally distributed, for $n = 50, 100, 150$ and 200. Thus, the Kruskal-Wallis test has been used, which showed that there was a significant difference between the four methods, for $n = 50, 100, 150$ and 200 ($p$-value$<0.001<0.05$, for $n = 50, 100, 150$ and 200). So, multiple comparison tests were performed, whose results allowed to sort the four methods, in ascending order on $|H|$, as follows (see Figure 3.20, in this figure are illustrated Venn diagrams where each ellipse represents the methods that are not significantly different):

- for $n = 50$:
  - $M_1$, with no significant differences from $M_2$;
  - $M_2$, with no significant differences from $M_4$;
  - $M_4$, with no significant differences from $M_2$ and $M_3$;
  - $M_3$, with no significant differences from $M_4$.
- for $n = 100$:
  - $M_1$, with no significant differences from $M_2$ and $M_4$;
  - $M_3$, with significant differences from the other three methods.
- for $n = 150$ and $n = 200$:
  - $M_1$, with no significant differences from $M_4$;
  - $M_4$, with with no significant differences from $M_1$ and $M_2$;
  - $M_2$, with with no significant differences from $M_4$;
  - $M_3$, with significant differences from the other three methods.




(a) $n = 50$ (b) $n = 100$ (c) $n = 150$ and 200

Figure 3.20: Multiple comparison tests of the four methods (orthogonal polygons).

As we can see, $M_3$ obtains solutions significantly better than the other methods, except for $n = 50$, where $M_3$ is not significantly different from $M_4$. So it was concluded that the best method is $M_3$.

Now, to infer about the average of the maximum number of vertices that can be hidden on an orthogonal polygon, $M_3$ (which was considered the best method) was applied to eight sets of orthogonal polygons, each one with 50 polygons of 30, 50, 70, 100, 110, 130, 150 and

200 vertex polygons, respectively. The average of the obtained results, concerning $|H|$, is shown in Table 3.9.

| Vertices | 30 | 50 | 70 | 100 | 110 | 130 | 150 | 200 |
|---|---|---|---|---|---|---|---|---|
| $\lvert H\rvert$ | 8.5 | 13.52 | 18.68 | 26.86 | 29.58 | 34.72 | 39.70 | 53.16 |

Table 3.9: Average number of hidden vertices (orthogonal polygons).

Then, using the least squares method, the following linear adjustment was obtained, with a correlation factor of 0.9999 (see Figure 3.21):

$$f(x) = 0.2631x + 0.4624 \approx \frac{x}{3.80} + 0.4624 \approx \frac{x}{3.80}.$$

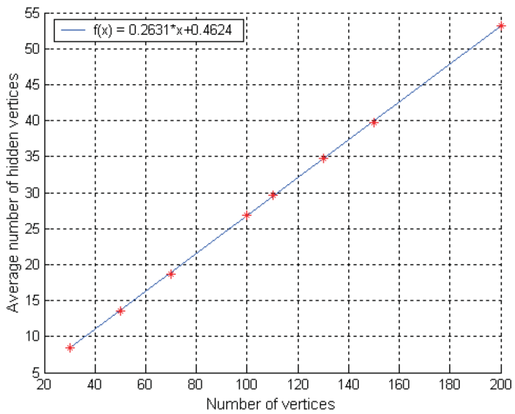
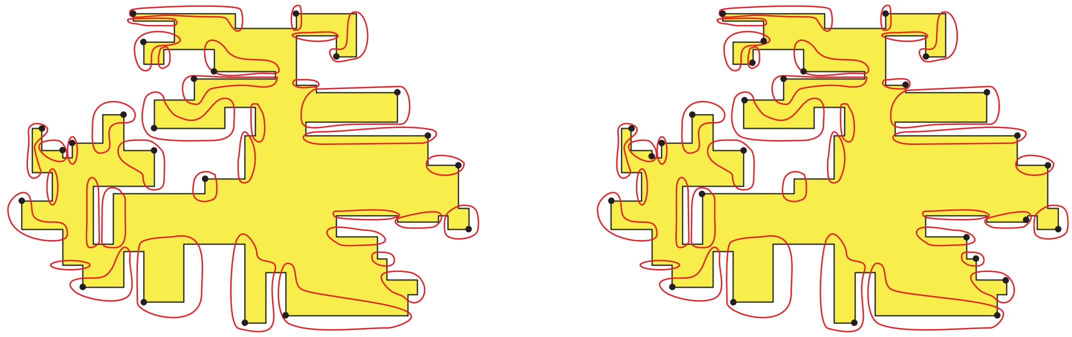



Figure 3.21: Least Squares Method (orthogonal polygons).

Thus, it can be concluded that on average, and approximately, the maximum number of hidden vertices on an orthogonal polygon $P$ with $n$ vertices is $\lceil \frac{n}{3.80} \rceil$. In order to get a quantitative measure on the quality of the calculated $|H|$ the minimum clique partitions were computed, to our instances (the eight sets of polygons described above). The ratio between minimum clique partition $C$ obtained with $A_1$ and the larger $H$ (see section 3.3) never exceeded 1.54 (with an average of 1.29 for the universe of 320 polygons). That implies that our algorithm has an approximation ratio less than or equal to 1.54.

Figure 3.22 shows a snapshot of one of the 100-vertex orthogonal polygons that has been tested with our software. In Figure 3.22(a) it is illustrated the minimum clique partition obtained with $A_1$ (see section 3.3), $|C| = 37$, and the solution obtained with the worst method $(M_1)$, $|H| = 23$. In Figure 3.22(b)) are illustrated the minimum clique partition obtained with $A_1$ and the solution obtained with the best strategy $(M_3)$, $|H| = 28$. The black dots represent the obtained hidden vertices.



(a) (b)

Figure 3.22: Example of a tested orthogonal polygon with $n = 100$. Clique partition obtained with algorithm $A_1$ and hidden vertex sets obtained with: (a) method $M_1$ and (b) method $M_3$.

Figure 3.23 shows a 20-vertex staircase polygon that has been tested with our software. In this Figure, it is illustrated the clique partition obtained with $A_1$, $|C| = 10$, and the solution obtained with $M_3$, $|H| = 10$. Note that, in this example, $\frac{|C|}{|H|} = 1$, that is, $M_3$ obtained the optimal solution (the black dots represent the obtained hidden vertices).

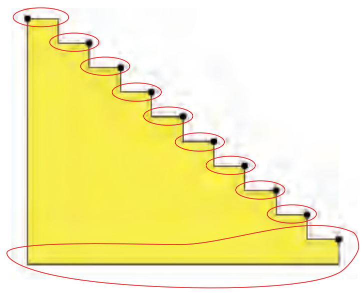


Figure 3.23: The $C$ and $H$ sets in a staircase polygon, with $n = 20$, obtained with $A_1$ and $M_3$.

## 3.5 Concluding Remarks

Four approximation algorithms were designed and implemented to tackle the Maximum Hidden Vertex Set problem on polygons. The first two, $M_1$ and $M_2$, are greedy strategies, and the other two, $M_3$ and $M_4$, are based on the general metaheuristics simulated annealing and genetic algorithms, respectively. It was also developed a method to compute clique partitions, whose cardinality approximates the minimum clique partition of the visibility graph of a polygon, allowing to obtain provable performance bounds in terms of approximation ratios.

From the performed computational experiments it can be concluded that:

(1) Concerning the analysis of the SA parameters, both for arbitrary and orthogonal polygons, the election of the initial temperature was not influential and though the FSA temperature reduction increases the response time of the algorithm, it improves the obtained solutions. So, if we are looking for a solution closer to the optimum, it seems to be more suitable to choose a slow decreasing of the temperature. Note, however, that $T_f$ was considered equal to 0.005. Clearly, if this value is reduced, the solutions obtained by the algorithms will be improved. And, this improvement is likely to be more evident for those cases that are farther from find the optimal solution, that is, for faster temperature decreases (VFSA and Geometric decreases). Therefore, if an acceptable and rapid solution is wished, this will be possible to obtain by reducing the value of $T_f$ and using a rapid decrease. It is also important to note that all alternatives regarding the parameters of the SA metaheuristic that could be explored are almost "infinite". In this work it was attempted to find references for these parameters, noting that a more exhaustive study in future investigations might improve the obtained results.

(2) As to the four approximation algorithms, the best one is method $M_3$, both for arbitrary and orthogonal polygons. It can also be concluded that on average, and approximately, the maximum number of vertices that can be hidden on a given arbitrary or orthogonal polygon, with $n$ vertices, was observed to be $\lceil \frac{n}{3.74} \rceil$ or $\lceil \frac{n}{3.80} \rceil$, respectively. The hidden vertex sets obtained with $M_3$ can be considered very satisfactory in the sense that they were always close to optimal. This method has an approximation ratio less than or equal to 1.62, for arbitrary polygons, and less than or equal to 1.54, for orthogonal polygons.

As a conclusion, the metaheuristic SA proved to behave very well in solving the MAXIMUM HIDDEN VERTEX SET problem.

# Chapter 4

# Minimum Vertex Guard Set Problem

The Minimum Vertex Guard Set problem is a variant of the original Art Gallery Problem, which is the pioneer of the guarding problems. These problems have been extensively studied in the context of the Art Gallery Problems in Computational Geometry. The original Art Gallery Problem was proposed by Victor Klee in 1973: *How many stationary guards are sufficient to cover an art gallery room with n walls?*. Informally an art gallery is modelled by a polygon $P$ with $n$ edges and a guard is considered a fixed point on $P$ with $2\pi$ range visibility. A set of guards covers a room if each point of the room could be seen by at least one guard. Thus, this problem deals with setting a minimal number of guards on a gallery room with a polygonal shape, so that they could see every point in the room. Many variations of this problem have been studied over the years, such as: where the guards may be positioned (anywhere or in specific positions, e.g., vertices), what kind of guards are to be used (e.g., stationary guards versus mobile guards), whether only the boundary or all the interior of the polygon must be guarded, and what assumptions are made on the input polygon (such as being orthogonal). The variant studied in this chapter is the problem of finding the minimum number of guards placed on vertices (vertex guards) needed to cover a given polygon $P$, which is $\mathcal{NP}$-hard both for arbitrary and orthogonal polygons [8, 115].

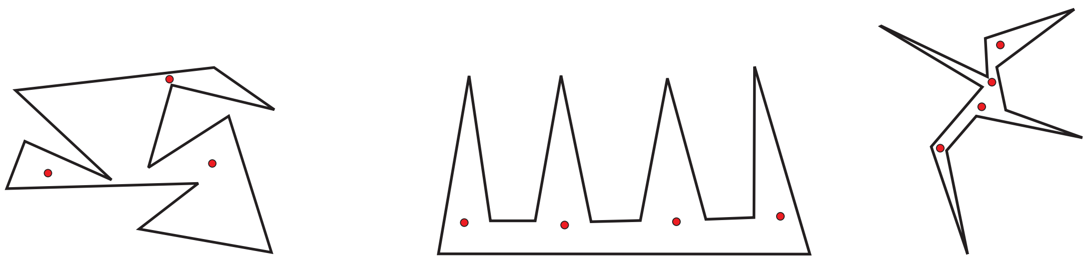
The chapter is divided in five sections. In section 4.1 the problem is described and formalized. In section 4.2 five approximation algorithms to tackle the problem are developed. The first is a greedy algorithm (subsection 4.2.2), the second is based on the simulated annealing metaheuristic (subsection 4.2.3), the third on the genetic algorithms metaheuristic (subsection 4.2.4) and the last two are hybrid algorithms, based on the simulated annealing and genetic algorithms metaheuristics (subsection 4.2.5). As the optimal solution for the MVGS($P$) problem is not known, in section 4.3 it is developed a method that allows to get the performance ratio of the developed approximation strategies. In section 3.5 some conclusions are presented.

Finally, let us mention that some of the results presented in this chapter have been published in [21, 22].

## 4.1 Problem Description

Let $P$ be a polygon with $n$ vertices, $v_0, v_1, \ldots, v_{n-1}$. Remember that, being $p \in P$, the set of all points $q \in P$ that are visible to $p$ is called visibility polygon of $p$ and it is denoted by $Vis(p, P)$. That is, $Vis(p, P) = \{q \in P : p \text{ sees } q\}$.

For any given fixed polygon $P$, there is a minimum number of guards that is necessary to cover it. For example, we can easily see that 3 guards are needed to cover the 12-vertex polygon illustrated in Figure 4.1 (a). An obvious question that arises is: is 3 the number of guards always sufficient to cover any 12-vertex polygon? The answer is: No! As we can see, the 12-vertex polygon illustrated in Figure 4.1 (b) requires 4 guards. So, what is the minimum number of guards that is ever sufficient to guard any polygon with 12 vertices?



(a) (b)

Figure 4.1: Three 12-vertex arbitrary polygons: (a) requires 3 guards; (b) require 4 guards.

This is what the original Art Gallery Problem asks for: Given $n$, it must be expressed as function of $n$, the smallest number of guards that is sufficient to cover any $n$-vertex polygon. This number is said to be necessary and sufficient for coverage: necessary for at least one $n$-vertex polygon and sufficient for any $n$-vertex polygon.

Let $P$ be a polygon with $n$ vertices. Let $g(P)$ be the smallest number of guards needed to cover $P$:

$$g(P) = \min\{|S| : S \subset P,\ P = \bigcup_{x \in S} Vis(x, P)\}.$$

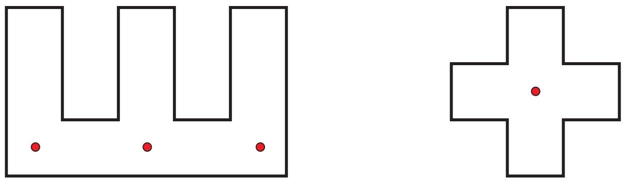
Denote by $G(n)$ the maximum of $g(P)$ over all polygons with $n$ vertices:

$G(n) = \max\{g(P) : P \in P_n\}$, where $P_n$ denotes the set of all polygons with $n$ vertices.

Thus, $G(n)$ guards always suffice to cover any $n$-vertex polygon, and are necessary to cover at least one $n$-vertex polygon. We will rewrite this as: $G(n)$ guards are always sufficient and occasionally necessary, or just sufficient and necessary. So, Klee's original Art Gallery Problem is to determine $G(n)$ [102].

The first proof that $G(n) = \lfloor \frac{n}{3} \rfloor$ was given by Chvátal and is known as the Art Gallery Theorem [34]. Three years later Fisk gave a simpler proof for this bound [59].

Concerning orthogonal polygons, the Orthogonal Art Gallery Theorem was first formulated and proved by Kahn *et al.* [77]. It states that, if the study is restricted to orthogonal polygons $G(n) = \lfloor \frac{n}{4} \rfloor$. That is, $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover any orthogonal polygon with $n$ vertices. Figure 4.2 shows two 12-vertex orthogonal polygons, which require 3 guards (Figure 4.2 (a)) and 1 guard (Figure 4.2 (b)).



(a) (b)

Figure 4.2: Two 12-vertex orthogonal polygons: (a) requires 3 guards; (b) requires 1 guard.

The Art Gallery and the Orthogonal Art Gallery theorems give a combinatorial solution since they respond to the generality of the polygons with $n$ vertices (arbitrary and orthogonal). However, as stated above, not all the polygons with $n$ vertices require the established number of guards. This reasoning justifies the following algorithmic problem: *given a $n$-vertex polygon $P$ (arbitrary or orthogonal) determine the minimum number of guards necessary to cover it.* This problem is designated by Minimum Guard Set problem and it is $\mathcal{NP}$-hard both for arbitrary and orthogonal polygons [8, 115]. If the guards are restricted to the vertices of $P$ (vertex guards), the combinatorial bounds established by the above theorems remain valid. Besides, the algorithmic problem of finding the minimum number of vertex guards needed to cover a given polygon is also $\mathcal{NP}$-hard for arbitrary polygons [84] and for orthogonal polygons [115]. This variant of the Minimum Guard Set problem is recognized as the Minimum Vertex Guard Set problem.

**Definition 4.1** *A given set $G$ of vertices of $P$ is a* ***vertex guarding set*** *for $P$ if they cover $P$, i.e., if $\bigcup_{v \in G} Vis(v, P) = P$. A vertex guarding set for $P$ is denoted by $G$ and its cardinality by $|G|$.*

The Minimum Vertex Guard Set problem will be denoted by MVGS($P$) and can be stated as follows:

**MVGS($P$)**
**Input**: A polygon $P$ with $n$ vertices.
**Question**: What is the minimum number of vertex guards necessary to cover $P$?

Given that the MVGS($P$) problem is $\mathcal{NP}$-hard, in this chapter it will be proposed some approximation methods to tackle it. A useful result related to this problem, and which will be used to develop the approximation methods, was proven by Urrutia [129]:

**Proposition 4.1** *Let $P$ be a polygon with $r$ reflex vertices. Then $r$ guards, placed on the reflex vertices of $P$, are always sufficient and occasionally necessary to cover $P$.*

In the next section the proposed approximation methods will be described.

## 4.2 Approximation Methods

Five approximation algorithms were developed to determine a vertex guarding set $G$, whose cardinality approximates the minimal number of vertex guards needed to cover a given polygon $P$. The first is a greedy algorithm, which is designated by $M_1$; the second is based on the simulated annealing metaheuristic, which is called $M_2$; the third is based on the genetic algorithms metaheuristic, which is named $M_3$ and the last two are hybrid algorithms, which are called $M_4$ and $M_5$.

### 4.2.1 Pre-processing Step

As the visibility polygon of each of the vertices of the polygon will be needed more than once along the five approximation algorithms, a pre-processing step is performed to compute and store the visibility polygons of the vertices , that is, $Vis(v_i, P)$ is computed and stored, for all $v_i \in V_P$. This information will decrease the algorithms' runtime because each time a vertex visibility polygon is required it is not necessary to calculate it again. To compute the visibility polygon of $v_i$, it was implemented the linear algorithm developed by Lee [85].

### 4.2.2 Greedy Strategy

A natural way to find a vertex guarding set is to do so with a greedy algorithm. Remember that that, this type of algorithms is simple, straightforward and most of the times quite efficient. In general, a greedy algorithm starts with a candidate set $C$ and with an initially empty solution $G$. Then the candidates are added one by one until a solution is obtained, selecting a candidate from $C$ in each step according to a certain rule. By proposition 4.1, it is clear that $C$ can be defined as the set of reflex vertices of $P$ and for each $v_i \in C$ we only add it to $G$ if the points seen by $v_i$ are not seen by the vertices already on $G$.

The method described above allows to obtain a vertex guarding set $G$, whose cardinality approximates the minimal number of vertex guards needed to guard a given polygon $P$. However it may be possible to find a set $U \subset G$ such that $\bigcup_{v\in G\setminus U} Vis(v, P) = P$. So, after the described greedy strategy we iteratively remove those redundant guards. This is done in the following way: for each $v_i \in G$, if $P$ is covered by $G \setminus \{v_i\}$, then $v_i$ is removed from $G$; otherwise it remains as part of the set $G$ (see Algorithms 4.1 and 4.2 for detailed descriptions).

---

**Algorithm 4.1** Greedy strategy (method $M_1$)

---

**Input:** A polygon $P$ with $n$ vertices;
a vector with $Vis(v_i, P)$, for $i = 0, \ldots, n-1$.
**Output:** A vertex guarding set, $G$

$G \leftarrow \emptyset$
Create a vector with the reflex vertices of $P$, $V_r$
$G \leftarrow V_r[0]$
$i \leftarrow 1$
**while** $((i < |V_r|)$ and ($P$ is not covered)) **do**
  **if** $Vis(V_r[i], P) \not\subset \bigcup_{v \in G} Vis(v, P)$ **then**
    $G \leftarrow G \cup \{V_r[i]\}$
  **end if**
  **if** $\bigcup_{v \in G} Vis(v, P) = P$ **then**
    $P$ is covered
  **end if**
**end while**
Remove redundant vertices from $G$ // see Algorithm 4.2
**return** $G$

---

**Algorithm 4.2** Removing Redundant Vertices

---

**Input:** A polygon $P$ with $n$ vertices;
a vertex guarding set, $G' = \{v_1, \ldots, v_k\}$;
a vector with $Vis(v_i, P)$, for $i = 0, \ldots, n-1$.
**Output:** A vertex guarding set, $G$

$G \leftarrow G'$
**for** $i = 1$ to $k$ **do**
  **if** $\bigcup_{v \in G \setminus \{v_i\}} Vis(v, P) = P$ **then**
    $G \leftarrow G \setminus \{v_i\}$
  **end if**
**end for**
**return** $G$

---

In the sequel, the greedy strategy is, sometimes, designated by $M_1$.

### 4.2.3 Simulated Annealing Strategy

Like in subsection 3.2.2 in Chapter 3, next it will be described how the SA parameters are defined to suit the MVGS($P$) problem. Note that, after defining the SA parameters, an approximation algorithm is obtained that allows to get a vertex guarding set $G$. However, as in method $M_1$, it may be possible that some elements of $G$ are redundant. Thus, the final step of the SA strategy is the removal of these elements (see Algorithm 4.2).

#### 1. Specific Parameters

**Solution space.** The solution space, set $S$, to the MVGS($P$) problem is the set of all vertex guarding sets for $P$. Therefore, $S$ is a finite set and can be represented by $S = \{S_1, S_2, \ldots, S_m\}$, where each $S_i$ is defined in a similar way to the one defined for the MHVS($P$) problem (see subsection 3.2.2). That is, $S_i = v_0^i v_1^i \ldots v_{n-1}^i$ for $i = 1, \ldots, m$. In this way, each candidate solution $S_i$ is represented by a chain of length $n$, where $v_j^i$, $j \in \{0, \ldots, n-1\}$, represents the vertex $v_j \in P$ and its value is 0 or 1. If $v_j^i = 1$ then the vertex $v_j$ is a vertex guard; otherwise ($v_j^i = 0$) the vertex $v_j$ is not a vertex guard. In order to illustrate these notions, a small example is presented in Figure 4.3.

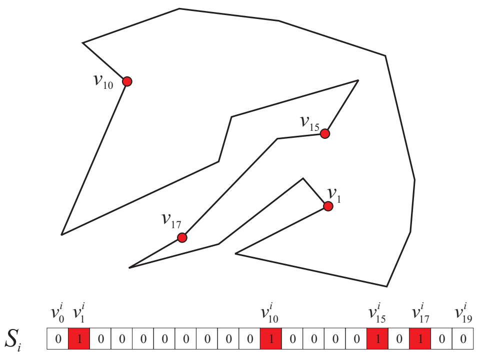



Figure 4.3: An element $S_i \in S$ (for a 20-vertex polygon) and its representation. Filled dots represent the vertex guards.

**Objective function.** The objective function $f : S \to \mathbb{N}$ assigns to each element of $S$ a natural value. For each $S_i \in S$, the function $f$ is defined by $f(S_i) = \sum_{j=0}^{n-1} v_j^i$, representing the cardinality of the vertex guarding set.

**Neighbourhood of each solution.** For the MVGS($P$) problem the generation of a neighbour $S_j$ of candidate solution $S_i \in S$ is done as follows. Let $S_i = v_0^i \ldots v_{n-1}^i \in S$, a natural number $t \in [0, n-1]$ is randomly generate (following a uniformly distribution), and then if:

- $v_t^i = 1$ then $v_t^j$ is set to 0, i.e., $v_t^j = 0$. If this new solution is a valid one, then it is accepted since the solution was improved; else the obtained solution is discarded.

- $v_t^i = 0$ then $v_t^j$ is set to 1, i.e., $v_t^j = 1$, and this new solution is accepted with a probability, since it is a worse solution.

Figure 4.4 exemplifies the generation of two neighbours of the solution illustrated in Figure 4.3.

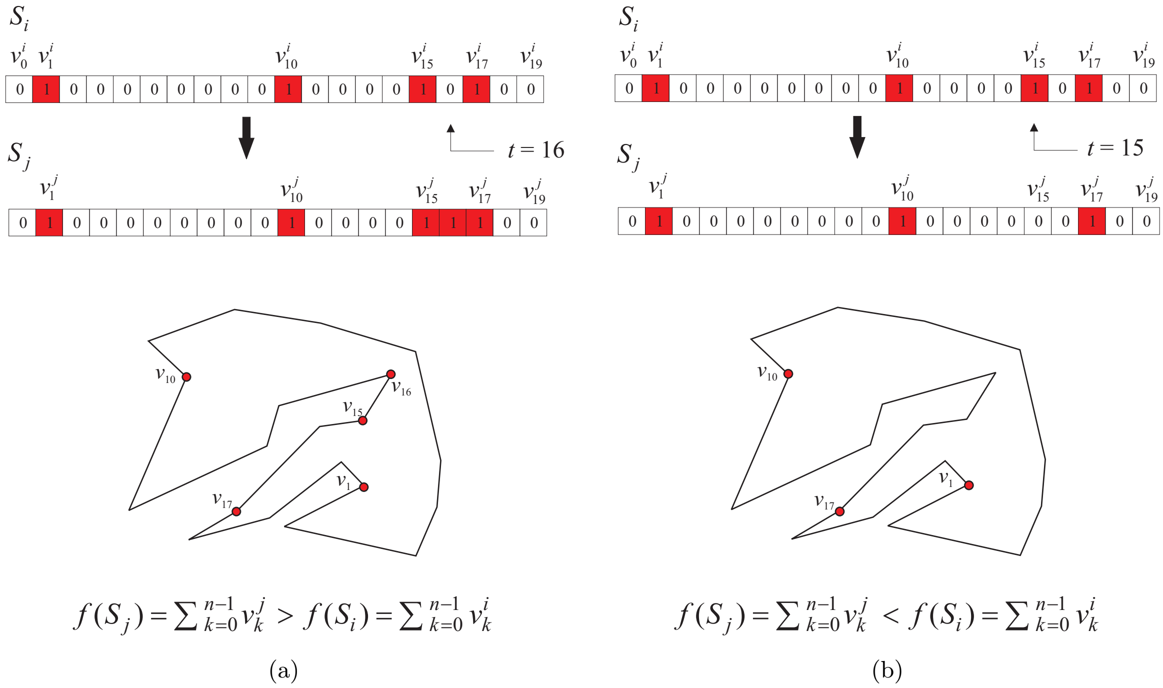



Figure 4.4: Generation of $S_j$, a neighbour of $S_i$: (a) $S_j$ is a worse solution; (b) $S_j$ is a better solution.

**Initial solution.** For the initial solution, in view of the proposition 4.1, all reflex vertices of $P$ were considered as vertex guards. Figure 4.5 exemplifies the initial solution of the polygon illustrated in Figure 4.3.

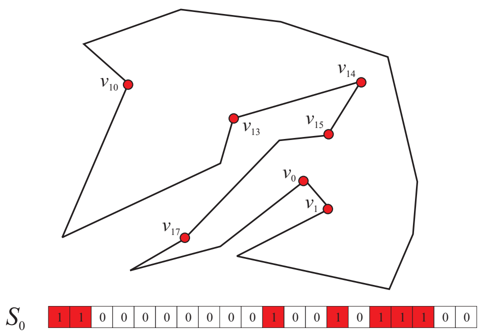



Figure 4.5: Initial solution.

**2. Generic Parameters**

**Initial temperature ($T_0$).** A comparative study was performed taking into account two different types of $T_0$:

1. An initial temperature dependent on the number of vertices of the polygon $P$, $T_0 = f(n)$ (in the study was considered $T_0 = n$ and $T_0 = \frac{n}{4}$);
2. A constant initial temperature: $T_0 = 500$.

**Temperature decrement rule.** As for the MHVS($P$) problem (see subsection 3.2.2), an analysis was made on three different types of temperature decrement rules:

1. $T_{k+1} = \frac{T_0}{1+k}$ (Fast Simulated Annealing (FSA) decrease);
2. $T_{k+1} = \frac{T_0}{e^k}$ (Very Fast Simulated Annealing (VFSA) decrease) and
3. $T_{k+1} = \alpha T_k$, where $0 < \alpha < 1$ (Geometric decrease). In the performed study it was chosen $\alpha = 0.9$.

**Number of iterations at each temperature ($N(T_k)$).** As for the MHVS($P$) problem (see subsection 3.2.2), for the MVGS($P$) problem was chosen $N(T_k) = \lceil T_k \lceil$ and this ensures that there are more iterations while the temperatures are high, i.e., when the solutions are still far from optimal.

**Termination condition.** As stated in subsection 2.1.1, theoretically, the search process should stop when a *frozen state* is achieved, i.e., when $T_k = 0$. Nevertheless, generally, it is possible to finish with a final temperature, $T_f$, greater than zero without quality loss in the solution. For instance, the search can be halted when it ceases to make progress. Lack of progress can be defined in different ways, but a useful basic definition is: no improvement (i.e. no new best solution) registered in the last $l$ consecutive series of temperatures, combined with the acceptance ratio falling below a given (small) value $\varepsilon$ [28]. In this sense, the termination condition chosen for the developed algorithm consists in finishing the search when the temperature is less than or equal to 0.005 or when during the last $l = 3000$ consecutive series of temperatures no new better solution is obtained and the percentage of accepted solutions is less than $\varepsilon = 2\%$ (the values of $l$ and $\varepsilon$ were chosen experimentally).

### 4.2.4 Genetic Algorithms Strategy

As in subsection 3.2.3, it will follow the description of how the GA parameters are defined to suit the MVGS($P$) problem. As in the greedy and SA strategies, the final step of the GA strategy is the removal of the redundant vertex guards elements (see Algorithm 4.2).

**Encoding.** The genetic representation of the candidate solutions to the MVGS($P$) problem is similar to the representation of each candidate solution, $S_i$, on SA strategy (see previous subsection). An individual $I$ is represented by a chain of 0's and 1's, with length $n$, i.e., $I = g_0 g_1 \dots g_{n-1}$, where each element $g_i$ (gene) represents a vertex of the polygon. That is, the $i^{th}$ gene represents the vertex $v_i \in P$ and its value is 0 or 1. If $g_i = 1$ then the vertex $v_i$ is marked as a vertex guard; otherwise ($g_i = 0$) the vertex $v_i$ is not a vertex guard

**Initial population.** Here it was chosen for the population size the number of reflex vertices of the polygon, in this way linking the problem input with the elements of the metaheuristic. Thus, the population for the generation $t$ is represented by: $P(t) = \{I_0^t, I_1^t, \dots, I_{r-1}^t\}$, where each $I_i^t$ represents an individual belonging to the population $P(t)$ and $r$ is the number of reflex vertices of the polygon $P$.

Concerning the initial population, remember that an individual represents a candidate solution for the MVGS($P$) problem, i.e., each individual must be a vertex guarding set. By Proposition 4.1, being $P$ a polygon with $r$ reflex vertices, $r$ guards placed on the reflex vertices of $P$ are always sufficient to cover $P$. Thus, let $R = \{u_0, u_1, \dots, u_{r-1}\}$ be the set of reflex vertices of $P$. To create the initial population, $P(0)$, each of the $r$ individuals is generated in the following way: $\forall i \in \{0, \dots, r-1\}$, if $R \setminus \{u_i\}$ is a vertex guarding set all the vertices of $R \setminus \{u_i\}$ are marked as vertex guards; otherwise the vertices of $R$ are marked as vertex guards. For example, Figure 4.6 presents a polygon with 20 vertices and its initial population, $P(0) = \{I_0^0, I_1^0, \dots, I_6^0\}$.

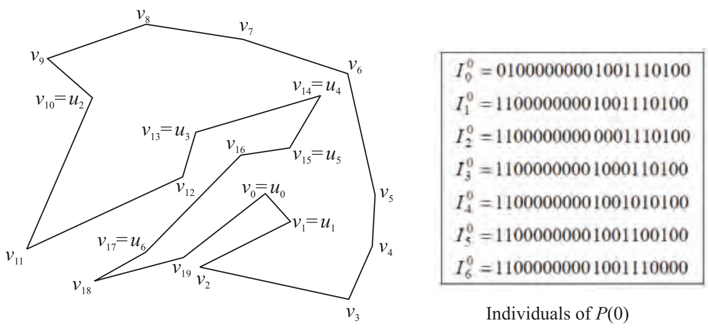



Figure 4.6: Polygon with $n = 20$ ($r = 7$) and its initial population.

**Fitness function.** This function was defined in a similar way to the objective function defined for SA strategy. For each $I$, $f$ is defined by $f(I) = \sum_{i=0}^{n-1} g_i$, representing the cardinality of the vertex guarding set. Note that this function does not assigns higher values to the solutions closer to the optimal one(s), as described in Chapter 2, subsection 2.1.2. On the contrary, it assigns lower values to the solutions closer to the optimal one(s), however the used selection methods were changed in order to reflect this behaviour.

**Selection.** In the developed algorithm it was made a comparative study taking into account the two common methods: roulette wheel selection and tournament selection (see Chapter 2, subsection 2.1.2). The roulette wheel selection was used to choose two individuals to be parents in crossover. In the employed roulette wheel method the probability of an individual be selected to be parent is inversely proportional to its fitness, that is, the lower the fitness is, the higher the probability of being selected. For that, in the implementation of roulette wheel selection scheme described in Chapter 2 (subsection 2.1.2) $f_i$ was replaced by $\frac{1}{f_i}$. In the tournament selection, $m$ individuals are randomly selected and the best one is chosen for parenthood. This selection scheme is performed $k$ times to choose $k$ parents. It was used a binary approach ($m = 2$) to select two individuals to be parents in crossover ($k = 2$). In this way, two pairs of individuals are randomly selected and then the parents are the individuals with the lowest fitness value in each pair.

**Crossover.** A comparative study was done with four different types of crossover: single point crossover, two-point crossover, uniform crossover and a variant of the single point crossover where the generated children cannot be clones of the parents (see Chapter 2, subsection 2.1.2). In any crossover method we only generate one child from two parents (see Figures 4.7, 4.8 and 4.9).

Remember that the uniform crossover decides, with some probability, which parent will contribute to each of the gene values of the child chromosomes. The chosen probability was 0.5, which allows that approximately half of the child genes are inherited from one parent and the other half from the other.

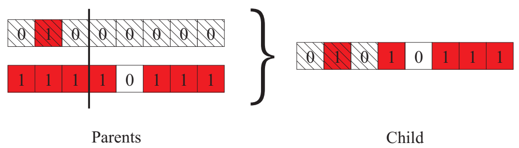



Figure 4.7: Single point crossover.

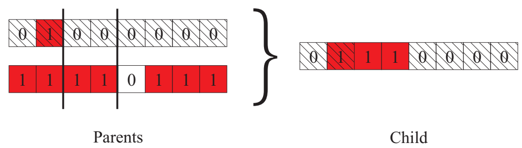



Figure 4.8: Two-point crossover.

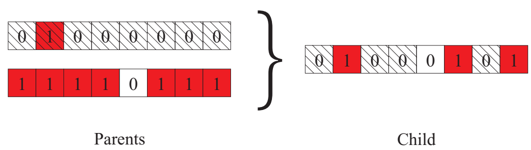



Figure 4.9: Uniform crossover.

For the crossover probability was chosen $p_c = 0.8$ (this value was chosen experimentally). Note that the child resulting from any of the described crossover methods may not be valid (i.e., it may not correspond to a vertex guarding set), see Figure 4.10, in this case it was decided not to accept the child.

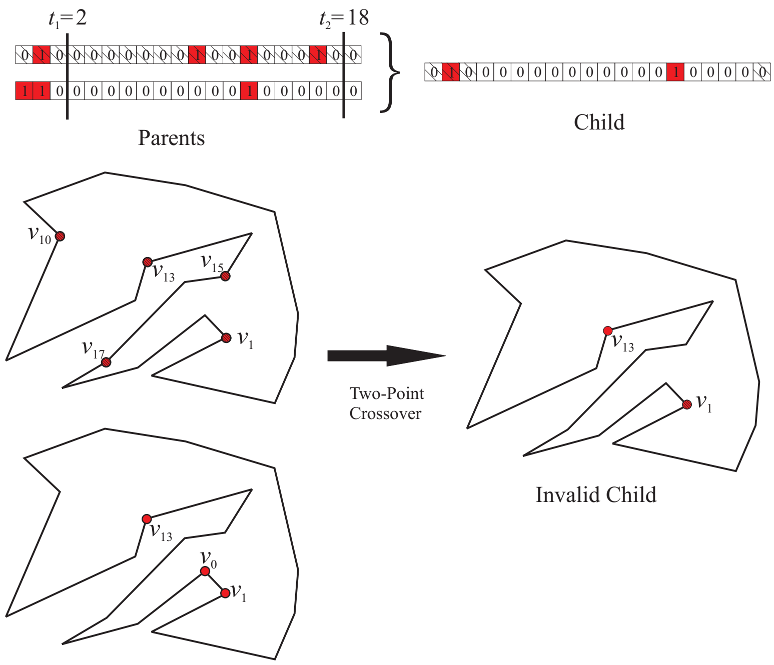



Figure 4.10: Generation of an invalid child.

**Mutation.** Since a binary encoding is used, the action of the mutation operation is relatively simple. For each gene it merely flips its value from zero to one or vice versa, with a mutation probability $p_m$ (see Figure 4.11).

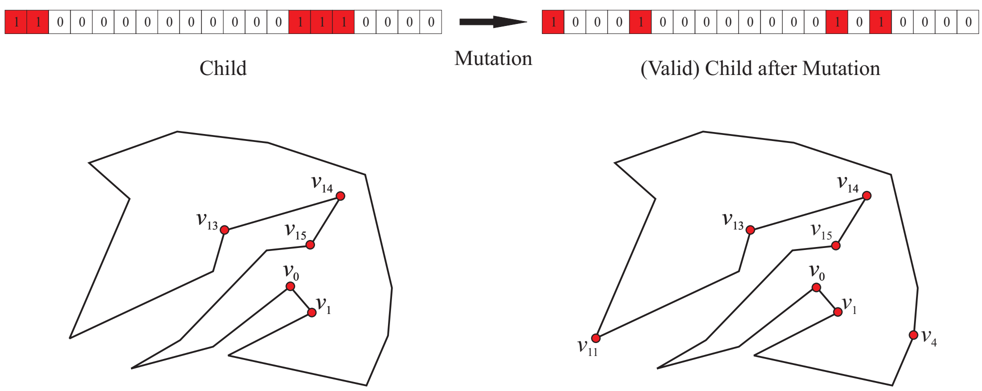



Figure 4.11: Mutation.

In the developed algorithm the mutation is applied to the child obtained with the crossover operation, with $p_m = 0.05$ (this value was experimentally chosen). As in the crossover, if the resultant individual is not valid it is discarded.

**Population generation.** As there are many different ways to generate a new population, it was used a common one: select the worst individual of the population and replace it by the child obtained at the crossover (steady-state reproduction).

**Population evaluation.** The fitness of a population was considered as the minimum value of the fitness function when applied to all individuals of the population, that is, $F(P(t)) = \min\{f(I_0^t), \ldots, f(I_{n-1}^t)\}$.

**Termination condition.** For the termination condition it was considered that if the fitness of the population $F(P(t))$ remains unchanged for a number of generations $h$, the search will stop. To define this parameter, several tests were made varying the value of $h$. It was observed that from $h = 500$ the quality of the solution does not improve much. So, it was chosen $h = 500$ in the developed algorithm.

### 4.2.5 Hybrid Strategies

To solve the MINIMUM VERTEX GUARD SET problem it was, also, developed two different combinations of GAs and SA metaheuristics, that is, two different hybrid metaheuristics. Although there are many different ways to hybridize these two metaheuristics, in this work it was chosen two different hybridizations (Chapter 2, section 2.2). In the first one, for the initial population of a genetic algorithm, $r$ individuals are generated, which are obtained by running a SA strategy $r$ times. Figure 4.12 illustrates this method. In this way, it can be observed how a GA behaves on including high quality solutions in the initial population.

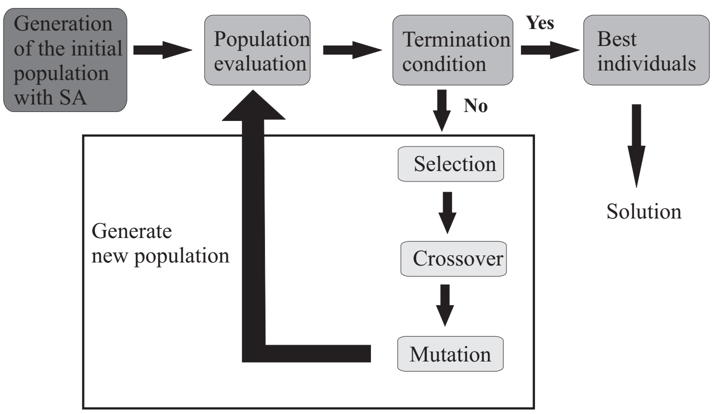



Figure 4.12: First hybrid strategy.

In the second method, a SA strategy is used as a genetic operator of a GA strategy. As the standard genetic operators, this one occurs with a certain probability $p_{sa}$. In the experimental evaluation was used $p_{sa} = 0.01$ (see Figure 4.13). This allows to observe how a GA behaves on reinforcing the intensification/exploitation during the search process.

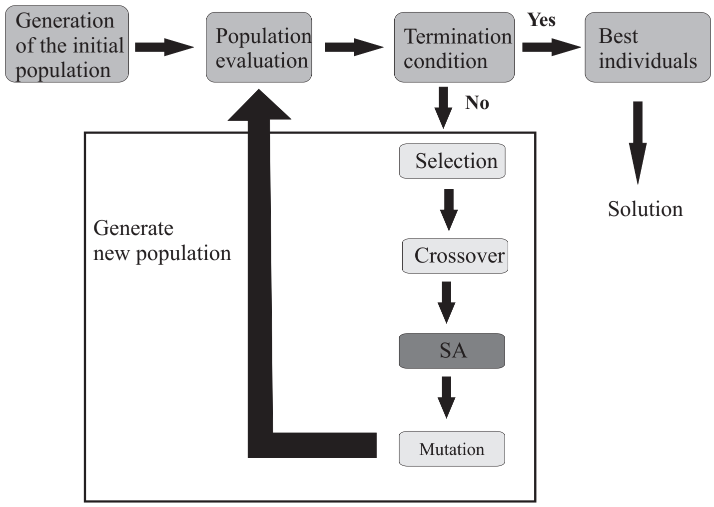



Figure 4.13: Second hybrid strategy.

## 4.3 Greedy Strategies for visibility-independent sets

As previously mentioned, the MVGS($P$) problem is $\mathcal{NP}$-hard both for arbitrary and orthogonal polygons [8,115], so its optimal solution is unknown. Such as for the MHVS($P$) problem (see Chapter 3, section 3.3), it was conducted an experimental analysis on the performance of the develloped algorithms. For that, it was developed a method to compute a lower bound on the optimal number of vertex guards for each instance in the performed experiments.

First, it was considered the concept of *visibility-independent set.*

**Definition 4.2** *Let $P$ be a $n$-vertex polygon. A* ***visibility-independent set*** *is a finite set of points on $P$, $IS \subset P$, such that the visibility polygons of its elements are pairwise disjoint, i.e., $\forall p, q \in IS, Vis(p, P) \cap Vis(p, P) = \emptyset$. Its elements are called* ***visibility-independent points*** *and its cardinality is denoted by $|IS|$* [11].

The example given in Figure 4.14 illustrates a visibility-independent set of cardinality 3.

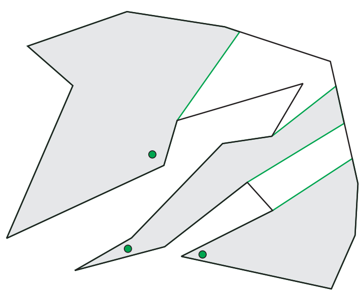


Figure 4.14: Visibility-independent set. Green dots represent visibility-independent points.

It is easy to verify that no single vertex guard is able to see more than one point of $IS$, consequently $\forall G, IS, |G| \geq |IS|$ . It can be easily concluded that $g(P) \geq is(P)$, where $g(P)$

is the number of vertex guards in a minimum-cardinality of a vertex guarding set of $P$ and $is(P)$ is the number of points in a maximum-cardinality of a visibility-independent set of $P$. Thus, $is(P)$ is a lower bound on the optimal number of vertex guards on $P$.

Nevertheless, the problem of determining this lower bound is also $\mathcal{NP}$-hard [11]. So, it was developed approximation algorithms to determine approximate solutions (which will be described later on). Let $|IS|$ be the cardinality of an approximate solution

$$|IS| \leq is(P) \leq g(P) \leq |G|, \forall P. \tag{4.1}$$

If there is a constant $c \in \mathbb{R}^+$ such that $|G| \leq c \times |IS|$, for any polygon $P$, it can be said that the approximation algorithm used to obtain $G$ has an approximation ratio of $c$ [13]. In other words, the approximate solution $|IS|$ is at most $c$ times the optimal solution $g(P)$. In fact,

$$|G| \leq c \times |IS| \Rightarrow |G| \leq c \times g(P). \tag{4.2}$$

As stated above, approximation algorithms were developed to find visibility-independent sets. These algorithms are greedy strategies, that is, they start with a set of candidates $C$ (not visibility-independent), then they add visibility-independent points one by one until a solution $IS$ is obtained ($IS$ initially is an empty set), selecting in each step a point from the candidate set $C$, according to a certain rule.

The candidate set used is the one proposed by Amit, Mitchell, and Packer [11], which is $C = C_1 \cup C_2$, where $C_1$ denotes the convex vertices of $P$ and $C_2$ denotes the midpoints of the edges incident on two reflex vertices. Concerning the rule to select the points, it was applied three different alternatives which result in three different greedy algorithms: $A_1$ , $A_2$ and $A_3$.

**Algorithm $A_1$:** For each candidate $c_i \in C$, the area of $Vis(c_i, P)$ is calculated. In each step the candidate, whose visibility polygon has the smallest area, is selected.

**Algorithm $A_2$:** For each candidate $c_i \in C$, $Vis(c_i, P)$ is computed. The number of intersections with the visibility polygons of the other candidates is calculated. In each step the candidate that has the smallest number of intersections is selected.

**Algorithm $A_3$:** For each candidate $c_i \in C$, the number of candidates it sees is determined. In each step the candidate that sees the smallest number of points in $C$ is selected. This method is one of the methods developed in [11].

In all these algorithms, after adding a point to $IS$, are removed from $C$ all the candidates $c_j$ such that $Vis(c_j, P)$ intersects the union of the visibility polygons of the elements of $IS$.

The algorithms stop when the set $C$ is empty. Algorithm $A_1$ is illustrated below.

---

**Algorithm 4.3** Computing $IS$ (greedy algorithm $A_1$)

---

**Input:** A polygon $P$ with $n$ vertices

**Output:** A visibility-independent set, $IS$

1. $IS \leftarrow \emptyset$
2. $C \leftarrow C_1 \cup C_2$
3. **for** each $c \in C$ **do**
4. calculate the area of $Vis(c, P)$
5. **end for**
6. **while** $C \neq \emptyset$ **do**
7. choose the $c_i \in C$ whose $Vis(c_i, P)$ has the smallest area
8. $IS \leftarrow IS \cup \{c_i\}$
9. remove $c_i$ from $C$ and all $c_j \in C$ such that $Vis(c_i, P) \cap \bigcup_{is \in IS} \neq \emptyset$
10. **end while**
11. **return** $IS$

---

Algorithms $A_2$ and $A_3$ are very similar to this one, the main differences are in steps 4 and 7. It turns out that $A_2$ and $A_3$ obtain the best and the worst results, respectively, both for orthogonal and arbitrary polygons.

The application of the greedy strategy, the SA strategy, the GA strategy and the hybrid strategies together with $A_1$, to each instance in our experiments, gives provable performance bounds in terms of approximation ratios. In the performed experiments, given a polygon $P$, the main objective is to find a small vertex guarding set $G$ and a large visibility-independent set $IS$; the obtained set $G$ approximates the optimal number of vertex guards $g(P)$ with approximation ratio $\frac{|G|}{|IS|}$. Note that, if a visibility-independent set $IS$ and a vertex guarding set $G$ are found, such that $|IS| = |G|$, then $G$ is an optimal vertex guarding set.

## 4.4 Experiments and Results

To identify which of the described approximation strategies yields the best approximate solutions in a reasonable time, they were implemented and its behaviour was tested over a large set of randomly generated polygons. In the next two subsections, subsections 4.4.1 and 4.4.2, it will be discussed the results and the conclusions from the accomplished experiments on arbitrary and orthogonal polygons, respectively.

### 4.4.1 Arbitrary Polygons

To choose the SA and the GA parameters that best fit on the MVGS($P$) problem, the experiments were made over four sets of polygons, each one formed by 40 polygons of 30, 50, 70 and 100 vertex polygons. To analyze the four methods, four sets of polygons were used, each one formed by 40 polygons of 50, 100, 150 and 200. To analyze the SA and the GA parameters the experiments were performed on polygons with fewer vertices due to the time of execution, which is relatively high for some cases. Our computational tests showed that to choose these parameters it would be sufficient to make experiments with polygons of up to 100 vertices. The other choices (related with the dimension of the sets of polygons), although not being theoretically justified, were dictated by practical reasons.

#### 4.4.1.1 Analysis of the SA Parameters

According to section 4.2.3, there are several choices for two of the SA parameters: the initial temperature ($T_0$) and the temperature decrement rule. The different combinations of the parameters values result into nine cases (see Table 4.1).

| **Cases** | |
|---|---|
| **Case 1** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 2** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 3** | $T_0 = n$ and $T_{k+1} = \alpha T_{k-1}$ ($\alpha = 0.9$) (Geometric decrease, $\alpha = 0.9$) |
| **Case 4** | $T_0 = 500$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 5** | $T_0 = 500$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 6** | $T_0 = 500$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |
| **Case 7** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 8** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 9** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |

Table 4.1: Studied cases for SA.

These nine cases were analyzed by comparing the number of vertex guards, the runtime and the number of iterations performed by each one of them. Tables 4.2, 4.3 and 4.4 presents the results obtained with the Cases 1 and 2, Cases 3 and 4 and Cases 5 and 6, respectively. These tables, as can be seen, show the average time of pre-processing in seconds ($PP$), the average number of vertex guards ($|G|$), the average runtime in seconds ($Time$) and the average number of iterations of the algorithm (*Iter.*).

In the first three cases, the selection of the initial temperature depends on the input of the problem, that is, it depends on $n$, number of vertices of $P$. It was considered $T_0 = n$, may be ground for future research studying the behaviour of approximation method for non-linear functions in the initial temperature.

| $n$ | Case 1 (FSA dec.) | | | | Case 2 (VFSA dec.) | | | | Case 3 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.12 | 5.40 | 16.40 | 4798.60 | 0.10 | 5.77 | 1.400 | 9.00 | 0.25 | 5.92 | 7.550 | 83.00 |
| 50 | 0.42 | 8.55 | 51.27 | 6550.90 | 0.35 | 9.30 | 4.97 | 10.00 | 0.60 | 9.65 | 24.45 | 88.00 |
| 70 | 0.65 | 11.80 | 108.75 | 7718.50 | 0.725 | 13.20 | 10.95 | 10.00 | 1.50 | 13.35 | 53.525 | 91.00 |
| 100 | 1.90 | 16.97 | 243.52 | 10162.00 | 1.80 | 18.65 | 24.35 | 10.00 | 1.90 | 18.92 | 117.275 | 94.00 |

Table 4.2: Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$) on arbitrary polygons.

As we can see, the best solution appears to correspond to a slow decrease in temperature (FSA decrease) with a larger number of iterations and a higher response time, i.e., the best solution in these first three cases seems to be obtained by Case 1.

In the following three cases it is going to be analyzed how the different types of temperature decreasing behave, being $T_0$ constant. As the number of vertices of the analyzed polygons is 50, 100, 150 and 200, it was chosen a constant value $T_0 = 500$. This way, we have a constant value greater than any $n$ value and considered small enough so that the algorithm is executed in a reasonable time.

| $n$ | Case 4 (FSA dec.) | | | | Case 5 (VFSA dec.) | | | | Case 6 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.07 | 4.95 | 233.72 | 39686.00 | 0.05 | 5.925 | 19.40 | 12.00 | 0.10 | 6.12 | 122.02 | 110.00 |
| 50 | 0.47 | 7.80 | 452.67 | 39001.00 | 0.37 | 9.27 | 38.52 | 12.00 | 0.30 | 9.42 | 233.55 | 110.00 |
| 70 | 0.87 | 11.12 | 726.10 | 39434.00 | 0.77 | 13.17 | 61.70 | 12.00 | 0.70 | 13.20 | 366.25 | 110.00 |
| 100 | 1.77 | 15.57 | 1133.70 | 39447.00 | 1.90 | 18.200 | 94.35 | 12.00 | 1.92 | 19.32 | 564.72 | 110.00 |

Table 4.3: Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$) on arbitrary polygons.

As we can see, the best solution in these three cases seems to be achieved by Case 4. Comparing these last three cases with the first three for the same type of temperature decrease, that is, Case 1 with Case 4, Case 2 with Case 5 and Case 3 with Case 6. We can see that the solutions provided by Case 4 seem to be better than the solutions provided by Case 1, although in Case 4 the algorithm runtime and the number of iterations also increase. Concerning Cases 2 and 5, appears that the obtained solutions are almost equal, being Case 5 slower. Finally, Case 3 is faster than Case 6 and seems to obtain slightly better solutions, for $n = 30$ and 100, being almost equal for $n = 50$ and $n = 70$. We can also see that, in general, if a solution nearer to the optimal one is searched, then it seems that it is more suitable to choose an initial temperature regardless $n$ and a slow temperature decrease (Case 4).

It should be noted that it should be expected that a geometrical decrease would produce better solutions than a rapid decrease (VFSA), what does not happen. The reason for this behaviour is the elimination of redundant vertex guards, so that the results have not considerable differences.

In the next cases it is going to be analyzed how the three temperature decreases behave if the initial temperature depends on $n$. Here it was considered $T_0 = \frac{n}{4}$. This value was chosen because it not only links $T_0$ with the algorithm input, but also it is lower than $n$, and it was wished to see how the algorithm behave under these conditions.

| $n$ | Case 7 (FSA dec.) | | | | Case 8 (VFSA dec.) | | | | Case 9 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\lvert G\rvert$ | Time | Iter. | PP | $\lvert G\rvert$ | Time | Iter. | PP | $\lvert G\rvert$ | Time | Iter. |
| 30 | 0.10 | 5.55 | 4.65 | 1399.00 | 0.125 | 5.67 | 0.52 | 8.00 | 0.07 | 5.92 | 1.97 | 69.00 |
| 50 | 0.40 | 9.02 | 16.45 | 2399.00 | 0.35 | 8.70 | 2.07 | 8.00 | 0.17 | 9.40 | 6.85 | 74.00 |
| 70 | 0.72 | 12.77 | 38.05 | 3399.00 | 0.75 | 12.325 | 4.70 | 9.00 | 0.70 | 13.70 | 15.07 | 78.00 |
| 100 | 1.92 | 18.27 | 85.42 | 4592.00 | 1.82 | 17.90 | 11.07 | 9.00 | 1.85 | 19.07 | 33.47 | 81.00 |

Table 4.4: Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$) on arbitrary polygons.)

Observing these last three cases we verify, surprisingly, that for an initial temperature $T_0 = \frac{n}{4}$ the solutions seem to improve slightly when the decrease of the temperature is fast (VFSA), particularly for $n = 100$. Again, the reason for this behaviour is the removal of redundant vertex guards. If this removal does not take place the results will reverse (the number of redundant guards is much greater in Case 8 than in Cases 7 and 9). Nevertheless, this improvement cannot be observed if the decrease of the temperature is slower and $T_0 = 500$ (Case 4), that seems to be the best case of the first six cases.

As always, a statistical study was carried out. The analysis of the data normality showed that the data obtained with Case 1 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). The $p$-values returned by the Kruskal-Wallis tests were less than 0.001 for the data obtained with the polygons with $n = 30, 50, 70$ and 100, respectively (note that, all $p$-values are less than 0.05). Then multiple comparison tests were performed to determine which pairs of averages were significantly different, and which were not. The answers provided by these tests are presented in Tables 4.5, 4.6, 4.7 and 4.8. The sign "+" indicates that the sample data (concerning $|G|$) is significantly different and the sign "-" indicates otherwise.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | - | - | - | - | + | - | - | - |
| Case 2 | - | • | - | + | - | - | - | - | - |
| Case 3 | - | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | - | + | + |
| Case 5 | - | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | - | - |
| Case 7 | - | - | - | - | - | - | • | - | - |
| Case 8 | - | - | - | + | - | - | - | • | - |
| Case 9 | - | - | - | + | - | - | - | - | • |

Table 4.5: Multiple comparison tests, of SA Cases, for 30-vertex arbitrary polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | - | + | - | - | - | - | - | - |
| Case 2 | - | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | + | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | - | - | - | + | • | - | - | - | - |
| Case 6 | - | - | - | + | - | • | - | - | - |
| Case 7 | - | - | - | + | - | - | • | - | - |
| Case 8 | - | - | + | + | - | - | - | • | - |
| Case 9 | - | - | - | + | - | - | - | - | • |

Table 4.6: Multiple comparison tests, of SA Cases, for 50-vertex arbitrary polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | - | - | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | + | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | - | - |
| Case 7 | - | - | - | + | - | - | • | - | - |
| Case 8 | - | - | - | + | - | - | - | • | + |
| Case 9 | + | - | - | + | - | - | - | + | • |

Table 4.7: Multiple comparison tests, of SA Cases, for 70-vertex arbitrary polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | - | + | + | - | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | - | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | + | - | - | + | - | - | • | - | - |
| Case 8 | - | - | - | + | - | + | - | • | - |
| Case 9 | + | - | - | + | - | - | - | - | • |

Table 4.8: Multiple comparison tests, of SA Cases, for 100-vertex arbitrary polygons.

The multiple comparison tests, also, allowed to conclude that:

- for $n = 30$, concerning
  - Cases 1, 2 and 3. The best is Case 1 not significantly different from Cases 2 and Case 3; the worst is Case 3 with no significant differences from Cases 1 and 2 (see Figure 4.15 (a));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.16);
  - Cases 7, 8 and 9. The best is Case 7 with no significant differences from Cases 8 and 9; the worst is Case 9 with no significant differences from Cases 7 and 8 (see

Figure 4.17 (a));

  - the nine cases. The best is Case 4, with no significant differences from Cases 1 and 7; the worst is Case 6, with no significant differences from Cases 2, 3, 5, 7 , 8 and 9.

- for $n = 50$, concerning

  - Cases 1, 2 and 3. The best is Case 1 with no significant differences from Case 2 and significantly better than Case 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.15 (b));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.16);
  - Cases 7, 8 and 9. The best is Case 8 with no significant differences from Cases 7 and 9; the worst is Case 9 with no significant differences from Cases 7 and 8 (see Figure 4.17 (a));
  - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 3, with no significant differences from Cases 2, 5, 6, 7 and 9.

- for $n = 70$, concerning

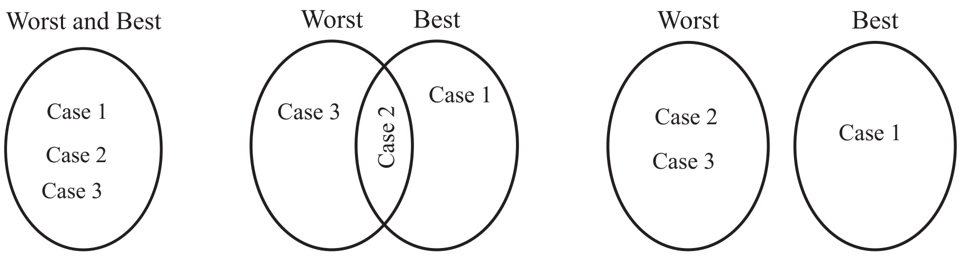
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.15 (c));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 5 with no significant differences from Case 6 (see Figure 4.16);
  - Cases 7, 8 and 9. The best is Case 8 with no significant differences from Case 7 and significantly better than Case 9; the worst is Case 9 with no significant differences from Case 7 (see Figure 4.17 (b));
  - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 9, with no significant differences from Cases 2, 3, 5, 6, and 7.

- for $n = 100$, concerning

  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.15 (c));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.16);
  - Cases 7, 8 and 9. The best is Case 8 with no significant differences from Cases 7 and 9; the worst is Case 9 with no significant differences from Case 7 and 8 (see Figure 4.17 (c));

– the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 6, with no significant differences from Cases 2, 3, 5, 7, and 9.

As can be noticed, using the multiple comparison tests, for $T_0 = n$ and $n$ small ($n = 30$) the selection of the rule to decrease the temperature is not influential on the obtained solutions, however when $n$ increases the best solutions are obtained when the temperature decrease is slow (FSA) (see Figure 4.15). For $T_0 = 500$ the best solutions always correspond to slow a temperature decrease (see Figure 4.16). Finally, for $T_0 = \frac{n}{4}$, the temperature decrease does not influence the obtained solutions (except for $n = 70$) (see Figure 4.17).




(a) $n = 30$ (b) $n = 50$ (c) $n = 70$ and 100

Figure 4.15: Multiple comparison tests, of SA Cases 1, 2 and 3 (arbitrary polygons).

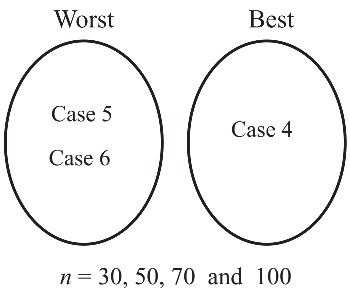
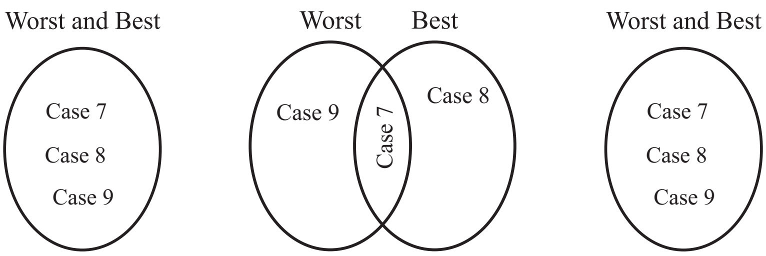



Figure 4.16: Multiple comparison tests, of SA Cases 4, 5 and 6 (arbitrary polygons).




(a) $n = 30, 50$ (b) $n = 70$ (c) $n = 100$

Figure 4.17: Multiple comparison tests, of SA Cases 7, 8 and 9 (arbitrary polygons).

Notice that for all types of initial temperature $T_0$ it should be expected that a geometrical decrease would produce better solutions than a fast decrease (VFSA), what does not happen. We can see that the obtained solutions have not significant differences (except when $T_0 = \frac{n}{4}$ and $n = 70$). As stated before, the reason for this behaviour is the elimination of redundant vertex guards.

Concerning the nine cases, if the temperature decrease is slow, the best solutions are obtained with $T_0 = 500$ and $T_0 = n$. If the temperature decrease is fast or geometric the initial temperature does not have influence, despite the observation that they seem to obtain different solutions (see Tables 4.2, 4.3 and 4.4). It can also be concluded that the best solutions are obtained with Case 4 for $n = 30, 50, 70$ and 100 and a significant difference was not found between the number of vertex guards obtained with this case and Case 1. Despite the observation that in Tables 4.2 and 4.3, Case 4 seems to outperform Case 1 in relation to the average of $|G|$, for $n = 30, 50$ and 100. So, the statistical analysis proceeded regarding the runtime. This analysis was made in a similar way and it allowed to conclude that Case 1 is significantly faster than Case 4, for $n = 30, 50, 70$ and 100. Given this, Case 1 was selected to be the best case.

Concluding, if a solution nearer to the optimal one is searched, then it is more suitable to choose a slow temperature decrease (FSA decrease) and an initial temperature $T_0 = n$. Notice, however, that the rapid decreases are useful when faster, but also worse, solutions are wished. Remember that, it is necessary to choose a simulated annealing strategy for the hybrid methods. Case 8 is the fastest case and although the returned number of vertex guards is worse, it is still acceptable. So, this case is the most appropriated to be used in the hybrid methods.

#### Improvements

After the accomplishment of this study, it was found that the runtime of the best case (Case 1) could be improved. Remember that, when a worse neighbour $S_j$ of a solution $S_i$ is generated, it is necessary to check if the new generated solution is valid (see subsection 4.2.3). For that, it is verified if $\bigcup_{v \in S_i \setminus \{v_t\}} Vis(v, P) = P$, where $t \in \{0, 1, \ldots, n-1\}$ is a randomly generated number. Besides in the step where the redundant guards are detected and deleted it necessary to check if $\bigcup_{v \in G \setminus \{v_i\}} Vis(v, P) = P$, for each $v_i \in G$ (see Algorithm 4.1). Hence, it seems that the running time of simulated annealing strategy could be improved if a *dominance matrix* is computed in the pre-processing step. Next, this concept will be defined.

**Definition 4.3** *Let $P$ be a polygon with $n$ vertices and $v_i, v_j \in V_P$. The vertex $v_i$ **dominates** the vertex $v_j$ (or, $v_j$ is dominated by $v_i$) if $Vis(v_j, P) \subset Vis(v_i, P)$.*

**Definition 4.4** *Let $P$ be a polygon with $n$ vertices. The **dominance matrix of** $P$ is a $n \times n$ matrix $A$, where $\forall i, j \in \{0, \ldots, n-1\}$, $A[i, j] = 1$, if the vertex $v_i \in V_P$ dominates the vertex $v_j \in V_P$; and $A[i, j] = 0$, otherwise.*

The dominance matrix could improve the runtime of the above described algorithms because each time that it necessary to see if $G \setminus \{v_i\}$, with $v_i \in G$, is still a vertex guarding set, being $G$ a vertex guarding set: first, it is checked if $v_i$ is dominated by some other vertex of $G$, that is, if $A[i, j] = 1$, for some $v_j \in G$. If so, then it is known that $G \setminus \{v_i\}$ is a vertex guarding set, and it is not necessary to determine if $\bigcup_{v \in G\setminus\{v_i\}} Vis(v, P) = P$ (what is much time consuming).

In this way, the calculation of $A$ was included in the pre-processing step and new results were obtained with Case 1, which was elected the best case. Table 5.9 shows the results obtained with Case 1, with and without the dominance matrix.

| $n$ | Case 1 (without dominance matrix) | | | | Case 1 (with dominance matrix) | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.12 | 5.40 | 16.40 | 4798.60 | 1.27 | 5.35 | 13.85 | 4733.00 |
| 50 | 0.42 | 8.55 | 51.27 | 6550.90 | 3.30 | 8.47 | 41.75 | 6205.20 |
| 70 | 0.65 | 11.80 | 108.75 | 7718.50 | 6.47 | 11.80 | 92.70 | 7874.90 |
| 100 | 1.90 | 16.97 | 243.52 | 10162.00 | 12.97 | 16.52 | 200.32 | 9942.00 |

Table 4.9: Results obtained with SA Case 1, with and without the use of the dominance matrix (arbitrary polygons).

Clearly, we can see that the runtime improves when the dominance matrix is employed. Although the pre-processing time increases, the overall time (pre-processing time plus runtime) is improved when the dominance matrix is employed. Given that, Case 1 with the calculation of the dominance matrix in the pre-processing step was selected to be method $M_2$.

#### 4.4.1.2 Analysis of the GA Parameters

According to section 4.2.4, there are various choices for two of the GA parameters: the selection and the crossover operators. The different combinations produce eight cases (see Table 4.10).

| **Cases** | |
|---|---|
| **Case 1** | Roulette Wheel Selection and Single Point Crossover |
| **Case 2** | Roulette Wheel Selection and Two-Point Crossover |
| **Case 3** | Roulette Wheel Selection and Single Uniform Crossover |
| **Case 4** | Roulette Wheel Selection and Variant of Single Point Crossover |
| **Case 5** | Tournament Selection and Single Point Crossover |
| **Case 6** | Tournament Selection and Two-Point Crossover |
| **Case 7** | Tournament Selection and Single Uniform Crossover |
| **Case 8** | Tournament Selection and Variant of Single Point Crossover |

Table 4.10: Studied cases for GA.

The eight cases were analyzed by comparing the number of vertex guards, the runtime and the number of iterations. In Tables 4.11, 4.12, 4.13, and 4.14 are exposed the results obtained with the first four methods. Note that, the tables show the average time of pre-processing in seconds (*PP*), the average number of vertex guards ($|G|$), the average runtime in seconds (*Time*) and the average number of iterations of the algorithm (*Iterations*).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.02 | 5.27 | 19.22 | 740.02 |
| 50 | 0.45 | 8.40 | 72.30 | 1134.30 |
| 70 | 0.65 | 11.72 | 190.05 | 1763.20 |
| 100 | 1.87 | 17.00 | 504.52 | 2690.30 |

Table 4.11: Results obtained with GA Case 1 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.02 | 5.35 | 18.87 | 709.75 |
| 50 | 0.42 | 8.40 | 72.17 | 1130.40 |
| 70 | 0.80 | 11.67 | 177.05 | 1632.40 |
| 100 | 1.90 | 16.92 | 468.55 | 2532.55 |

Table 4.12: Results obtained with GA Case 2 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.075 | 5.275 | 17.625 | 685.225 |
| 50 | 0.175 | 8.275 | 68.825 | 1098.900 |
| 70 | 0.750 | 11.575 | 171.300 | 1641.600 |
| 100 | 1.775 | 16.900 | 460.450 | 2604.400 |

Table 4.13: Results obtained with GA Case 3 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.15 | 5.22 | 10.70 | 721.20 |
| 50 | 0.37 | 8.25 | 60.35 | 1145.00 |
| 70 | 0.72 | 11.75 | 158.50 | 1632.20 |
| 100 | 1.85 | 16.77 | 443.95 | 2529.50 |

Table 4.14: Results obtained with GA Case 4 (arbitrary polygons).

As we can see, in these first four methods there are almost no differences on the average number of vertex guards. So, the different types of crossover seem to have not influence on the obtained solutions (number of vertex guards) when the roulette wheel selection is used. Concerning the average runtime, Case 4 seems to be the best one. Thus the variant of the single point seems to improve the algorithm runtime.

In the following four cases, it is analyzed how the different types of crossover behave, when the tournament selection is used. The obtained results are shown in Tables 4.15, 4.16, 4.17, and 4.19.

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.02 | 5.35 | 17.32 | 686.20 |
| 50 | 0.27 | 8.50 | 60.90 | 1009.70 |
| 70 | 0.67 | 12.00 | 142.97 | 1399.60 |
| 100 | 1.75 | 16.85 | 384.77 | 2173.10 |

Table 4.15: Results obtained with GA Case 5 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.17 | 5.35 | 16.80 | 682.87 |
| 50 | 0.22 | 8.45 | 59.30 | 973.17 |
| 70 | 0.72 | 11.85 | 138.52 | 1355.80 |
| 100 | 1.82 | 17.00 | 365.60 | 1033.40 |

Table 4.16: Results obtained with GA Case 6 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.07 | 5.30 | 16.72 | 680.52 |
| 50 | 0.40 | 8.47 | 57.40 | 958.10 |
| 70 | 0.82 | 11.77 | 130.12 | 1296.80 |
| 100 | 1.90 | 16.82 | 321.32 | 1822.50 |

Table 4.17: Results obtained with GA Case 7 (arbitrary polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.02 | 5.20 | 9.05 | 709.42 |
| 50 | 0.25 | 8.37 | 44.92 | 985.80 |
| 70 | 0.67 | 11.62 | 110.97 | 1324.70 |
| 100 | 1.80 | 16.90 | 310.50 | 1987.70 |

Table 4.18: Results obtained with GA Case 8 (arbitrary polygons).

Again, in these four methods there are almost no differences on the average number of vertex guards. However, Case 8 seems to be the method that obtains slightly better solutions, with the exception of $n = 100$. Concerning the average runtime, Case 8, also, seems to be the best one. Once more, the crossover method seems to have no influence on the obtained solutions, however it seems that the variant of the single point seems to improve the algorithm runtime.

Comparing the eight methods, we notice that the obtained results are approximately the same for all methods (concerning $|G|$). Concerning the average runtime, Case 8 seems to be the best method. So it does not matter which selection and the crossover operators used, concerning the obtained solutions. But, it seems that the variant of the single point improves the algorithm runtime.

A statistical study was performed. To compare the average number of vertex guards it was used the non-parametric Kruskall-Wallis tests because the Kolmogorov-Smirnof tests showed that the data obtained with Case 1 were always non-normally distributed ($p$-value$<0.001<0.05$, for $n = 30, 50, 70$ and 100). As expected, the Kruskal-Wallis tests showed that there were no significant differences among the eight cases regarding $|G|$, for $n = 30$, 50, 70 and 100 ($p$-value $= 0.9676 \geq 0.05$, for $n = 30$, $p$-value $= 0.9665 \geq 0.05$, for $n = 50$, $p$-value $= 0.9191 \geq 0.05$, for $n = 70$ and $p$-value $= 0.9933 \geq 0.05$, for $n = 100$).

According to the previous conclusion, a statistical analysis was made regarding the runtime. Since the data obtained with Case 1 were always non-normally distributed ($p$-value$<0.001 < 0.05$, for $n = 30, 50, 70$ and 100), the Kruskal-Wallis tests were used again. These tests established that at least one case is significantly different, concerning the runtime ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). So, multiple comparison tests were performed to determine which pairs of methods were significantly different, and which were not. The results provided by these tests allowed to conclude that: for $n = 30$, Case 8 is the fastest case with no significant difference from Case 4 and it is significantly faster than the other cases; for $n = 50$, Case 8 is significantly faster than all the other cases; for $n = 70$, Case 8 is the fastest case with no significant difference from Case 7 and significantly better than the other cases; and finally, for $n = 100$, Case 8 is the fastest case with no significant difference from Cases 6 and 7 and significantly faster than the other cases. According to these results, Case 8 was considered the best case of all.

Remember that, there are other parameters that can be changed in the GAs meta-heuristic, for instance, the genetic operator mutation (see section 4.2.4). It was decided to experiment two different mutation operations in the selected case, Case 8, to see how the algorithm behaves.

Up to now the mutation operation that was applied was to flip all the gene values from zero to one or vice versa, with a probability of 5% ($p_m = 0.05$), as described in section 4.2.4. Now, two different mutations are going to be tested:

- The first one consists in randomly selecting one gene and then flip its value from zero to one or vice versa, with a probability of 5%. This case is designated by Case 8.1.
- The second one consists in applying the mutation described in section 4.2.4, but with a probability of 5%. That is, the mutation described in section 4.2.4 will not occur always, it will only occur with a probability of 5%. For that, it is generated a randomly real number $U \in [0, 1]$, following an uniform distribution, and then if $U \leq 0.05$ the mutation happen; otherwise it does not happens. This case is designated by Case 8.2.

Table 4.19 shows the results obtained with Case 8, Case 8.1 and Case 8.2.

| $n$ | Case 8 | | | | Case 8.1 | | | | Case 8.2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.02 | 5.20 | 9.05 | 709.42 | $> 0.001$ | 5.45 | 4.47 | 652.65 | 0.12 | 5.42 | 3.45 | 618.62 |
| 50 | 0.25 | 8.37 | 44.92 | 985.80 | 0.25 | 8.62 | 17.92 | 812.30 | 0.32 | 8.52 | 17.97 | 758.12 |
| 70 | 0.67 | 11.62 | 110.97 | 1324.70 | 0.67 | 11.92 | 40.50 | 982.32 | 0.67 | 11.82 | 42.02 | 918.25 |
| 100 | 1.80 | 16.90 | 310.50 | 1987.70 | 1.92 | 17.05 | 117.12 | 1362.80 | 1.77 | 17.12 | 121.65 | 1257.70 |

Table 4.19: Results obtained with GA Cases 8, 8.1 and 8.2 (arbitrary polygons).

As we can see, it seems that there are almost no differences on the average number of vertex guards. However, concerning the average runtime, Case 8 seems to be the worst one.

A statistical study was performed. To compare the average number of vertex guards it was used the non-parametric Kruskall-Wallis tests, because the Kolmogorov-Smirnof tests showed that the data obtained with Case 8 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). As expected, the Kruskal-Wallis tests showed that there were no significant differences among the three cases regarding $|G|$, for $n = 30, 50, 70$ and 100 ($p$-value $= 0.4088 \geq 0.05$, for $n = 30$, $p$-value $= 0.5028 \geq 0.05$, for $n = 50$, $p$-value $= 0.6494 \geq 0.05$, for $n = 70$ and $p$-value $= 0.8084 \geq 0.05$, for $n = 100$).

According to the previous conclusion, a statistical analysis was made regarding the runtime. Since the data obtained with Case 8 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100), the Kruskal-Wallis tests were used again. These tests established that at least one case is significantly different, concerning the runtime ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). So, multiple comparison tests were performed to determine which pairs of methods were significantly different, and which were not. The results provided by these tests allowed to conclude that: for $n = 30, 50, 70$ and 100, Case 8 is significantly slower than Case 8.1 and 8.2 and a significant difference was not found between the runtime of Cases 8.1 and 8.2. According to these results, Cases 8.1 and 8.2 are not significantly different. Case 8.2 was selected as the case to be used.

**Improvements**

As for the analysis of the SA parameters, it was found that the runtime of Case 8.2 could be improved if the dominance matrix was used. Here, it is possible to use the dominance matrix in the generation of the initial population and in the step where the redundant guards are detected and removed. Consequently, the calculation dominance matrix of $P$ was included in the pre-processing step and new results were obtained with Case 8.2. Table 4.20 show the results obtained with Case 8.2, with and without the dominance matrix.

| $n$ | Case 8.2 (without dominance matrix) | | | | Case 8.2 (with dominance matrix) | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 30 | 0.12 | 5.42 | 3.45 | 618.62 | 1.25 | 5.35 | 3.92 | 611.72 |
| 50 | 0.32 | 8.52 | 17.97 | 758.12 | 3.35 | 8.62 | 16.20 | 735.27 |
| 70 | 0.67 | 11.82 | 42.02 | 918.25 | 6.30 | 11.77 | 40.55 | 916.10 |
| 100 | 1.77 | 17.12 | 121.65 | 1257.70 | 12.77 | 16.90 | 122.50 | 1311.30 |

Table 4.20: Results obtained with GA Case 8.2, with and without the use of the dominance matrix (arbitrary polygons).

As we can see, when the dominance matrix is employed the runtime seems to be slightly better for $n = 30, 70$ and 100. However, the overall time (pre-processing time plus runtime) is not improved. Given this, Case 8.2, without the calculation of the dominance matrix in the pre-processing step, was selected to be the method $M_3$.

Remember that it is necessary to choose a genetic algorithm for the hybrid methods, it was Case 8.2, without the calculation of the dominance matrix in the pre-processing step, i.e, method $M_3$, that was chosen to be used in the hybrid methods.

#### 4.4.1.3 Comparison of the five strategies

This section proceeds with the analysis and evaluation of the results obtained with the five approximation methods: $M_1$, greedy strategy; $M_2$, SA strategy; $M_3$, GA strategy and the hybrid strategies which are going to be denoted by $M_4$ (the strategy in which the initial population of a GA is generated by a SA algorithm) and $M_5$ (the strategy in which a SA strategy is a genetic operator). Remember that, for the hybrid strategies is necessary to choose a SA strategy and a GA strategy. As stated above, the selected SA strategy was the SA Case 8, i.e., method $M_2$, and the the selected GA strategy was the GA Case 8.2, , i.e., method $M_3$. Experiments were made with the methods $M_4$ and $M_5$, with and without the calculation of the dominance matrix on the pre-processing step and it was concluded that using the dominance matrix these methods were faster. So all the results related to the hybrid strategies where obtained using the dominance matrix.

The methods $M_1$, $M_2$, $M_3$, $M_4$ and $M_5$ were applied over four sets of arbitrary polygons, each one with 40 polygons with 50, 100, 150 and 200 vertex polygons. The obtained results are tabulated in Tables 4.21 and 4.22.

| $n$ | $M_1$ | | | | $M_2$ | | | | $M_3$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 50 | 0.30 | 8.32 | 0.62 | 19.97 | 3.30 | 8.47 | 41.75 | 6205.20 | 0.32 | 8.52 | 17.97 | 758.12 |
| 100 | 1.95 | 16.82 | 3.30 | 44.32 | 12.97 | 16.52 | 200.32 | 9942.00 | 1.77 | 17.12 | 121.65 | 1257.70 |
| 150 | 5.27 | 23.95 | 7.70 | 68.75 | 28.20 | 24.60 | 480.47 | 1401.00 | 5.20 | 25.20 | 388.52 | 2057.40 |
| 200 | 11.75 | 32.45 | 14.87 | 93.72 | 51.35 | 33.25 | 899.85 | 1698.20 | 11.47 | 33.10 | 917.92 | 3075.80 |

Table 4.21: Results obtained with $M_1$, $M_2$ and $M_3$ (arbitrary polygons).

Comparing the results obtained with the non-hybrid methods ($M_1$, $M_2$, $M_3$), concerning the average of $|G|$ (see Table 4.21), we can notice that: for 50-vertex polygons the results are almost equal; for 100-vertex polygons methods $M_1$ and $M_2$ seem to be the best strategies; and for 150 and 200-vertex polygons method $M_1$ seems to be the best strategy. We can also see that, in any case, $M_1$ is much faster than the other methods.

| $n$ | $M_4$ | | | | $M_5$ | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 50 | 3.32 | 7.52 | 46.97 | 506.47 | 3.225 | 7.57 | 32.80 | 581.45 |
| 100 | 12.65 | 15.12 | 461.00 | 538.00 | 12.47 | 15.20 | 252.67 | 759.57 |
| 150 | 28.32 | 22.45 | 1648.00 | 635.17 | 28.35 | 22.75 | 677.22 | 858.35 |
| 200 | 51.55 | 29.85 | 4332.90 | 672.05 | 51.75 | 30.35 | 1448.80 | 944.25 |

Table 4.22: Results obtained with $M_4$ and $M_5$ (arbitrary polygons).

Contrasting, now, the results obtained using the hybrid methods (see Table 4.22) we can see that $M_4$ is slower but the average number of vertex guards seems to be slightly better for $n = 200$. Now, comparing the results achieved with the hybrid methods ($M_4$ and $M_5$) with the results obtained with the non-hybrid strategies ($M_1$, $M_2$ and $M_3$), we can observe that the non-hybrid strategies are, in most cases, faster. Nevertheless, they seem to obtain worse solutions. Summing up, regarding the average number of vertex guards, the methods $M_4$ and

$M_5$ seem to be the best ones and the methods $M_2$ and $M_3$ seem to be the worst approximation techniques (see Figure 4.18).

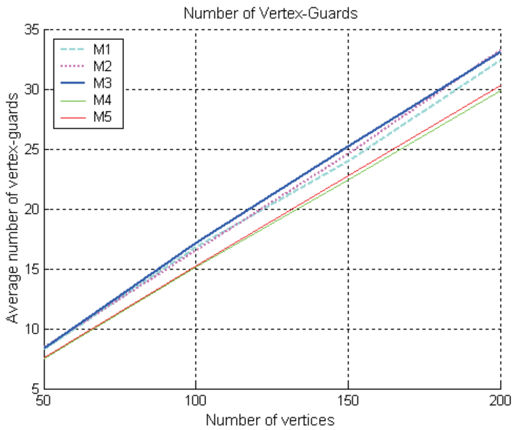



Figure 4.18: Solutions obtained with strategies $M_1, M_2, M_3$, $M_4$ and $M_5$ (arbitrary polygons).

As usual, a statistical study was carried out. The data obtained with the method $M_1$ were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 50, 100, 150$ and 200). The $p$-values returned by the Kruskal-Wallis tests were less than 0.001 for the data obtained with the polygons with $n = 50, 100, 150$ and 200, respectively (note that, all $p$-values are less than 0.05). Then multiple comparison tests were performed. to determine which pairs of averages were significantly different, and which were not. The answers provided by these tests allowed to conclude that (see Figure 4.19):

- for $n = 50, 100$ and $n = 200$, the best method is $M_4$, with no significant differences from $M_5$; and the worst method is $M_2$, with no significat differences from methods $M_1$ and $M_3$.
- for $n = 150$, the best method is $M_4$, with no significant differences from $M_5$; and the worst method is $M_3$, with no significant differences from methods $M_1$ and $M_2$.

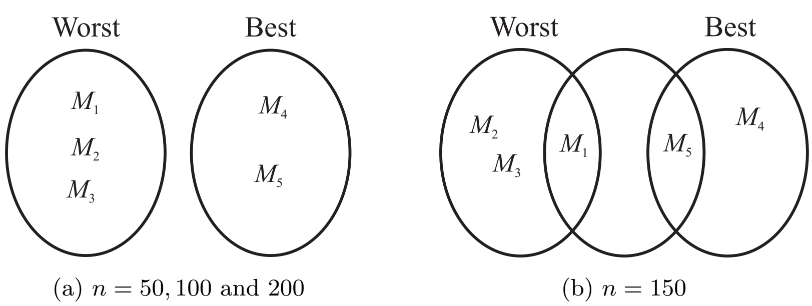



Figure 4.19: Multiple comparison tests of the five methods (arbitrary polygons).

Hence, regarding the hybrid methods, we note that $M_4$ is always significantly better than all non-hybrid methods and that there are no significant differences between these two methods, for $n = 50, 100, 150$ and 200. The statistical study continued regarding the runtime. This study allowed to conclude that $M_5$ is significantly faster than $M_4$, for $n = 50, 100, 150$ and 200. Since a compromise between the quality of the solution and the algorithm runtime is wanted, the study continued considering $M_5$ as the best strategy.

Now, to infer about the average of the minimum number of vertex guards needed to cover an arbitrary polygon, $M_5$ was applied to eight sets of arbitrary polygons, each one with 40 polygons with 30, 50, 70, 100, 110, 130, 150 and 200 vertex polygons. The average of the obtained results, concerning $|G|$, are shown in Table 4.23.

| $n$ | 30 | 50 | 70 | 100 | 110 | 130 | 150 | 200 |
|---|---|---|---|---|---|---|---|---|
| $\|G\|$ | 4.82 | 7.57 | 10.67 | 15.20 | 16.67 | 19.55 | 22.75 | 30.35 |

Table 4.23: Average of the minimum number of vertex guards (arbitrary polygons).

Then, using the least squares method, the following linear adjustment was obtained, with a correlation factor of 0.9997 (see Figure 4.20):

$$f(x) = 0.1505x + 0.1465 \approx \frac{x}{6.64} + 0.1465 \approx \frac{x}{6.64}.$$

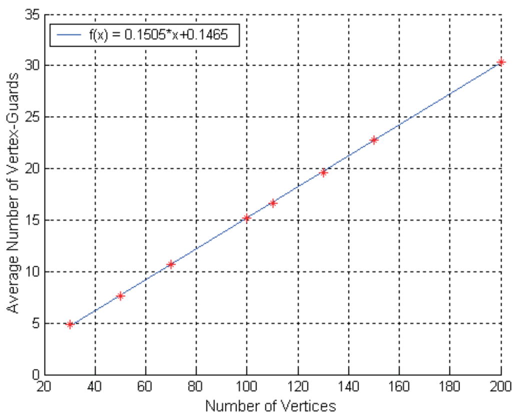
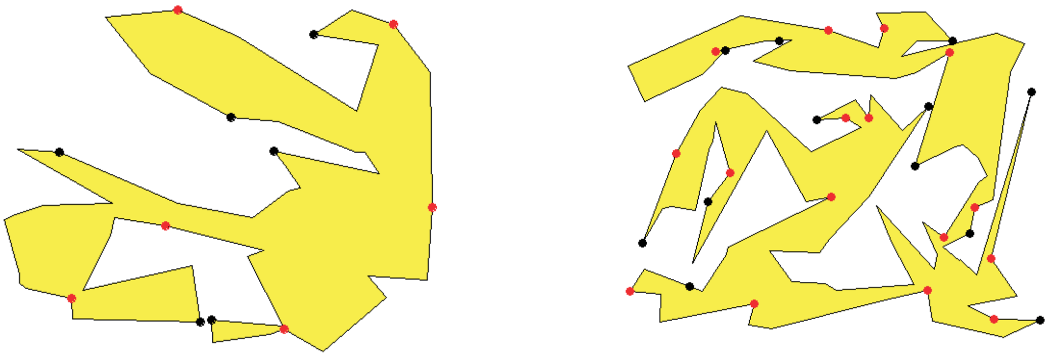



Figure 4.20: Least Squares Method (arbitrary polygons).

So, it can be concluded that on average, and approximately, the minimum number of vertex guards needed to cover an arbitrary polygon with $n$ vertices was observed to be $\lceil \frac{n}{6.64} \rceil$. In order to get a quantitative measure on the quality of the calculated $|G|$ , the visibility-independent sets were computed for our instances (the eight sets of polygons described above). The ratio between the smaller $G$ (obtained with $M_5$) and the larger visibility-independent set, $IS$ obtained with $A_2$ (see section 4.3) never exceeded 1.66, with an average of 1.21 for the universe of 320 polygons. That implies that algorithm $M_5$ has an approximation ratio less than or equal to 1.66.

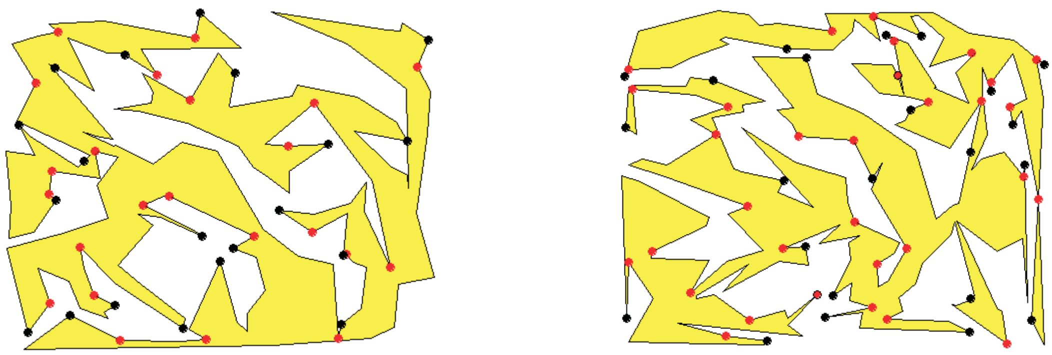
Figures 4.21 and 4.22 shows snapshots obtained with our software. In these figures are illustrated four arbitrary polygons for which the visibility-independent sets *IS* were obtained with $A_2$ and the solutions $G$ were obtained with $M_5$.



(a) $|IS| = 6$ and $|G| = 6$

(b) $|IS| = 12$ and $|G| = 16$

Figure 4.21: *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on arbitrary polygons with: (a) $n = 50$ ; (b) $n = 100$.



(a) $|IS| = 20$ and $|G| = 23$

(b) $|IS| = 25$ and $|G| = 32$

Figure 4.22: *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on arbitrary polygons with: (a) $n = 150$ and (b) $n = 200$.

### 4.4.2 Orthogonal Polygons

In this subsection, like in subsection 4.4.1, it will be analyzed the nine and the eight cases, resulting from the choice of the SA and GA parameters (see Tables 4.1 and 4.10, respectively), to select the ones that best fit on the MVGS($P$) problem for orthogonal polygons. After that, it will be studied the solutions obtained with the five developed strategies.

#### 4.4.2.1 Analysis of the SA Parameters

Table 4.24 shows the results obtained with the first three cases. Similar to arbitrary polygons, the best solution appears to correspond to a slow decrease in temperature (FSA) with a larger number of iterations and a higher response time, i.e., the best solution in these first three cases seems to be obtained by Case 1.

| $n$ | Case 1 (FSA dec.) | | | | Case 2 (VFSA dec.) | | | | Case 3 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.10 | 4.77 | 11.22 | 4745.40 | 0.10 | 5.47 | 1.125 | 9.00 | 0.17 | 5.80 | 5.20 | 83.00 |
| 50 | 0.57 | 8.17 | 38.65 | 6392.80 | 0.40 | 8.77 | 3.65 | 10.00 | 0.25 | 9.10 | 17.95 | 88.00 |
| 70 | 1.05 | 10.72 | 82.47 | 7912.20 | 1.15 | 12.07 | 8.52 | 10.00 | 1.05 | 12.92 | 40.97 | 91.00 |
| 100 | 2.55 | 15.97 | 190.65 | 10290.00 | 2.57 | 17.25 | 19.17 | 11.00 | 2.57 | 17.85 | 90.85 | 94.00 |

Table 4.24: Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$) on orthogonal polygons.

In the following three cases it is going to be analyzed how the different types of temperature decreasing behave, being $T_0$ constant ($T_0 = 500$).

| $n$ | Case 4 (FSA dec.) | | | | Case 5 (VFSA dec.) | | | | Case 6 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.10 | 4.475 | 160.45 | 39092.00 | 0.12 | 5.72 | 13.72 | 12.00 | 0.12 | 5.47 | 84.85 | 110.00 |
| 50 | 0.30 | 7.575 | 338.70 | 38408.00 | 0.55 | 9.07 | 27.70 | 12.00 | 0.52 | 9.12 | 171.10 | 110.00 |
| 70 | 1.05 | 9.95 | 552.75 | 39335.00 | 1.02 | 12.05 | 46.95 | 12.00 | 1.02 | 12.55 | 282.70 | 110.00 |
| 100 | 2.62 | 14.77 | 883.45 | 39397.00 | 2.62 | 17.65 | 74.07 | 12.00 | 2.47 | 17.90 | 438.70 | 110.00 |

Table 4.25: Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$) on orthogonal polygons.

As for arbitrary polygons, the best solution in these three cases seems to be obtained by Case 4. Comparing these last three cases with the first three ones for the same type of temperature decrease, that is, Case 1 with Case 4, Case 2 with Case 5 and Case 3 with Case 6, we can see that the solutions provided by Case 4 seems to be better than the solutions provided by Case 1, although in Case 4 the algorithm runtime and the number of iterations also increase. Concerning Cases 2 and 5, it appears that the obtained solutions are almost equal, except for $n = 50$, where the Case 2 appears to obtain better solutions. Finally, Cases 3 and 6 it seems that they obtain similar solutions, being Case 6 slower. Therefore, in general, if a solution nearer to the optimal one is searched it seems that it is more suitable to choose a slow temperature decrease (FSA) and $T_0 = 500$; for faster temperature decreases (VFSA and Geometric) the election of the initial temperature does not turn out to be influential.

Similar to arbitrary polygons, it would be expected that a geometrical decrease should produce better solutions than a rapid decrease (VFSA), what does not happens. Again, the reason for this behaviour is the elimination of redundant vertex guards, so that the solutions have not considerable differences.

In the following cases it is analyzed how the three temperature decreases behave if $T_0 = \frac{n}{4}$.

| $n$ | Case 7 (FSA dec.) | | | | Case 8 (VFSA dec.) | | | | Case 9 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.10 | 5.250 | 3.37 | 1399.00 | 0.17 | 4.85 | 0.37 | 8.00 | 0.10 | 5.57 | 1.525 | 69.00 |
| 50 | 0.55 | 8.62 | 12.35 | 2399.00 | 0.35 | 8.22 | 1.75 | 8.00 | 0.25 | 9.15 | 5.150 | 74.00 |
| 70 | 1.12 | 12.20 | 30.42 | 3399.00 | 1.05 | 11.00 | 3.82 | 9.00 | 1.12 | 12.75 | 11.62 | 78.00 |
| 100 | 2.55 | 16.62 | 65.65 | 4548.60 | 2.57 | 16.15 | 8.85 | 9.00 | 2.52 | 17.95 | 26.72 | 81.00 |

Table 4.26: Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$) on orthogonal polygons.

As for arbitrary polygons, on observing these last three cases we verify that for an initial temperature $T_0 = \frac{n}{4}$ the solutions seem to improve when the decrease of the temperature is fast (VFSA). Such as for arbitrary polygons, the reason for this behaviour is the removal of redundant vertex guards. Nevertheless, this improvement cannot be observed if $T_0 = 500$ and the decrease of the temperature is slow (Case 4), that seems to be the best case of the first six cases.

As always, a statistical study was carried out. The analysis of the data normality showed that the data obtained with Case 1 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100) . The $p$-values returned by the Kruskal-Wallis tests were less then 0.001 for the data obtained with the polygons with $n = 30, 50, 70$ and 100, respectively (note that, all $p$-values are less than 0.05). Then multiple comparison tests were performed to determine which pairs of averages were significantly different, and which were not. The answers provided by these tests are presented in Tables 4.27, 4.28, 4.29 and 4.30. The sign "+" indicates that the sample data (concerning $|G|$) is significantly different and the sign "-" indicates otherwise.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | - | + | - | + | - | - | - | + |
| Case 2 | - | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | + | - |
| Case 4 | - | + | + | • | + | + | + | - | + |
| Case 5 | + | - | - | + | • | - | - | + | - |
| Case 6 | - | - | - | + | - | • | - | - | - |
| Case 7 | - | - | - | + | - | - | • | - | - |
| Case 8 | - | - | + | - | + | - | - | • | + |
| Case 9 | + | - | - | + | - | - | - | + | • |

Table 4.27: Multiple comparison tests, of SA Cases, for 30-vertex orthogonal polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | - | + | - | - | + | - | - | + |
| Case 2 | - | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | - | + |
| Case 5 | - | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | - | - | - | + | - | - | • | - | - |
| Case 8 | - | - | - | - | - | + | - | • | - |
| Case 9 | + | - | - | + | - | - | - | - | • |

Table 4.28: Multiple comparison tests, of SA Cases, for 50-vertex orthogonal polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | + | - | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | + | - |
| Case 4 | - | + | + | • | + | + | + | - | + |
| Case 5 | + | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | + | - | - | + | - | - | • | + | - |
| Case 8 | - | - | + | - | - | + | + | • | + |
| Case 9 | + | - | - | + | - | - | - | + | • |

Table 4.29: Multiple comparison tests, of SA Cases, for 70-vertex orthogonal polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | - | - | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | + | - |
| Case 4 | - | + | + | • | + | + | + | - | + |
| Case 5 | + | - | - | + | • | - | - | + | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | - | - | - | + | - | - | • | - | + |
| Case 8 | - | - | + | - | + | + | - | • | + |
| Case 9 | + | - | - | + | - | - | + | + | • |

Table 4.30: Multiple comparison tests, of SA Cases, for 100-vertex orthogonal polygons.

The multiple comparison tests, also, allowed to conclude that:

- for $n = 30$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with no significant differences from Case 2 and significantly better than Case 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.23 (a));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.24);
  - Cases 7, 8 and 9. The best is Case 8 with no significant differences from Case 7

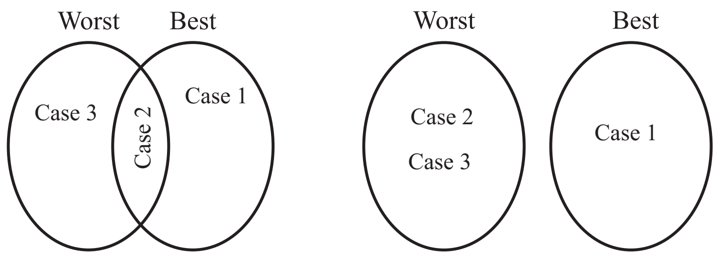
and significantly better than 9; the worst is Case 9 with no significant differences from Case 7 (see Figure 4.25 (a));

- the nine cases. The best is Case 4, with no significant differences from Cases 1 and 8; the worst is Case 3, with no significant differences from Cases 2, 5, 6, 7 and 9.

- for $n = 50$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Case 2 and significantly better than Case 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.23 (a));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.24);
  - Cases 7, 8 and 9. The best is Case 8 with no significant differences from Cases 7 and 9; the worst is Case 9 with no significant differences from Cases 7 and 8 (see Figure 4.25 (b));
  - the nine cases. The best is Case 4, with no significant differences from Case 1 and Case 8; the worst is Case 6, with no significant differences from Cases 2, 3, 5, 7 and 9.
- for $n = 70$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.23 (b));
  - Cases 4, 5 and 6. Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.24);
  - Cases 7, 8 and 9. The best is Case 8 with significant differences from Cases 7 and 9; the worst is Case 9 with no significant differences from Case 7 (see Figure 4.25 (c));
  - the nine cases. The best is Case 4, with no significant differences from Cases 1 and 8; the worst is Case 3, with no significant differences from Cases 2, 5, 6, 7 and 9.
- for $n = 100$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 4.23 (b));

- Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 4.24);
- Cases 7, 8 and 9. The best is Case 8 with no significant differences from Cases 7 and significantly better then Case 9; the worst is Case 9 with significant differences from Case 7 and 8 (see Figure 4.25 (d));
- the nine cases. The best is Case 4, with no significant differences from Case 1 and 8; the worst is Case 9, with no significant differences from Cases 2, 3, 5, and 6.

As it can be noticed, using the multiple comparison tests, for $T_0 = n$ the best solutions are obtained when the temperature decrease is slow (FSA) with no significant differences with a fast temperature decrease (VFSA), for $n = 30$ and $n = 50$ (see Figure 4.23).




(a) $n = 30$ and 50 (b) $n = 70$ and 100

Figure 4.23: Multiple comparison tests, of SA Cases 1, 2 and 3 (orthogonal polygons).

For $T_0 = 500$ the best solutions always correspond to a slow temperature decrease (see Figure 4.24).

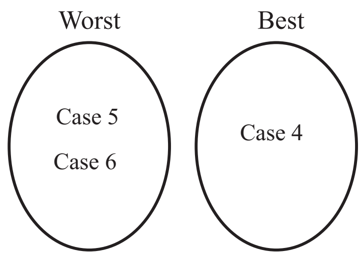



Figure 4.24: Multiple comparison tests, of SA Cases 4, 5 and 6, for $n = 30, 50, 70$ and 100 (orthogonal polygons).

Finally, for $T_0 = \frac{n}{4}$, the best solutions are obtained when the temperature decrease is fast (VFSA), with no significant differences with a slow temperature decrease (FSA), except for $n = 70$ (see Figure 4.25).

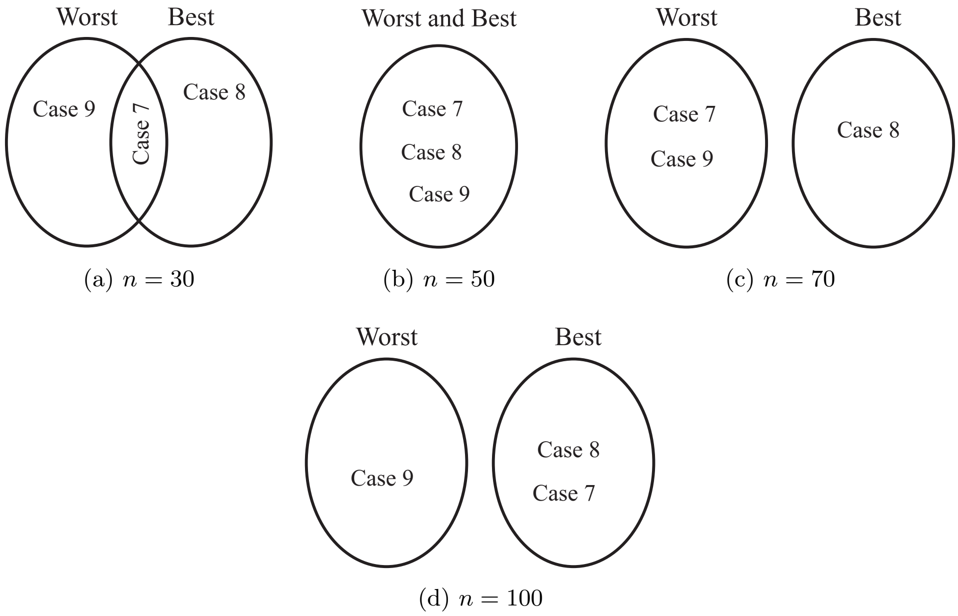



Figure 4.25: Multiple comparison tests, of SA Cases 7, 8 and 9 (orthogonal polygons).

Notice that, for all types of initial temperature $T_0$, it would be expected that a geometrical decrease should produce better solutions than a rapid decrease (VFSA), what does not happen. We can see that, for $T_0 = n$ and $T_0 = 500$, the obtained solutions have not significant differences. However, for $T_0 = \frac{n}{4}$ the solutions obtained with a rapid decrease is significantly better than the solutions obtained with a geometric decrease (except for $n = 50$). As stated before, the reason for this behaviour is the elimination of redundant vertex guards.

Comparing these results with the ones obtained with arbitrary polygons, we can notice that the great difference is for $T_0 = \frac{n}{4}$. For orthogonal polygons, Case 8 (fast temperature decrease) is always significantly better than Case 9 (geometric temperature decrease), except for $n = 50$. So, for $T_0 = \frac{n}{4}$, the best solutions are always obtained with a fast temperature decrease with no significant differences with a geometric decrease, except for $n = 70$. So, surprisingly, a slow decrease does not correspond to the best solutions.

Concerning the nine cases, if the temperature decrease is slow, the best solutions are obtained with $T_0 = 500$ and $T_0 = n$. If the temperature decrease is fast the initial temperature does not have influence, except for $n = 30$ and 100, where the obtained solutions by Case 8 ($T_0 = \frac{n}{4}$) are significantly worst than Case 5 ($T_0 = n$). If the temperature decrease is geometric the initial temperature does not have influence. It can also be concluded that the best solutions are obtained with Case 4 for $n = 30$ 50, 70 and 100 and a significant difference was not found between the number of vertex guards obtained with this case and Cases 1 e 8. Despite our observing in Tables 4.2, 4.3 and 4.4, that Case 4 seems to outperform Cases 1 and 8 in relation to the average of $|G|$, for $n = 50, 70$ and 100. So, the statistical analysis proceeded regarding the runtime. This analysis was made in a similar way and it allowed to conclude that Case 8 is significantly faster than Cases 1 and 4, for $n = 30, 50, 70$ and 100. Given this, for orthogonal polygons, Case 8 was elected the best case and it was used to generate the initial population and also used as a genetic operator in the hybrid strategies.

After this study, it was concluded that the runtime of the best case (Case 8) was good enough, so it was not necessary to improve it. Hence, Case 8 was selected to be the method $M_2$.

#### 4.4.2.2 Analysis of the GA Parameters

The results obtained by the eight methods are shown in Tables 4.31, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37 and 4.38.

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.15 | 4.72 | 14.95 | 746.02 |
| 50 | 0.50 | 7.95 | 59.75 | 1150.90 |
| 70 | 1.10 | 10.67 | 162.00 | 1789.50 |
| 100 | 2.55 | 15.47 | 421.80 | 2738.50 |

Table 4.31: Results obtained with GA Case 1 (orthogonal polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.17 | 4.62 | 15.25 | 749.22 |
| 50 | 0.50 | 7.87 | 59.10 | 1151.20 |
| 70 | 1.00 | 10.85 | 157.57 | 1712.50 |
| 100 | 2.50 | 15.50 | 401.87 | 2601.80 |

Table 4.32: Results obtained with GA Case 2 (orthogonal polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.05 | 4.62 | 14.12 | 724.42 |
| 50 | 0.45 | 7.77 | 54.80 | 1102.70 |
| 70 | 1.02 | 10.32 | 149.87 | 1729.80 |
| 100 | 2.50 | 14.92 | 389.62 | 2745.40 |

Table 4.33: Results obtained with GA Case 3 (orthogonal polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.05 | 4.57 | 9.52 | 735.20 |
| 50 | 0.45 | 7.77 | 50.25 | 1102.70 |
| 70 | 1.05 | 10.60 | 145.67 | 1733.80 |
| 100 | 2.62 | 15.35 | 387.42 | 2673.10 |

Table 4.34: Results obtained with GA Case 4 (orthogonal polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.12 | 4.70 | 14.05 | 712.85 |
| 50 | 0.45 | 7.90 | 50.05 | 1035.00 |
| 70 | 1.07 | 10.72 | 124.45 | 1399.80 |
| 100 | 2.55 | 15.70 | 312.40 | 2058.20 |

Table 4.35: Results obtained with GA Case 5 (orthogonal polygons).

| $n$ | PP (sec.) | $\|G\|$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.20 | 4.75 | 14.40 | 740.17 |
| 50 | 0.42 | 7.90 | 48.72 | 985.70 |
| 70 | 1.07 | 10.72 | 122.07 | 1394.20 |
| 100 | 2.52 | 15.55 | 296.85 | 1964.00 |

Table 4.36: Results obtained with GA Case 6 (orthogonal polygons).

| $n$ | PP (sec.) | $\lvert G\rvert$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.12 | 4.70 | 12.80 | 681.42 |
| 50 | 0.37 | 7.90 | 44.97 | 932.65 |
| 70 | 1.17 | 10.85 | 110.70 | 1268.50 |
| 100 | 2.55 | 15.02 | 285.80 | 1994.40 |

Table 4.37: Results obtained with GA Case 7 (orthogonal polygons).

| $n$ | PP (sec.) | $\lvert G\rvert$ | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.10 | 4.75 | 7.10 | 684.47 |
| 50 | 0.37 | 7.82 | 37.52 | 995.62 |
| 70 | 1.12 | 10.67 | 99.47 | 1339.80 |
| 100 | 2.50 | 15.37 | 265.62 | 1973.40 |

Table 4.38: Results obtained with GA Case 8 (orthogonal polygons).

As for arbitrary polygons, comparing the eight methods, we notice that the obtained results, concerning the average of $|G|$ , are approximately the same for all cases. Concerning the average runtime, Case 8 seems to be the faster one.

To compare the average number of vertex guards it was used the non-parametric Kruskall-Wallis tests, because the Kolmogorov-Smirnof tests showed that the data obtained with Case 1 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). As expected, and as in the case of arbitrary polygons, the Kruskal-Wallis tests showed that there was no significant differences in the eight cases regarding $|G|$, for $n = 30, 50, 70$ and 100 ($p$-value $= 0.9407 \geq 0.05$, for $n = 30$, $p$-value $= 0.9873 \geq 0.05$, for $n = 50$, $p$-value $= 0.6175 \geq 0.05$, for $n = 70$ and $p$-value $= 0.2108 \geq 0.05$, for $n = 100$).

According to the previous conclusion, a statistical analysis was made regarding the runtime. Since the data obtained with Case 1 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100), the Kruskal-Wallis tests were used again. These tests established that at least one case is significantly different concerning the runtime ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). So, multiple comparison tests were performed. The results provided by these tests allowed to conclude that: for $n = 30$, Case 8 is the fastest case with no significant difference from Case 4 and it is significantly faster than the other cases; for $n = 50$ and 70, Case 8 is the fastest case with no significant difference from Case 7 and significantly better than the other cases; and, finally, for $n = 100$, Case 8 is the fastest case with no significant difference from Cases 6 and 7 and it is significantly faster than the other cases. According to these results it was concluded, as for arbitrary polygons, that Case 8 is the best case.

Similar to arbitrary polygons, it was decided to try two different mutation operations on the elected case, Case 8, to see how the algorithm behaves. The mutation operation, that was applied up to now, was to flip all the gene value from zero to one or vice versa, with a probability of 5% ($p_m = 0.05$), as described in section 4.2.4. Now, the two different mutations proposed in subsection 4.4.1.2 are going to be tested:

- The first one consists in randomly selecting one gene and then flipping its value from zero to one or vice versa, with a probability of 5%. This case is designated by Case 8.1.

- The second one consists in applying the mutation described in section 4.2.4, but with a probability of 5%. That is, the mutation described in section 4.2.4 will not always occur, it will only occur with a probability of 5%. This case is designated by Case 8.2.

Table 4.39 tabulates the results obtained with Case 8, Case 8.1 and Case 8.2.

| $n$ | Case 8 | | | | Case 8.1 | | | | Case 8.2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. | PP | $\|G\|$ | Time | Iter. |
| 30 | 0.10 | 4.75 | 7.10 | 684.47 | 0.07 | 4.82 | 2.90 | 608.32 | 0.100 | 4.82 | 3.15 | 622.65 |
| 50 | 0.37 | 7.82 | 37.52 | 995.62 | 0.37 | 7.97 | 12.55 | 721.25 | 0.450 | 7.82 | 12.80 | 730.60 |
| 70 | 1.12 | 10.67 | 99.47 | 1339.80 | 0.97 | 10.75 | 35.35 | 947.87 | 1.075 | 10.70 | 36.25 | 896.12 |
| 100 | 2.50 | 15.37 | 265.62 | 1973.40 | 2.45 | 15.50 | 107.77 | 1543.10 | 2.675 | 15.27 | 109.07 | 1366.00 |

Table 4.39: Results obtained with GA Cases 8, 8.1 and 8.2 (orthogonal polygons).

As we can see, it seems that there are almost no differences on the average number of vertex guards. Concerning the average runtime, Case 8 seems to be the worst case.

As usually, a the statistical study was performed. To compare the average number of vertex guards the non-parametric Kruskall-Wallis tests were used, because the Kolmogorov-Smirnof tests showed that the data obtained with Case 8 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). As expected, the Kruskal-Wallis tests showed that there were no significant differences among the three cases regarding $|G|$, for $n = 30, 50, 70$ and 100 ($p$-value $= 0.9002 \geq 0.05$, for $n = 30$, $p$-value $= 0.5460 \geq 0.05$, for $n = 50$, $p$-value $= 0.8892 \geq 0.05$, for $n = 70$ and $p$-value $= 0.7538 \geq 0.05$, for $n = 100$). Accordingly, a statistical analysis was made regarding the runtime. Since the data obtained with Case 8 were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100), the Kruskal-Wallis tests were used again. These tests established that at least one case is significantly different, concerning the runtime ($p$-value $< 0.001 < 0.05$, for $n = 30, 50, 70$ and 100). So, multiple comparison tests were performed to determine which pairs of methods were significantly different, and which were not. The results provided by these tests allowed to conclude that: for $n = 30, 50, 70$ and 100, Case 8 is significantly slower than Case 8.1 and 8.2 and a significant difference was not found between the runtime of Cases 8.1 and 8.2. According to these results, Cases 8.1 and 8.2 are not significantly different. Case 8.2 was considered the best case.

**Improvements**

Similar to the analysis of the SA parameters, it was found that the runtime of Case 8.2 could be improved if the dominance matrix was used. Consequently, the calculation dominance matrix was included in the pre-processing step and new results were obtained with Case 8.2. Table 4.40 shows the results obtained with Case 8.2, with and without the dominance matrix.

| $n$ | Case 8.2 (without dominance matrix) | | | | Case 8.2 (with dominance matrix) | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 30 | 0.10 | 4.82 | 3.15 | 622.65 | 1.12 | 4.90 | 2.25 | 596.72 |
| 50 | 0.45 | 7.82 | 12.80 | 730.60 | 3.10 | 7.82 | 12.75 | 745.55 |
| 70 | 1.07 | 10.70 | 36.25 | 896.12 | 6.52 | 10.57 | 37.87 | 968.17 |
| 100 | 2.67 | 15.27 | 109.07 | 1366.00 | 13.15 | 15.25 | 102.87 | 1325.50 |

Table 4.40: Results obtained with GA Case 8.2, with and without the use of the dominance matrix (orthogonal polygons).

As we can see, when the dominance matrix is employed the runtime seems to be slightly better for $n = 30, 70$ and 100. However, the overall time (pre-processing time plus runtime) is not improved when the dominance matrix is employed. Given this, Case 8.2, without the calculation of the dominance matrix in the pre-processing step, was selected to be the method $M_3$.

Remember that it is necessary to choose a genetic algorithm for the hybrid methods. It was also Case 8.2, without the calculation of the dominance matrix, that was chosen to be used in the hybrid methods.

#### 4.4.2.3 Comparison of the five strategies

In this section it is going to be analyzed and evaluated the results obtained with the five approximation methods: $M_1$, greedy strategy; $M_2$, SA strategy; $M_3$, GA strategy and the hybrid strategies, which are going to be denoted by $M_4$ (the strategy in which the initial population of a GA is generated by a SA algorithm) and $M_5$ (the strategy in which a SA strategy is a genetic operator). Remember that, for the hybrid strategies, $M_4$ and $M_5$, it was necessary to choose a SA strategy and a GA strategy. As stated before, the selected SA strategy is the method $M_2$ and the method $M_3$, respectively. Experiments were made with the methods $M_4$ and $M_5$, with and without the calculation of the dominance matrix in the pre-processing step, it was concluded that using the dominance matrix these methods were faster. So all the results related to the hybrid strategies were obtained by using the dominance matrix. Tables 4.41 and 4.42 present the results obtained with the five methods.

Comparing the solutions obtained with the non-hybrid methods (see Table 4.41) we can notice that: the method $M_2$ appears to be the method with which the obtained solutions are the worst (especially for $n = 50$ and 100) and the method $M_1$ seems to be slightly better than $M_3$ for $n = 150$ and 200. We can also see that, for all $n$, $M_1$ is much faster than the other methods.

| $n$ | $M_1$ | | | | $M_2$ | | | | $M_3$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 50 | 0.42 | 7.72 | 0.60 | 19.42 | 0.35 | 8.22 | 1.75 | 8.00 | 0.45 | 7.82 | 12.80 | 730.60 |
| 100 | 2.42 | 15.22 | 3.20 | 45.20 | 2.57 | 16.15 | 8.85 | 9.00 | 2.67 | 15.27 | 109.07 | 1366.00 |
| 150 | 6.77 | 22.25 | 8.02 | 69.67 | 6.87 | 23.72 | 23.72 | 9.00 | 6.92 | 23.15 | 384.50 | 2235.30 |
| 200 | 14.60 | 29.40 | 15.62 | 94.72 | 14.55 | 30.87 | 46.30 | 10.00 | 14.50 | 30.30 | 857.07 | 2916.60 |

Table 4.41: Results obtained with $M_1$, $M_2$ and $M_3$ (orthogonal polygons).

| $n$ | $M_4$ | | | | $M_5$ | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\|G\|$ | Time | Iterations | PP | $\|G\|$ | Time | Iterations |
| 50 | 3.22 | 7.27 | 39.62 | 506.60 | 3.25 | 7.35 | 26.90 | 568.12 |
| 100 | 13.62 | 13.90 | 405.72 | 555.40 | 13.40 | 14.05 | 200.57 | 704.77 |
| 150 | 31.27 | 20.65 | 1574.80 | 600.75 | 31.20 | 21.07 | 584.90 | 817.65 |
| 200 | 58.27 | 27.17 | 4115.30 | 667.82 | 58.57 | 27.80 | 1339.20 | 894.87 |

Table 4.42: Results obtained with $M_4$ and $M_5$ (orthogonal polygons).

Contrasting, now, the results obtained using the hybrid methods (see Table 4.42) we can see that $M_4$ is slower but the average number of vertex guards seems to be slightly better, especially for $n = 100$ and 150. Now, comparing the results obtained with the hybrid methods ($M_4$ and $M_5$) with the results obtained with the non-hybrid strategies ($M_1$, $M_2$ and $M_3$), we can observe that the non-hybrid strategies are much faster. Nevertheless, they seem to obtain worse solutions (except for $n = 50$, where they are almost equal). As for arbitrary polygons, concerning the average number of vertex guards, the methods $M_4$ and $M_5$ seem to be the best ones and the method $M_2$ appear to be the worst approximation technique (see Figure 4.26).

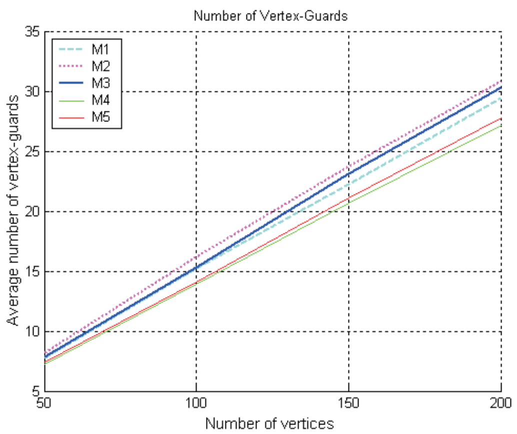



Figure 4.26: Solutions obtained with strategies $M_1, M_2, M_3$, $M_4$ and $M_5$ (orthogonal polygons).

As usual, a statistical study was carried out. The data obtained with the method $M_1$

were always non-normally distributed ($p$-value $< 0.001 < 0.05$, for $n = 50, 100, 150$ and 200). The $p$-values returned by the Kruskal-Wallis tests were less than 0.001 for $n = 50, 100, 150$ and 200, respectively (note that, all $p$-values are less than 0.05). Then multiple comparison tests were performed. These tests allowed to conclude that (see Figure 4.27):

- for $n = 50$, the best method is $M_4$, with no significant differences from $M_1, M_3$ and $M_5$; and the worst method is $M_2$, with no significant differences from methods $M_1$ and $M_3$;
- for $n = 100$, the best method is $M_4$, with no significant differences from $M_5$; and the worst method is $M_2$, with no significant differences from methods $M_1$ and $M_3$;
- for $n = 150$, the best method is $M_4$, with no significant differences from $M_5$; and the worst method is $M_2$, with no significant differences from method $M_3$;
- for $n = 200$, the best method is $M_4$, with no significant differences from $M_5$; and the worst method is $M_2$, with no significant differences from methods $M_1$ and $M_3$.

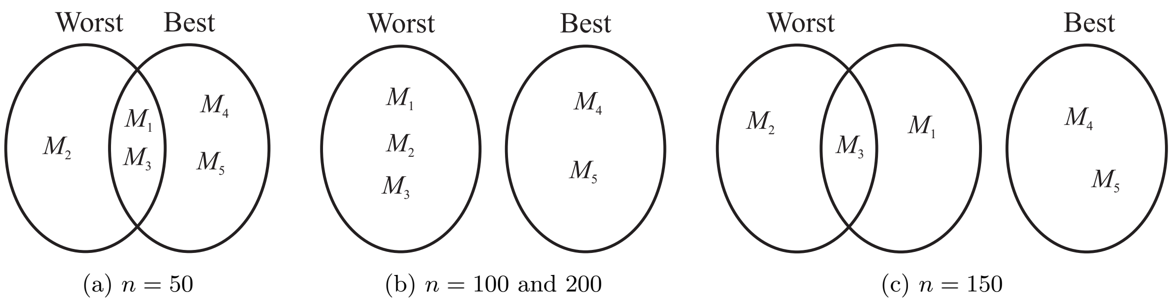



Figure 4.27: Multiple comparison tests of the five methods (orthogonal polygons).

Regarding the hybrid methods, we note that $M_4$ is always significantly better than all non-hybrid methods, for $n = 100, 150$ and 200, and that there are no significant differences between these two methods, for $n = 50, 100, 150$ and 200. The statistical study continued regarding the runtime. This study allowed to conclude that $M_5$ is significantly faster than $M_4$, for $n = 50, 100, 150$ and 200. As it is desired a compromise between the quality of the solution and the algorithm runtime, the study proceeded considering $M_5$ as being our best strategy.

Now, to infer about the average of the minimum number of vertex guards needed to cover an arbitrary polygon, we applied $M_5$ to eight sets of arbitrary polygons, each one with 40 polygons with 30, 50, 70, 100, 110, 130, 150 and 200 vertex polygons. The average of the obtained results, concerning $|G|$, are shown in Table 4.43.

| $n$ | 30 | 50 | 70 | 100 | 110 | 130 | 150 | 200 |
|---|---|---|---|---|---|---|---|---|
| $\lvert G\rvert$ | 4.42 | 7.35 | 9.77 | 14.05 | 15.07 | 17.07 | 21.07 | 27.80 |

Table 4.43: Average of the minimum number of vertex guards (orthogonal polygons).

Then, using the least squares method, the following linear adjustment was obtained, with a correlation factor of 0.9994 (see Figure 4.28):

$$f(x) = 0.1371x + 0.2739 \approx \frac{x}{7.29} + 0.2739 \approx \frac{x}{7.29}.$$

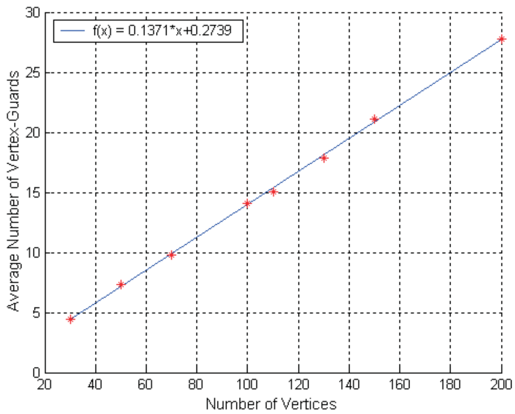
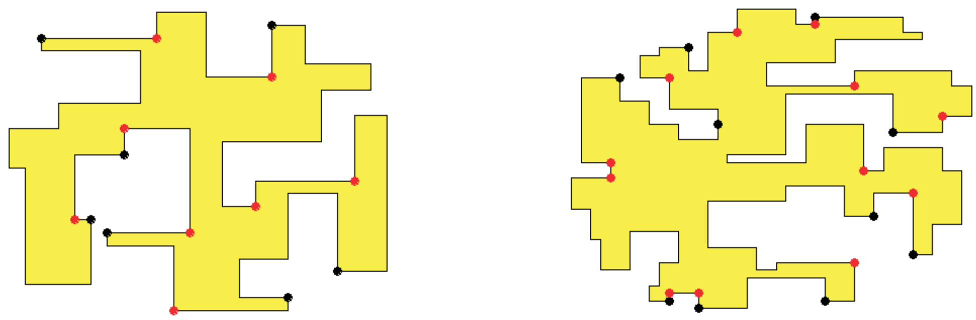



Figure 4.28: Least Squares Method (orthogonal polygons).

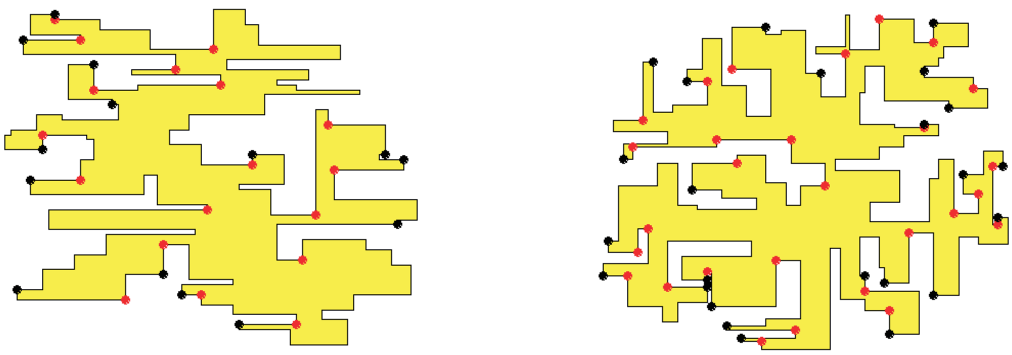
Hence, it can be concluded that on average, and approximately, the minimum number of vertex guards needed to cover an arbitrary polygon with $n$ vertices is $\lceil \frac{n}{7.29} \rceil$. In order to get a quantitative measure on the quality of the calculated $|G|$ , the visibility-independent sets were computed for our instances (the eight sets of polygons described above). The ratio between the smallest $G$ (obtained with $M_5$ ) and the largest visibility-independent set, $IS$ obtained with $A_2$ (see section 4.2.3), never exceeded 1.80 (with an average of 1.28 for the universe of 320 polygons). This implies that algorithm $M_5$ has an approximation ratio less than or equal to 1.80.

Figures 4.29 and 4.30 shows snapshots obtained with our software. In these figures it is illustrated four orthogonal polygons for which the visibility-independent sets $IS$ were obtained with $A_2$ and the solutions $G$ were obtained with $M_5$.



(a) $|IS| = 7$ and $|G| = 8$ (b) $|IS| = 10$ and $|G| = 12$

Figure 4.29: *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on orthogonal polygons with: (a) $n = 50$ ; (b) $n = 100$.



(a) $|IS| = 14$ and $|G| = 18$ (b) $|IS| = 24$ and $|G| = 28$

Figure 4.30: *IS* and $G$ sets (represented by black and red dots, respectively) obtained with $A_2$ and $M_5$ on orthogonal polygons with: (a) $n = 150$ and (b) $n = 200$.

## 4.5 Concluding Remarks

In this chapter it was proposed approximation algorithms that allow to obtain a vertex guarding set $G$, whose cardinality approximates the minimal number of vertex guards needed to guard a given polygon $P$. In other words, approximation algorithms were designed and implemented to tackle the MINIMUM VERTEX GUARD SET problem on polygons. Five approximation strategies were studied: one greedy, $M_1$; one based on the simulated annealing metaheuristic, $M_2$; one based on genetic algorithms metaheuristic, $M_3$; and two hybrid metaheuristics, $M_4$ and $M_5$. It was, also, developed a greedy algorithm to compute visibility-independent sets, allowing to obtain provable bounds on how close our results are to the optimal.

Using a large set of randomly generated polygons (arbitrary and orthogonal), an experimental comparative study was made on the suitability of the developed methods which allowed to conclude that:

(1) Concerning the SA strategy, for arbitrary polygons, the best case was observed to be Case 1, that is, $T_0 = n$ and a slow temperature decrease (FSA). And, unexpectedly, for orthogonal polygons, the best case was observed to be Case 8, that is, $T_0 = \frac{n}{4}$ and a fast temperature decrease (VFSA). The only reasonable justification for this behaviour is the removal of redundant vertex guards as the final step of the approximation strategy.

(2) Regarding the GA strategy, both for arbitrary and orthogonal polygons, it was observed that the strategies implemented with the different studied operators (selection, crossover and mutation) do not show differences as to the obtained solutions. However, it was observed that the strategy that uses the tournament selection, the variant of the single point crossover and the mutation operator, which occurs with a probability of 5% flipping the value of each gene with a probability of 5%, was the fastest one. So, this strategy was chosen to be the best one (designated by method $M_3$).

(3) About the hybrid strategies, both for arbitrary and orthogonal polygons, it was observed that they do not obtain different solutions. But, the hybrid strategy where SA is applied after the crossover operator in a GA and which was designated by method $M_5$, is faster than the other one, in which the initial population of a GA is generated by SA (designated by method $M_4$). It was also saw that both hybridizations obtain better solutions than the "pure" GA, i.e., method $M_3$.

(4) Finally and as to the five approximation strategies, the best one was method $M_5$. The computational experiments also allowed to conclude that, on average and approximately, the minimal number of vertex guards needed to cover an arbitrary and an orthogonal polygon was observed to be $\lceil \frac{n}{6.64} \rceil$ and $\lceil \frac{n}{7.29} \rceil$, respectively. These values are much less than the theoretical bounds $\lfloor \frac{n}{3} \rfloor$ and $\lfloor \frac{n}{4} \rfloor$, respectively. To end and in terms of quality of the solutions, it was also conclude that the approximation ratio is less than or equal to 1.66 and less than or equal to 1.80, for arbitrary and orthogonal polygons, respectively.

It is important to point out that all alternatives, with respect to the parameters of the SA, GA and hybrid metaheuristics to be explored are almost "infinite". In this work was attempted to find references for these parameters, noting that a more exhaustive study in future investigations might improve the obtained results.

As a conclusion, the hybrid metaheuristics, especially the strategy $M_5$, proved to behave well in solving the MINIMUM VERTEX GUARD SET problem. This way and as future work, it should be studied more hybridizations which will permit to improve the obtained solutions as well as the algorithms' runtime

# Chapter 5

# Minimum Vertex Floodlight Set Problem

In the previous chapter, Chapter 4, where the MINIMUM VERTEX GUARD SET problem was studied, it was assumed that the guards could see around them in all directions, that is, the guards have a $2\pi$ range visibility (or equivalently, that the lights sources could emit light in all directions). However, many illumination or guarding devices cannot illuminate or search all around themselves. Floodlights, for example illuminate only a restricted angle of illumination. Therefore, in some cases it is interesting and useful to consider visibility/illumination problems in which the guards have a restricted visibility range (or equivalently, that the light sources have a restricted angle of illumination). In this chapter, the visibility problem is considered on orthogonal polygons, in which the guards have a $\frac{\pi}{2}$ visibility range. The problem is known as Minimum Vertex Floodlight Set problem and denoted by MVFS($P$).

The chapter is divided in five section. The problem is described and formalized in section 5.1. In section 5.2 four approximation algorithms are developed. The first is based on the simulated annealing metaheuristic (subsection 5.2.2), the second is based on the genetic algorithms metaheuristic (subsection 5.2.3) and the last two are hybrid algorithms based on used metaheuristics (subsection 5.2.4). In section 5.3 it is presented a method to determine a lower bound for the unknown optimal solution. This method permits to get the performance ratio of the approximation algorithms. In section 5.4 the experiments made over a large set of randomly generated orthogonal polygons are described. Finally, in section 5.5 some conclusions are presented.

## 5.1 Problem Description

Some necessary definitions will follow.

**Definition 5.1** *A* $\boldsymbol{\alpha_i}$***-floodlight*** $f_i$ *is a source of light with a restricted angle of illumination*

$0 < \alpha_i < 2\pi$. *A $\alpha_i$-floodlight is also called* ***$\alpha_i$-guard****, that is a static guard with a $\alpha_i$ range visibility.*

**Definition 5.2** *Being $P$ a polygon, a $\alpha_i$-floodlight placed on a vertex of $P$ is called* ***vertex $\alpha_i$-floodlight*** *or* ***vertex $\alpha_i$-guard****.*

**Definition 5.3** *A $\frac{\pi}{2}$-floodlight is called* ***orthogonal floodlight****.*

As stated above, in this chapter it is going to be considered the illumination (covering) of orthogonal polygons with vertex $\frac{\pi}{2}$-floodlights (vertex $\frac{\pi}{2}$-guards). That is, the determination of a set of vertex $\frac{\pi}{2}$-floodlights ($\frac{\pi}{2}$-guards) that completely illuminate (cover) an orthogonal polygon $P$. Since this work only deals with orthogonal floodlights, and to simplicity, the term *"floodlight"* is used instead of *"orthogonal floodlights"*.

**Definition 5.4** *Let $P$ be an orthogonal polygon. A given set $F$ of floodlights placed on the vertices of $P$ is a* ***vertex floodlighting set*** *for $P$ if they cover $P$, i.e., if $\bigcup_{f \in F} Vis(f,P) = P$. A vertex floodlighting set for $P$ is denoted by $F$ and its cardinality by $|F|$.*

Urrutia [129] proved that:

**Proposition 5.1** $\lfloor \frac{3n-4}{8} \rfloor$ *vertex floodlights are occasionally necessary and always sufficient to illuminate an orthogonal polygon $P$ with $n$ vertices.*

The proof of this proposition employs a set of four different rules for the placement of the floodlights. First of all, the edges of an orthogonal polygon are classified into four types: *top*, *left*, *bottom* and *right*. An horizontal edge $e$ is said to be a top edge if the interior of the polygon is immediately below $e$; otherwise (the interior of the polygon is immediately above $e$) $e$ is said to be a bottom edge. A vertical edge $e$ is said to be a left edge if the interior of the polygon is immediately on right of $e$; otherwise (the interior of the polygon is immediately on left of $e$) $e$ is said to be a right edge (see Figure 5.1). Then it is defined the top-left rule illumination as follows:

(i) at the top vertex of every left edge $e$ of $P$ it is placed a floodlight aligned with $e$, that is a floodlight that illuminates the angular sector $\frac{3\pi}{2}$ to $2\pi$;

(ii) at the left vertex of every top edge $e$ of $P$ it is placed a floodlight aligned with $e$, that is a floodlight that illuminates the angular sector $\frac{3\pi}{2}$ to $2\pi$.

Finally, it is proved that the floodlights placed according to the top-left rule illuminate the polygon $P$ (see Figure 5.1).

In a similar way the top-right, bottom-left and bottom-right illumination rules are defined, each of which illuminates $P$. To prove the sufficiency of the $\lfloor \frac{3n-4}{8} \rfloor$ vertex floodlights it is used the placement of the floodlights by the four rules. Figure 5.2 shows the vertex floodlights placed by the four rules.

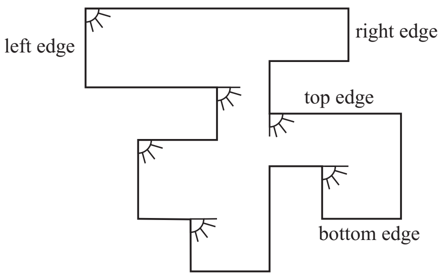



Figure 5.1: Illuminating an orthogonal polygon with the top-left illumination rule.

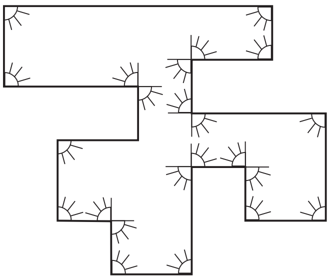


Figure 5.2: Illuminating an orthogonal polygon with the four illumination rules.

So, in the proof of this proposition it is assumed that the vertex floodlights have a restricted orientation: they are edge-aligned. It is also assumed that each reflex vertex has at most two vertex floodlights and, obviously, each convex vertex has at most one vertex floodlight.

Note that, according to the stated above, four different types of vertex floodlights can be defined (see Figure 5.3):

1. TL-floodlight, which is edge-aligned and illuminates the angular sector $\frac{3\pi}{2}$ to $2\pi$;
2. TR-floodlight, which is edge-aligned and illuminates the angular sector $\pi$ to $\frac{3\pi}{2}$;
3. BL-floodlight, which is edge-aligned and illuminates the angular sector 0 to $\frac{\pi}{2}$;
4. BR-floodlight, which is edge-aligned and illuminates the angular sector $\frac{\pi}{2}$ to $\frac{3\pi}{2}$.


Figure 5.3: Vertex floodlights: (a) TL-floodlight; (b) TR-floodlight; (c) BL-floodlight and (d) BR-floodlight.

Figure 5.4 shows two polygons that need $\lfloor \frac{3n-4}{8} \rfloor$ vertex floodlights, which prove the necessity of those number of vertex floodlights.

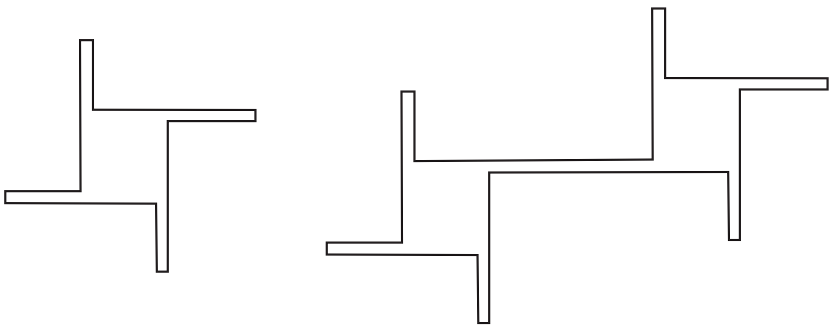


Figure 5.4: Orthogonal polygons that require $\lfloor \frac{3n-4}{8} \rfloor$ floodlights.

But while it is possible to illuminate some polygons with the above established number of vertex floodlights, for many others this number is clearly too large. This reasoning justifies the algorithmic Minimum Vertex Floodlight Set problem, which formally will be denoted by MVFS($P$) and can be stated as follows:

> **MVFS($P$)**
> **Input**: An orthogonal polygon $P$ with $n$ vertices.
> **Question**: What is the minimum number of vertex floodlights necessary to illuminate $P$?

It is strongly believed that the MVFS($P$) problem is $\mathcal{NP}$-hard [129]. Accordingly, in this chapter, will be developed approximation methods to tackle it. These methods will be described in the next section.

## 5.2 Approximation Methods

Let $P$ be an orthogonal polygon with $n$ vertices. As stated in the previous subsection, at each convex vertex of $P$ can be placed at most one vertex floodlight and at each reflex vertex can be placed at most two vertex floodlights. So, the maximum number of orthogonal vertex floodlights that can be placed on $P$ is $n+r$, where $r$ denotes the number of reflex vertices of $P$. Each vertex floodlight that can be placed on $P$ is denoted by $f_j^i$, where $j \in \{0, 1, \ldots, n+r-1\}$ and $i \in \{0, 1, \ldots, n-1\}$. That is, $f_j^i$ is the $j$-esim floodlight and $v_i$ is the vertex where it is placed.

The visibility polygon of a vertex floodlight $f_j^i$, is designated by $Vis(f_j^i, P)$. If $v_i$ is a convex vertex, then $Vis(f_j^i, P) = Vis(v_i, P)$. Otherwise ($v_i$ is a reflex vertex), if the incident edges on $v_i$ are:

1. a bottom and a left edge, then the angular sector from 0 to $\frac{\pi}{2}$ is removed from $Vis(v_i, P)$. In this way, two polygons are obtained, $Vis(f_j^i, P)$ and $Vis(f_{j+1}^i, P)$, that correspond to the visibility polygons of a BR-floodlight and a TL-floodlight, respectively (see Figure 5.5);
2. a bottom and a right edge, then the angular sector from $Vis(v_i, P)$ from $\frac{\pi}{2}$ to $\pi$ is removed from $Vis(v_i, P)$. In this way, two polygons are obtained, $Vis(f_j^i, P)$ and $Vis(f_{j+1}^i, P)$, that correspond to the visibility polygons of a TR-floodlight and a BL-floodlight, respectively;
3. a top and a left edge, then the angular sector from the angular sector from $\frac{3\pi}{2}$ to $2\pi$ is removed from $Vis(v_i, P)$. In this way, two polygons are obtained, $Vis(f_j^i, P)$ and $Vis(f_{j+1}^i, P)$, that correspond to the visibility polygons of a TR-floodlight and a BL-floodlight, respectively;

4. a top and a right edge, then the angular sector from $\pi$ to $\frac{3\pi}{2}$ is removed from $Vis(v_i, P)$. In this way, are obtained two polygons, $Vis(f_j^i, P)$ and $Vis(f_{j+1}^i, P)$, that correspond to the visibility polygons of a TL-floodlight and a BR-floodlight, respectively.

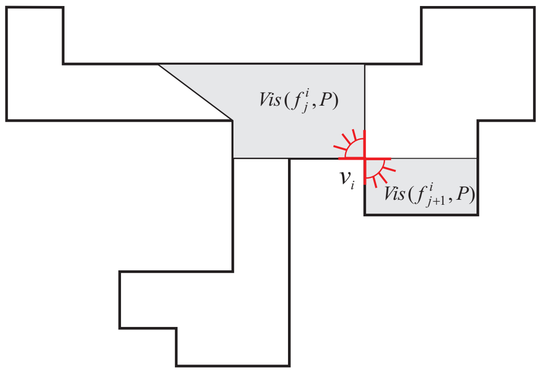



Figure 5.5: Visibility polygons of a BR-floodlight and a TL-floodlight.

Four approximation algorithms were developed to determine a vertex floodlighting set $F$, whose cardinality approximates the minimal number of vertex floodlights needed to cover a given orthogonal polygon $P$. The first is based on the SA metaheuristic, which is called $M_1$; the second is based on the GAs metaheuristic, which is named $M_2$ and the last two are hybrid algorithms, which are designated by $M_3$ and $M_4$.

### 5.2.1 Pre-processing Step

Given an orthogonal polygon $P$ with $n$ vertices ($n$-ogon, for short), the maximum number floodlights that can be placed on the vertices of $P$ is $n + r = \frac{3n-4}{2}$ (remember that $n = 2r + 4$, for all $n$-ogons). Along the approximation algorithms the visibility polygons of these floodlights are needed more than once. Hence, a pre-processing step is performed where the visibility polygons of all possible floodlights are computed and stored. In other words, all $Vis(f_j^i, P)$, for $j = 0, \ldots, \frac{3n-4}{2} - 1$ and $i = 0, 1, \ldots, n-1$, are computed and stored. This information will decrease the algorithms' runtime because each time a floodlight visibility polygon is required it is not necessary to calculate it again.

To compute the visibility polygon of each vertex floodlight $f_j^i$, first $Vis(v_i, P)$ is calculated, using the linear algorithm developed by Lee [85]. Then, if:

(i) $v_i$ is a convex vertex, $Vis(f_j^i, P) = Vis(v_i, P)$;

(ii) otherwise ($v_i$ is a reflex vertex), depending on the type of incident edges on $v_i$, the appropriate angular sector is removed from $Vis(v_i, P)$ and two visibility polygons $Vis(f_j^i, P)$ and $Vis(f_{j+1}^i, P)$ are obtained, which are the visibility polygons of the floodlights that can be placed on $v_i$.

The methods that will be described in the next subsections allow to obtain a vertex floodlighting set $F$. However it may be possible to find a set $U \subset F$ such that $\bigcup_{f \in F \setminus U} Vis(f, P) = P$. So, after the described strategies the redundant floodlights are iteratively removed. This removal is done in a similar way to the removing of redundant vertex guards (see subsection 4.2.2, Algorithm 4.2)

### 5.2.2 Simulated Annealing Strategy

As stated in Chapter 2, subsection 2.1.1, to solve an optimization problem with the SA metaheuristic it is necessary to identify some parameters. These parameters were defined to suit the MVFS($P$) problem and a description of such procedure will follow.

#### 1. Specific Parameters

**Solution space.** The solution space, set $S$, to the MVFS($P$) problem is the set of all vertex floodlighting sets for $P$. Thus, $S$ is a finite set and can be represented by $S = \{S_1, S_2, \ldots, S_m\}$, where $S_l = f^0_{0,l} \cdots f^{n-1}_{n+r-1,l}$, for $l = 1, \ldots, m$. In this way, each element of $S$ is represented by a chain with length $n + r$, where each $f^i_{j,l}$, with $j \in \{0, \ldots, n+r-1\}$ and $i \in \{0, \ldots, n-1\}$, represents the floodlight $f^i_j$ (that is, represents the $j$-esim vertex floodlight and $v_i \in V_P$ is the vertex where it can be placed) and its value is 0 or 1. If $f^i_{j,l} = 1$ then the floodlight $f^i_j$ is placed on vertex $v_i$; otherwise ($f^i_{j,l} = 0$) the floodlight $f^i_j$ is not placed on vertex $v_i$. In Figure 5.6 is presented an example that illustrate these notions.

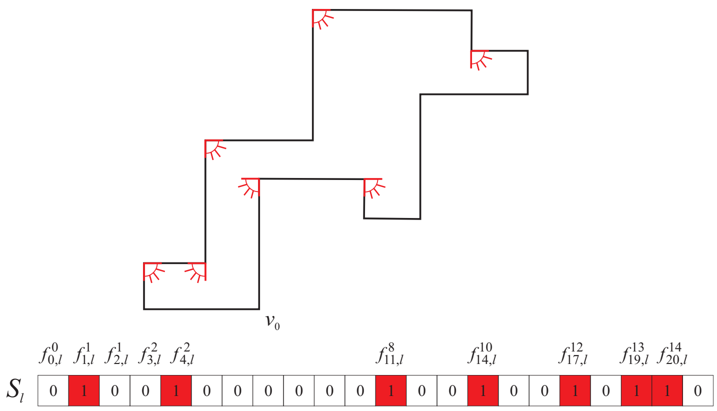



Figure 5.6: An element $S_l \in S$ for a 16-vertex orthogonal and its representation.

**Objective function.** The objective function $f : S \to \mathbb{N}$ assigns to each element of $S$ a natural value. For each $S_l \in S$, $f(S_l)$ is equal to the number of 1's in $S_l$, representing the cardinality of the vertex floodlighting set.

**Neighbourhood of each solution.** For the MVFS($P$) problem the generation of a neighbour $S_w$ of candidate solution $S_l \in S$ is similar to the one made for the MVGS($P$) problem (see 4.2.3). Let $S_l = f^0_{0,l}, \ldots, f^{n-1}_{n+r-1,l}$ be an element of $S$, a natural number $t \in [0, n+r-1]$ is randomly generated (following a uniformly distribution), and then if:

- $f^i_{t,l} = 1$ then $f^i_{t,w}$ is set to 0, i.e., $f^i_{t,w} = 0$. If this new solution is a valid solution, then it is accepted, since the solution was improved; else the obtained solution is rejected.
- $f^i_{t,l} = 0$ then $f^i_{t,w}$ is set to 1, i.e., $f^i_{t,w} = 1$, and this new solution is accepted with a probability, since it is a worse solution.

**Initial Solution.** The initial solution needed to solve the MVFS($P$) problem with the SA strategy is an initial vertex floodlighting set for $P$, that is designated by $S_0$ and it will be the first solution to be analyzed and iterated. In the developed algorithm, to generate this first solution, it was used the top-left rule explained in subsection 5.1. That is, all possible TL-floodlights were placed on the vertices of $P$. Figure 5.7 exemplifies the initial solution on a 16-vertex orthogonal polygon.

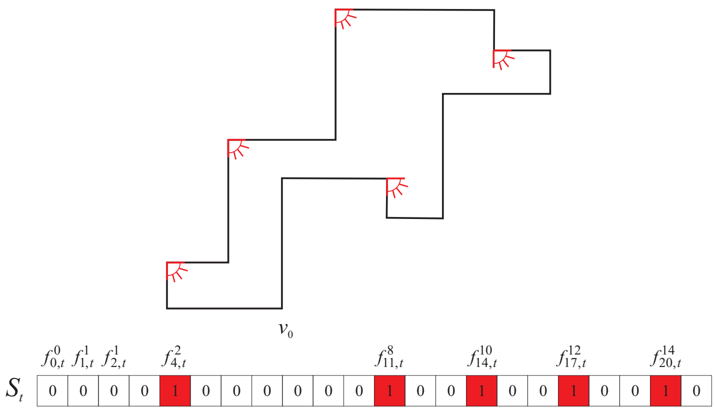



Figure 5.7: Initial Solution.

### 2. Generic Parameters

**Initial temperature ($T_0$).** As for the MHVS($P$) and MVGS($P$) problems (see subsections 3.2.2 and 4.2.3, respectively), a comparative study was performed taking into account two different types of $T_0$:

1. An initial temperature dependent on the number of vertices of the polygon $P$, $T_0 = f(n)$ (in the performed study it was considered $T_0 = n$ and $T_0 = \frac{n}{4}$);
2. A constant initial temperature: $T_0 = 500$.

**Temperature decrement rule.** The value of the temperature at each iteration $k$, $T_k$, is established by a temperature decrement rule. As for the MHVS($P$) and MVGS($P$) problems (see subsections 3.2.2 and 4.2.3, respectively), an analysis was made on three different types of rules:

1. $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease);
2. $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) and
3. $T_{k+1} = \alpha T_k$, where $0 < \alpha < 1$ (Geometric decrease). It was chosen $\alpha = 0.9$.

**Number of iterations at each temperature ($N(T_k)$).** Similar to the MHVS($P$) and MVGS($P$) problems (see subsections 4.2.3 and 3.2.2), here $N(T_k) = \lceil T_k \rceil$.

**Termination condition.** As for the MVGS($P$) problem (see subsection 4.2.3), the termination condition chosen consists in finishing the search when the temperature is less than or equal to 0.005, i.e., $T_f = 0.005$ or when during 3000 consecutive series of temperatures, no new best solution is obtained and the percentage of accepted solutions is less than 2%.

### 5.2.3 Genetic Algorithms Strategy

As stated in Chapter 2, subsection 2.1.1, to solve an optimization problem with the GAs metaheuristic it is necessary to identify some parameters. Below, are described how these parameters were defined to suit the MVFS($P$) problem.

**Encoding.** The genetic representation of the candidate solutions to the MVFS($P$) problem is similar to the representation of each candidate solution, $S_l$, on the SA strategy. An individual $I$ is represented by a chain of 0's and 1's, with length $n+r-1$, i.e., $I = g_0^0 \dots g_{n+r-1}^{n-1}$, where each gene $g_j^i$ represents the vertex floodlight $f_j^i$, $j \in \{0, 1, ..., n+r-1\}$ and $i \in \{0, 1, ..., n-1\}$. The value of each gene is 0 or 1. If $g_j^i = 1$ then the floodlight $f_j^i$ is placed on vertex $v_i$; otherwise ($g_j^i = 0$) the floodlight $f_j^i$ is not placed on vertex $v_i$.

**Initial Population.** In the developed algorithm the population size was chosen to be $\lfloor \frac{3n-4}{8} \rfloor$, which is the number of vertex floodlights sufficient to cover any $n$-vertex orthogonal polygon. In this way, the input of the problem is linked with the elements of the metaheuristic. Thus, the population for the generation $t$ is represented by: $P(t) = \{I_0^t, I_1^t, \dots, I^t_{\lfloor \frac{3n-4}{8} \rfloor - 1}\}$, where each $I_i^t$ represents an individual belonging to the population $P(t)$.

Remember that an individual represents a candidate solution for the MVFS($P$) problem, i.e., each individual must be a vertex floodlighting set. To create the initial population, $P(0)$, each individual $I_i^0$, for $i = 0, \dots, \lfloor \frac{3n-4}{8} \rfloor$, is generated in the following way: all of its genes are set to 1, then a gene is randomly selected and its value is set to 0 if the resultant individual is valid; otherwise its value remains 1. In Figure 5.8 it is illustrated a 20-vertex orthogonal polygon and its initial population, $P(0) = \{I_0^0, I_1^0, \dots, I_6^0\}$.

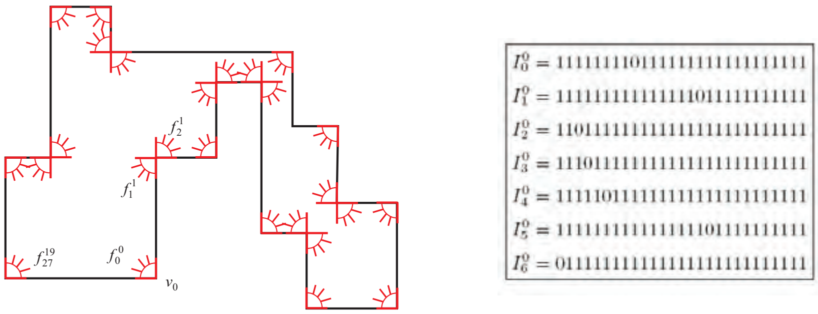



Figure 5.8: On the left a 20-vertex orthogonal polygon $P$ and all possible floodlights; on the right the initial population for $P$.

**Fitness Function.** This function was defined in a similar way to the objective function defined for the SA strategy. For each $I$, $f$ is defined by $f(I) = g_0^0 + \ldots + g_{n+r-1}^{n-1}$, representing the cardinality of the vertex floodlighting set. Such as for the MVGS($P$) problem this function assigns lower values to the solutions closer to the optimal one(s) and the used selection method was adapted in order to reflect this behaviour.

**Selection.** The selection method should choose the best individuals to be reproduced. In the developed algorithm it was used the tournament selection because this method was elected for the MVGS($P$) problem and as to the MVFS($P$) problem, it is assumed that, very likely this method will behave well.

**Crossover.** Similar to the selection method, it was used the crossover that was chosen for the MVGS($P$) problem. Remember that, it was chosen a variant of the single point crossover to generate one child and the crossover only occurs with a given probability $p_c = 0.08$ (see subsection 4.2.4).

**Mutation.** The action of the mutation operation is relatively simple. With a probability of $p_m$ the following happens: the value of each binary gene is flipped from zero to one or vice versa, with a probability of $p_m$ (see Case 8.2 in subsection 4.4.1.2). In the developed strategy, as for the MVGS($P$) problem, the mutation was applied to the child obtained in the crossover operation, with $p_m = 0.05$, and if the obtained individual is not valid it will not be accepted.

**Population Generation.** As for the MVGS($P$) problem, a new population is generated replacing the worst individual by the child obtained at the crossover.

**Population Evaluation.** The evaluation of a population, i.e., the fitness of a population, $F(P(t))$, is considered as the minimum value of the fitness function when applied to all individuals of the population, i.e., $F(P(t)) = \min\{f(I_0^t), \ldots, f(I^t_{\lfloor\frac{3n-4}{8}\rfloor-1})\}$.

**Termination condition.** As always, if in a sufficiently large number of generations the fitness has not changed, it can be assumed that the solution is close to optimal. Thus, for the termination condition it was considered that if the fitness of the population $F(P(t))$ remains unchanged for a number of generations $h$, the search should stop. It was chosen $h = 500$, such as for the MVGS($P$) problem.

In the sequel, the GA strategy is, sometimes, designated by $M_2$.

### 5.2.4 Hybrid Strategies

As for the MVFS($P$), two different hybrid metaheuristics were developed to solve the MVFS($P$). These methods are similar to the ones developed for the MVGS($P$) (see subsection 4.2.5). In the first method, for the initial population of a GA, $\lfloor\frac{3n-4}{8}\rfloor$ individuals are generated, which are obtained by running a SA strategy $\lfloor\frac{3n-4}{8}\rfloor$. In the second a SA strategy is a genetic operator, of a GA, that occurs with a certain probability $p_{sa}$ (in the experimental evaluation was used $p_{sa} = 0.01$).

The first and the second methods allow to observe how a GA behaves including high quality solutions in the initial population and enforcing intensification during the search process, respectively.

## 5.3 Greedy Strategy for floodlight visibility-independent sets

Since the existence of an efficient algorithm to solve MVFS($P$) problem remains open (remember that it is strongly believed that this problem problem is $\mathcal{NP}$-hard), the optimal solution of the MVFS($P$) is unknown. So, as for the previous problems (MHVS($P$) and MVGS($P$)), if one can not compute the optimal value, how can one expect to prove that the output of the approximation algorithms are near it?

Once more, it was conducted an experimental analysis on the performance of the developed algorithms. This analysis is similar to the one that was made for the MVGS($P$) problem (see section 4.3). In this way, it was developed a method to compute a lower bound on the optimal number of vertex floodlights for each instance in the performed experiments.

First, it was considered the *floodlight visibility-independent set* concept.

**Definition 5.5** *Let $P$ be a $n$-vertex polygon. A **floodlight visibility-independent set** is a finite set of points on $P$, $FIS \subset P$, such that $\forall p, q \in FIS$ $p$ and $q$ are not illuminated by the same floodlight. Its elements are called **floodlight visibility-independent points** and its cardinality is denoted by $|FIS|$.*

The example given in Figure 5.9 illustrates a floodlight visibility-independent set of cardinality 3.

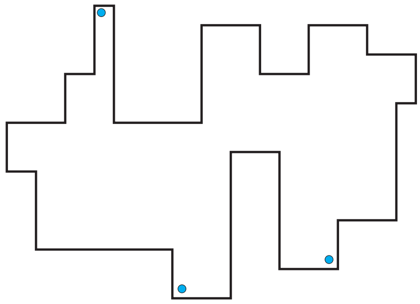


Figure 5.9: Floodlight visibility-independent set of an orthogonal polygon. Blue dots represent visibility-independent points.

By definition, it is easy to verify that no single vertex floodlight is able to illuminate more than one point of *FIS*, consequently $\forall F, \mathit{FIS}, |F| \geq |\mathit{FIS}|$. Easily it can be concluded that the number of points on a maximum-cardinality floodlight visibility-independent set is a lower bound for the optimal number of vertex floodlights on $P$. However, as far as it is known, the existence of an efficient algorithm to determine this lower bound is unknown. Thus, with a similar reasoning to what was done for the MVGS($P$) problem:

Being $F$ and *FIS* approximate solutions for the MVFS($P$) problem and for the problem of determining a floodlight visibility-independent set with maximum cardinality, respectively. If there is a constant $c \in \mathbb{R}^+$ such that $|F| \leq c \times |\mathit{FIS}|$, for any orthogonal polygon $P$, it can be said that the approximation algorithm used to obtain $F$ has an approximation ratio of $c$ [13].

Therefore, it was developed a greedy algorithm to find large floodlight visibility-independent sets, which is designated by $A_1$ (see Algorithm 5.1 for illustration). As usually, it starts with a set of candidates $C$ (not floodlight visibility-independent), then it adds floodlight visibility-independent points one by one to until a solution *FIS* is obtained (*FIS* initially is an empty set), selecting at each step a point from the candidate set $C$, according to some rule. The candidate set used here is equal to the one used in subsection 4.3, which is $C = C_1 \cup C_2$, where $C_1$ and $C_2$ denote the convex vertices and the midpoints of the edges incident on two reflex vertices, respectively. In the developed algorithm, first of all, for each candidate $c \in C$ the number of floodlights that illuminate it is calculated. Then, on each step of the algorithm the candidate that is illuminated by the smaller number of floodlights is selected. After that, all the candidates $c_j$ that are illuminated by the same floodlights that illuminate $c_i$ are removed from $C$. The process stops when the set $C$ is empty.

The application of the the SA strategy, the GA strategy and the hybrid strategies together with $A_1$, to each instance in our experiments, gives provable performance bounds in terms of

**Algorithm 5.1** Computing *FIS* (greedy algorithm $A_1$)

**Input:** An orthogonal polygon $P$ with $n$ vertices

**Output:** A floodlight visibility-independent set, *FIS*

1. $FIS \leftarrow \emptyset$
2. $C \leftarrow C_1 \cup C_2$
3. **for** each $c \in C$ **do**
4. calculate the number of floodlights that illuminate $c$
5. **end for**
6. **while** $C \neq \emptyset$ **do**
7. choose the $c_i \in C$ that is illuminated by the smaller number of floodlights
8. $FIS \leftarrow FIS \cup \{c_i\}$
9. remove $c_i$ from $C$ and all $c_j \in C$ such that they are illuminated by the same floodlights that illuminate $c_i$
10. **end while**
11. **return** *FIS*

approximation ratios. In the performed experiments, given a polygon $P$, the main objective is to find a small vertex floodlighting set $F$ and a large floodlight visibility-independent set *FIS*; the obtained set $F$ approximates the optimal number of vertex floodlights with approximation ratio $\frac{|F|}{|FIS|}$. Note that, if a visibility-independent set *FIS* and a vertex floodlighting set $F$ are found, such that $|FIS| = |F|$, then $F$ is an optimal vertex floodlighting set.

## 5.4 Experiments and Results

In this section the objective is to find which of the approximation methods obtains the best solutions in a reasonable time. In the next subsection will be discussed the results and conclusions resulting from the accomplished experiments on orthogonal polygons.

### 5.4.1 Orthogonal Polygons

To choose the SA parameters that best fit on the MVFS($P$) problem, the experiments were done over four sets of polygons, each formed by 40 polygons of 30, 50, 70 and 100 vertex polygons. To analyze the four methods four sets each one formed by 40 polygons of 50, 100, 150 and 200, were used. To analyze the SA parameters the experiments were performed on polygons with fewer vertices due to the time of execution, which is relatively high for some cases. The performed computational tests showed that to choose these parameters it would be sufficient to do experiments with polygons of up to 100 vertices. The other choices (associated with the dimension of the sets of polygons), not being theoretically justified, were dictated by practical reasons.

#### 5.4.1.1 Analysis of the SA Parameters

According to section 5.2.2, there are several choices for two of the SA parameters: $T_0$ and the temperature decrement rule. The different combinations of their values give rise to nine cases (see Table 5.1).

| Cases | |
|---|---|
| **Case 1** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 2** | $T_0 = n$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 3** | $T_0 = n$ and $T_{k+1} = \alpha T_{k-1}$ $(\alpha = 0.9)$ (Geometric decrease, $\alpha = 0.9$) |
| **Case 4** | $T_0 = 500$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 5** | $T_0 = 500$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 6** | $T_0 = 500$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |
| **Case 7** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease) |
| **Case 8** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease) |
| **Case 9** | $T_0 = \frac{n}{4}$ and $T_{k+1} = \alpha T_{k-1}$ (Geometric decrease, $\alpha = 0.9$) |

Table 5.1: Studied cases for SA.

These nine cases were analyzed by comparing the number of vertex floodlights, the runtime and the number of iterations performed by each of them. Table 5.2 presents the obtained results with the first three cases. Table 5.3 presents the obtained results with the Cases 4, 5 and 6. Finally, Table 5.4 presents the obtained results with the last three cases. These tables, as can be seen, show the average time of pre-processing in seconds ($PP$), the average number of vertex floodlights ($|F|$), the average runtime in seconds ($Time$) and the average number of iterations of the algorithm ($Iter.$).

| $n$ | Case 1 (FSA dec.) | | | | Case 2 (VFSA dec.) | | | | Case 3 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. |
| 30 | 0.15 | 7.55 | 14.25 | 4839.40 | 0.20 | 9.17 | 1.00 | 9.00 | 0.25 | 9.05 | 5.300 | 83.00 |
| 50 | 0.62 | 12.75 | 43.67 | 6281.40 | 0.50 | 14.60 | 3.45 | 10.00 | 0.57 | 14.45 | 18.00 | 88.00 |
| 70 | 1.35 | 17.85 | 97.22 | 8113.10 | 1.30 | 20.42 | 7.32 | 10.00 | 1.05 | 19.92 | 40.30 | 91.00 |
| 100 | 2.60 | 25.77 | 206.72 | 10130.00 | 2.65 | 28.62 | 17.00 | 10.00 | 2.65 | 28.92 | 88.80 | 94.00 |

Table 5.2: Results obtained with SA Cases 1, 2 and 3 ($T_0 = n$).

In the first three cases, the selection of the initial temperature depends on the input of the problem, that is, it depends on $n$, number of vertices of $P$. It was considered $T_0 = n$, may be ground for future research studying the behaviour of approximation method for non-linear functions in the initial temperature. As we can see, the best solution appears to correspond to a slow decrease in temperature, FSA, with a larger number of iterations and a higher response time, i.e., the best solution given in these first three cases seems to be obtained by Case 1.

In the following three cases it is going to be analyzed how the different types of temperature decrease behave, being $T_0 = 500$. As for the MVGS($P$) problem, here it was chosen $T_0 = 500$, because the number of vertices of the analyzed polygons is 50, 100, 150 and 200 and in with this way we have a constant value greater than any $n$ value and considered small enough so that the algorithm is executed in a reasonable time.

| $n$ | Case 4 (FSA dec.) | | | | Case 5 (VFSA dec.) | | | | Case 6 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. |
| 30 | 0.25 | 6.70 | 185.82 | 39005.00 | 0.15 | 8.575 | 14.22 | 12.00 | 0.10 | 8.65 | 89.55 | 110.00 |
| 50 | 0.47 | 11.30 | 380.92 | 38707.00 | 0.65 | 14.35 | 28.70 | 12.00 | 0.57 | 14.25 | 178.725 | 110.00 |
| 70 | 1.12 | 16.07 | 605.65 | 38703.00 | 1.15 | 20.35 | 45.325 | 12.00 | 1.20 | 20.15 | 279.87 | 110.00 |
| 100 | 2.77 | 24.00 | 954.37 | 39042.00 | 2.67 | 28.90 | 71.475 | 12.00 | 2.67 | 29.52 | 435.77 | 110.00 |

Table 5.3: Results obtained with SA Cases 4, 5 and 6 ($T_0 = 500$).

As we can see, the best solution in these three cases seems to be achieved by Case 4. Comparing these last three with the first three cases for the same type of temperature decrease, that is, Case 1 with Case 4, Case 2 to Case 5 and Case 3 with Case 6. We can see that the solutions provided by Case 4 seems to be better than the solutions provided by Case 1. Concerning Cases 2 and 5, it appears that the obtained solutions are almost equal (except for $n = 30$, where Case 5 seems to be a little better), being Case 5 slower. Finally, Case 3 seems to obtain slightly better solutions than Case 6, for $n = 70$ and 100; and Case 6 seems to obtain slightly better solutions than Case 3, for $n = 30$, being Case 3 always faster than Case 6.

So, it seems that when the FSA decrease is used the obtained solutions appear to be better when $T_0 = 500$. When the VFSA decrease is used the initial temperature does not seem to have much influence. Finally, for a geometric decrease the obtained solutions appears to be better for $T_0 = n$ and $n = 70$ and $n = 100$. We can also see that, in general, if a solution nearer to the optimal one is searched it seems that it is more suitable to choose the FSA decrease and $T_0 = 500$.

Notice that, it was to be expected that a geometrical decrease should produce better solutions than a fast decrease, what does not happen (see Cases 3 and 4 and Cases 5 and 6). The reason for this behaviour is the elimination of the redundant floodlights, so that the results seem to have not considerable differences.

In the following cases, it is going to be analyzed how the three temperature decreases behave if the $T_0 = \frac{n}{4}$. As for the MVGS($P$) problem, this value was chosen because it not only links $T_0$ with the algorithm input, but also it is lower than $n$, and we wanted to see how the algorithm behave under these conditions.

| $n$ | Case 7 (FSA dec.) | | | | Case 8 (VFSA dec.) | | | | Case 9 (Geometric dec.) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. | PP | $\|F\|$ | Time | Iter. |
| 30 | 0.32 | 8.37 | 4.17 | 1399.000 | 0.175 | 9.87 | 0.35 | 8.00 | 0.05 | 8.60 | 1.60 | 69.00 |
| 50 | 0.30 | 13.85 | 15.37 | 2399.000 | 0.525 | 15.72 | 1.25 | 8.00 | 0.37 | 14.47 | 5.05 | 74.00 |
| 70 | 1.12 | 19.70 | 34.70 | 3399.000 | 1.200 | 21.92 | 2.75 | 9.00 | 1.20 | 20.10 | 10.60 | 78.00 |
| 100 | 2.67 | 28.05 | 76.40 | 4634.00 | 2.65 | 31.02 | 6.40 | 9.00 | 2.70 | 28.97 | 24.27 | 81.00 |

Table 5.4: Results obtained with SA Cases 7, 8 and 9 ($T_0 = \frac{n}{4}$).

Observing these last three cases we verify that, the best solutions seem to be obtained with Case 7, followed by the solutions obtained by Case 9 and, finally, the solutions obtained by Case 8. Of the nine cases, the best case seems to be Case 4, which corresponds to a constant initial temperature and a slow temperature decrease.

As always, a statistical study was carried out. The data obtained with Case 1 is non-normally distributed, for $n = 30, 50, 70$, and 100 (the obtained $p$-values are less than 0.001, for $n = 30, 50, 70$ and 100). The $p$-values returned by the Kruskal-Wallis tests were $< 0.001 < 0.05$ for the data obtained with the polygons with $n = 30, 50, 70$ and 100. Then multiple comparison tests were performed. The answers provided by these tests are presented in Tables 5.5, 5.6, 5.7 and 5.8. The sign "+" indicates that the sample data (concerning $|F|$) is significantly different and the sign "-" indicates otherwise.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | - | + | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | + | - | - | + | • | - | - | + | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | - | - | - | + | - | - | • | + | - |
| Case 8 | + | - | - | + | + | + | + | • | + |
| Case 9 | + | - | - | + | - | - | - | + | • |

Table 5.5: Multiple comparison tests, of SA Cases, for 30-vertex orthogonal polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | - | + | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | + | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | - | - | - | + | - | - | • | + | - |
| Case 8 | + | - | - | + | - | + | + | • | - |
| Case 9 | + | - | - | + | - | - | - | - | • |

Table 5.6: Multiple comparison tests, of SA Cases, for 50-vertex arbitrary polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | + | + | + |
| Case 2 | + | • | - | + | - | - | - | - | - |
| Case 3 | + | - | • | + | - | - | - | + | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | + | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | + | - |
| Case 7 | + | - | - | + | - | - | • | + | - |
| Case 8 | + | - | + | + | - | + | + | • | + |
| Case 9 | + | - | - | + | - | - | - | + | • |

Table 5.7: Multiple comparison tests, of SA Cases, for 70-vertex arbitrary polygons.

| Methods | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | • | + | + | - | + | + | + | + | + |
| Case 2 | + | • | - | + | - | - | - | + | - |
| Case 3 | + | - | • | + | - | - | - | - | - |
| Case 4 | - | + | + | • | + | + | + | + | + |
| Case 5 | + | - | - | + | • | - | - | - | - |
| Case 6 | + | - | - | + | - | • | - | - | - |
| Case 7 | + | - | - | + | - | - | • | + | - |
| Case 8 | + | + | - | + | - | - | + | • | - |
| Case 9 | + | - | - | + | - | - | - | - | • |

Table 5.8: Multiple comparison tests, of SA Cases, for 100-vertex arbitrary polygons.

The multiple comparison tests, also, allowed to conclude that:

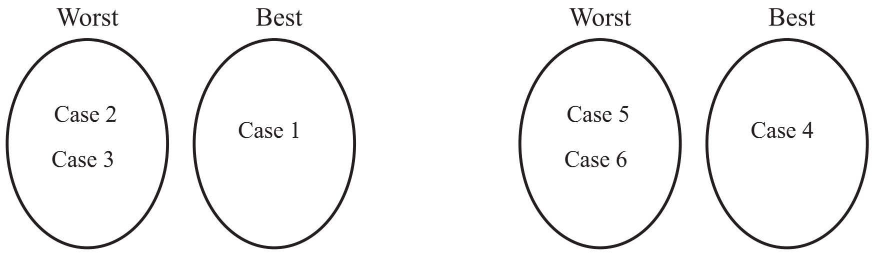
- for $n = 30$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 2 with no significant differences from Case 3 (see Figure 5.10 (a));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 5.10 (b));
  - Cases 7, 8 and 9. The best is Case 7 with no significant differences from Case 9; the worst is Case 8 with significant differences from Cases 7 and 9 (see Figure 5.11 (a));
  - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 8, with no significant differences from Cases 2 and 3.
- for $n = 50$, concerning
  - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 2 with no significant differences from Case 3 (see Figure 5.10 (a));
  - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and

6; the worst is Case 5 with no significant differences from Case 6 (see Figure 5.10 (b));
   - Cases 7, 8 and 9. The best is Case 7 with no significant differences from Case 9; the worst is Case 8 with no significant differences from Case 9 (see Figure 5.11 (b));
   - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 8, with no significant differences from Cases 2, 3, 5, and 9.

- for $n = 70$, concerning
   - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 2 with no significant differences from Case 3 (see Figure 5.10 (a));
   - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 5 with no significant differences from Case 6 (see Figure 5.10 (b));
   - Cases 7, 8 and 9. The best is Case 7 with no significant differences from Case 9; the worst is Case 8 with significant differences from Cases 7 and 9 (see Figure 5.11 (b));
   - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 8, with no significant differences from Cases 2 and 5.

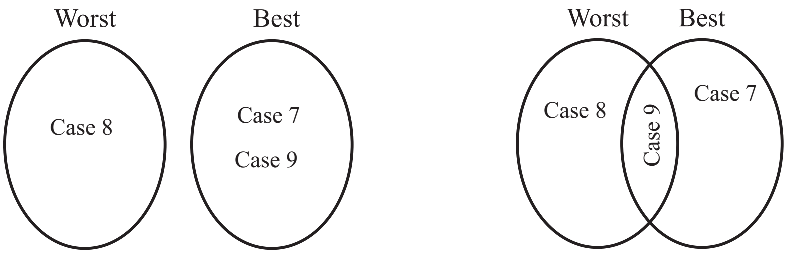
- for $n = 100$, concerning
   - Cases 1, 2 and 3. The best is Case 1 with significant differences from Cases 2 and 3; the worst is Case 3 with no significant differences from Case 2 (see Figure 5.10 (a));
   - Cases 4, 5 and 6. The best is Case 4 with significant differences from Cases 5 and 6; the worst is Case 6 with no significant differences from Case 5 (see Figure 5.10 (b));
   - Cases 7, 8 and 9. The best is Case 7 with no significant differences from Case 9; the worst is Case 8 with no significant differences from Case 9 (see Figure 5.11 (b));
   - the nine cases. The best is Case 4, with no significant differences from Case 1; the worst is Case 8, with no significant differences from Cases 3, 5, 6, and 9.

As it can be noticed, on using the multiple comparison tests, for $T_0 = n$ and $T_0 = 500$ the solutions are always significantly better when the temperature decrease is slow (FSA) (see Figure 5.10 (a) and (b), respectively) and for $T_0 = \frac{n}{4}$, the best solutions are obtained when the temperature decrease is slow with no significantly differences with a geometric temperature decrease (see Figure 5.11).




(a) $n = 30, 50, 70$ and 100 (b) $n = 30, 50, 70$ and 100

Figure 5.10: Multiple comparison tests of: (a) Cases 1, 2 and 3; (b) Cases 4, 5 and 6.




(a) $n = 30$ and 70 (b) $n = 50$ and 100

Figure 5.11: Multiple comparison tests of Cases 7, 8 and 9.

Notice that, for all types of initial temperature $T_0$, it would be expected that a geometrical decrease should produce better solutions than a rapid decrease what does not happen. We can see that the obtained solutions have not significant differences (except when $T_0 = \frac{n}{4}$, with $n = 30$ and $n = 70$). As stated before, the reason for this behavior is the elimination of the redundant vertex floodlights.

Concerning the nine cases, if the temperature decrease is slow, the best solutions are obtained with $T_0 = 500$ and $T_0 = n$. If the temperature decrease is fast the initial temperature does not have influence, except for $n = 100$, where the obtained solutions by Case 8 ($T_0 = \frac{n}{4}$) are significantly worst than Case 2 ($T_0 = n$). If the temperature decrease is geometric the initial temperature does not have influence, despite the observation that different initial temperatures seem to give rise to different solutions (see Tables 5.2, 5.3 and 5.4). It can also be concluded that the best solutions are obtained with Case 4 for $n = 30$ 50, 70 and 100 and a significant difference was not found between the number of vertex floodlights obtained with this case and with Case 1. Despite the observation that in Tables 5.2 and 5.3, Case 4 seems to outperform Case 1 concerning the average number of vertex floodlights $|F|$, for $n = 30, 50, 70$ and 100. So, the statistical analysis proceeds regarding the runtime. This analyze was made in a similar way and it allowed to conclude that Case 1 is significantly faster than Case 4, for $n = 30, 50, 70$ and 100. Given this, Case 1 was selected to be the best case.

Notice, however, that the rapid decreases are useful when faster, but worse, solutions are wished. For instance, remember that it is necessary to choose a SA strategy for the hybrid methods. Case 8 is the fastest case and although the returned number of floodlights is the worst, it can be used in the hybrid methods. This is due to the fact that, one of the objectives of these methods is to see how they behave when compared to a "pure" GA strategy. In the first method, the initial population generated by the SA strategy is always better than the one proposed for the "pure" algorithm in subsection 5.2.3. In the second method, being the initial temperature low (as $T_0 = \frac{n}{4}$), the danger of losing the good solutions found so far is decreased. Moreover, as intended, the intensification keeps on being reinforced in the search carried out by the GA. Thus, Case 8 was the SA strategy selected to use in the developed hybrid methods.

**Improvements**

As it was done for the MVGS($P$) problem, it was found that the runtime of the best case (Case 1) could be improved if a *floodlight dominance matrix* is computed in the pre-processing step. (see subsection 4.4.1.1). This concept is defined as follows.

**Definition 5.6** *Let $P$ be a polygon with $n$ vertices and $v_i, v_j \in V_P$. The vertex floodlight $f_k^i$* ***dominates*** *a vertex floodlight $f_m^j$ (or, $f_m^j$ is dominated by $f_k^i$) if $Vis(f_m^j, P) \subset Vis(f_k^i, P)$.*

**Definition 5.7** *Let $P$ be a polygon with $n$ vertices. The* ***floodlight dominance matrix of*** *$P$ is a $(n+r) \times (n+r)$ matrix $A$, where $\forall k, m \in \{0, \ldots, n+r-1\}$, $A[k,m] = 1$, if the k-esim floodlight dominates the m-esim floodlight; and $A[k,m] = 0$, otherwise.*

The floodlight dominance matrix of $P$ could improve the runtime of the above described algorithms, since each time that it is necessary to see if $F \setminus \{f_k^i\}$, with $f_k^i \in F$, is still a vertex floodlighting set for $P$, being $F$ a vertex floodlighting set for $P$. First, it is checked if $f_k^i$ is dominated by some other floodlight of $F$, that is, if $A[k,m] = 1$, for some $f_m^j \in F$. If so, we already know that $F \setminus \{f_k^i\}$ is vertex floodlighting set, and it is not necessary to determine if $\bigcup_{f_i \in F \setminus \{f_k^i\}} Vis(f_k^i, P) = P$, since this is true.

Therefore, the calculation dominance matrix of $P$ was included in the pre-processing step and new results were obtained with Case 1. Table 5.9 show the obtained results with Case 1, with and without the dominance matrix. We can observe that the runtime improves when the dominance matrix is employed. In spite of the pre-processing time increase, the overall time (pre-processing time plus runtime) is improved when the dominance matrix is employed. Given that, Case 1, with the calculation of the dominance matrix in the pre-processing step, was selected to be method $M_1$.

| $n$ | Case 1 (without dominance matrix) | | | | Case 1 (with dominance matrix) | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\lvert F\rvert$ | Time | Iter. | PP | $\lvert F\rvert$ | Time | Iter. |
| 30 | 0.15 | 7.55 | 14.25 | 4839.40 | 1.62 | 7.52 | 13.45 | 5020.30 |
| 50 | 0.62 | 12.75 | 43.67 | 6281.40 | 2.60 | 10.50 | 25.57 | 5604.10 |
| 70 | 1.35 | 17.85 | 97.22 | 8113.10 | 8.42 | 18.02 | 86.35 | 7897.00 |
| 100 | 2.60 | 25.77 | 206.72 | 10130.00 | 17.30 | 25.12 | 185.10 | 10202.00 |

Table 5.9: Results obtained with SA Case 1, with and without the use of the dominance matrix.

#### 5.4.1.2 Comparison of the four strategies

In this section it is analyzed and evaluated the results obtained with the four approximation methods: $M_1$, SA strategy; $M_2$, GA strategy and and the hybrid strategies which are going to be denoted by $M_3$ (the strategy in which the initial population of a GA is generated by a SA algorithm) and $M_4$ (the strategy in which a SA strategy is a genetic operator). Remember that, for the hybrid strategies, $M_3$ and $M_4$, is necessary to choose a SA strategy and a GA strategy. As stated above, the selected SA strategy was the SA Case 8 ($T_0 = \frac{n}{4}$ and VFSA temperature decrease) and the selected GA strategy was the developed GA algorithm (method $M_2$). Experiments were done with the methods $M_3$ and $M_4$, with and without the calculation of the dominance matrix in the pre-processing step, and it was concluded that using the dominance matrix these methods were faster. So, all the results associated with the hybrid strategies were obtained using the dominance matrix.

Tables 5.10 and 5.11 present the obtained results with methods $M_1$, $M_2$, $M_3$ and $M_4$. Comparing the solutions obtained with the non-hybrid methods, (see Table 5.10) we can notice that: the method $M_1$ appears to be the best method since it seems to be faster and the obtained solutions appear to be better. In relation to the results obtained using the hybrid methods (see Table 5.11) we can see that $M_4$ is slower but the average number of vertex floodlights seems to be better.

| $n$ | $M_1$ | | | | $M_2$ | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\lvert G\rvert$ | Time | Iterations | PP | $\lvert G\rvert$ | Time | Iterations |
| 50 | 4.02 | 12.52 | 39.85 | 6436.30 | 0.450 | 14.20 | 37.37 | 1071.10 |
| 100 | 17.30 | 25.12 | 185.10 | 10202.00 | 2.52 | 29.05 | 290.35 | 2278.30 |
| 150 | 40.70 | 37.65 | 441.57 | 13661.00 | 7.12 | 42.02 | 763.60 | 3284.60 |
| 200 | 75.82 | 50.25 | 871.90 | 17972.00 | 14.92 | 56.82 | 1584.50 | 4568.10 |

Table 5.10: Results obtained with $M_1$ and $M_2$.

| $n$ | $M_3$ | | | | $M_4$ | | | |
|---|---|---|---|---|---|---|---|---|
| | PP | $\lvert G\rvert$ | Time | Iterations | PP | $\lvert G\rvert$ | Time | Iterations |
| 50 | 4.12 | 12.27 | 28.05 | 580.60 | 4.17 | 11.62 | 44.17 | 683.85 |
| 100 | 17.92 | 25.57 | 243.20 | 754.55 | 17.47 | 23.30 | 373.55 | 1061.50 |
| 150 | 41.55 | 38.47 | 842.50 | 843.05 | 41.17 | 34.70 | 1078.10 | 1241.40 |
| 200 | 76.00 | 52.22 | 2163.30 | 919.05 | 76.17 | 46.70 | 2660.20 | 1520.60 |

Table 5.11: Results obtained with $M_3$ and $M_4$.

Contrasting, now, the results achieved with the hybrid methods ($M_4$ and $M_5$) with the results obtained with the non-hybrid strategies ($M_1$ and $M_2$), we can observe that $M_4$ seems to obtain better solutions than $M_1$, for $n = 50, 100, 150$ and 200. Consequently, regarding the average number of vertex floodlights, the best method seems to be the method $M_4$, followed by the methods $M_1$ and $M_3$; and the worst one seems to be the method $M_2$ (see Figure 5.12).

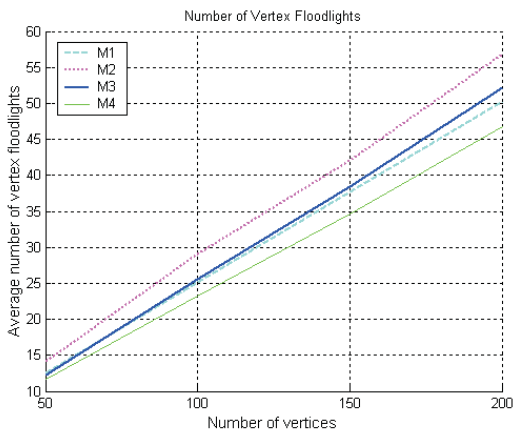



Figure 5.12: Solutions obtained with strategies $M_1, M_2, M_3$, and $M_4$.

As usual, a statistical study was carried out. The data obtained with method $M_1$ is non-normally distributed, for $n = 50, 100, 150$ and 200 (the obtained $p$-values are less than 0.001, for $n = 50, 100, 150$ and 200). The $p$-values returned by the Kruskal-Wallis tests were $< 0.001 < 0.05$ for the data obtained with the polygons with $n = 50, 100, 150$ and 200. Then multiple comparison tests were performed to determine which pairs of averages were significantly different, and which were not. The answers provided by these tests allowed to conclude that (see Figures 5.13(a) and 5.13(b)):

- for $n = 50$, the best method is $M_4$, with no significant differences with the method $M_3$; and the method $M_2$ is significantly worse than the other methods.
- for $n = 100, 150$ and 200, the method $M_4$ is significantly better than the other methods and the method $M_2$ is significantly worse than the other methods.

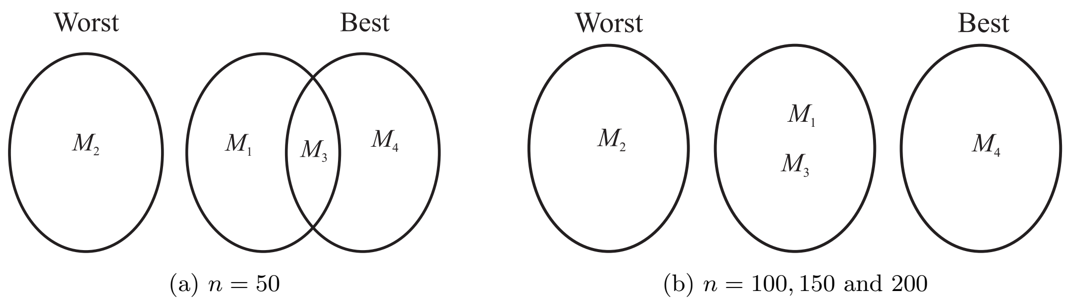



Figure 5.13: Multiple comparison tests of the five methods.

Therefore, unmistakably, concerning the obtained solutions, the hybrid method $M_4$ is the best one and the method $M_2$ the worst one. The methods $M_1$ and $M_3$ can be considered equal. Consequently, the study continues considering $M_4$ as the best strategy.

To infer about the average of the minimum number of vertex floodlights needed to cover an orthogonal polygon, it was applied $M_4$ to eight sets of arbitrary polygons, each one with 40 polygons with 30, 50, 70, 100, 110, 130, 150 and 200 vertex polygons. The average of the obtained results, concerning $|F|$, are shown in Table 5.12.

| $n$ | 30 | 50 | 70 | 100 | 110 | 130 | 150 | 200 |
|---|---|---|---|---|---|---|---|---|
| $\lvert F\rvert$ | 6.97 | 11.62 | 16.25 | 23.30 | 25.00 | 30.12 | 34.70 | 46.70 |

Table 5.12: Average of the minimum number of vertex floodlights.

Then, using the least squares method, the following linear adjustment was obtained, with a correlation factor of 0.9997 (see Figure 5.14):

$$f(x) = 0.2328x - 0.1091 \approx \frac{x}{4.29} - 0.1091 \approx \frac{x}{4.29}.$$

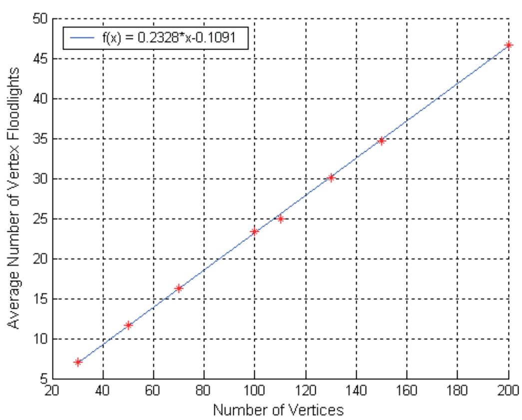
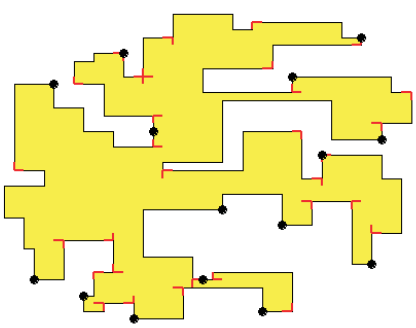


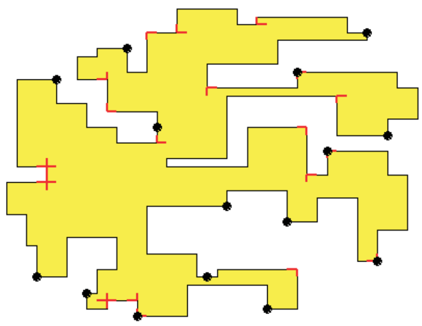

Figure 5.14: Least Squares Method.

Thus, it can be concluded that on average, and approximately, the minimum number of vertex floodlights needed to cover an orthogonal polygon with $n$ vertices was observed to be $\lceil \frac{n}{4.29} \rceil$. In order to get a quantitative measure on the quality of the calculated $|F|$, the floodlights visibility-independent sets were computed on our instances (the eight sets of polygons described above). The ratio between the smallest $F$ (obtained with $M_4$ ) and the largest visibility-independent set, *FIS* obtained with $A_1$ (see section 5.3) never exceeded 2 (with an average of 1.68 for the universe of 320 polygons). That implies that algorithm $M_4$ has an approximation ratio less than or equal to 2.

Figure 5.15 shows snapshots obtained with our software. In this figure is illustrated an orthogonal polygon for which the floodlight visibility-independent set *FIS* was obtained with $A_1$, $|FIS| = 15$, and the solution $F$ was obtained with $M_2$ and $M_4$, $|F| = 31$ and $|F| = 22$, respectively.



(a) $|FIS| = 15$ and $|G| = 31$



(b) $|FIS| = 15$ and $|G| = 22$

Figure 5.15: *FIS* and $F$ sets obtained on a 100-vertex orthogonal polygon with the methods $A_1$ and: (a) $M_2$; (b) $M_4$.

## 5.5 Concluding Remarks

In this chapter approximation algorithms were proposed that allow to obtain a vertex floodlighting set $F$, whose cardinality approximates the minimal number of vertex floodlights needed to illuminate a given polygon orthogonal. In other words, approximation algorithms were designed and implemented to tackle the MINIMUM VERTEX FLOODLIGHT SET problem on orthogonal polygons. Four approximation strategies were studied: one based on the SA metaheuristic ($M_1$), one based on the GAs metaheuristic ($M_2$) and two others based on hybrid metaheuristics ($M_3$ and $M_4$). It was also developed a greedy algorithm to compute floodlights visibility-independent sets, permitting to obtain provable bounds on how close our results are to the optimal.

Using a large set of randomly generated orthogonal polygons, an experimental comparative study was made on the suitability of the developed methods, allowing to conclude that:

(1) Concerning the SA strategy. The best case was observed to be Case 1, that is, $T_0 = n$ and a slow temperature decrease (FSA).

(2) About the hybrid strategies it was observed that the hybrid strategy $M_4$ (a SA strategy is a genetic operator in the developed GA) is significantly better (except for $n = 50$) then the other one (the initial population of the GA is generated by SA). Thus, the use of a SA strategy as a genetic operator to reinforce the intensification on the search carried out by the GA, improves the obtained solutions. It was also seen that both hybridizations obtain significantly better solutions than the "pure" GA, i.e., method $M_2$

(3) Finally and as to the four approximation strategies, the best one was observed to be method $M_4$. The performed computational experiments allowed to conclude that on average, and approximately, the minimal number of vertex floodlights needed to cover an orthogonal polygon was observed to be $\lceil \frac{n}{4.29} \rceil$. This value is much less than the theoretical bound $\lfloor \frac{3n-4}{8} \rfloor$. Finally, in terms of quality of the solutions, it is also concluded that the approximation ratio is less than or equal to 2.

It is important to point out, again, that all alternatives with respect to parameters of the SA, GA and the hybrid metaheuristic that could be explored are almost "infinite". Once more, in this work it was attempted to find references for these parameters, noting that a more exhaustive study in future investigations might improve the obtained results.

As a conclusion, the hybrid metaheuristics, especially the strategy $M_4$, have proven to behave well in solving the MINIMUM VERTEX FLOODLIGHT SET problem. However, the obtained approximation ratio was not as good as for the MVSG($P$) problem. This behaviour may be due to method $M_4$ or due to the greedy strategy $A_1$. Therefore, it is our intention, as a future research, to investigate and detect where the problem is and to improve $M_4$ (or study different hybridizations) and/or the greedy strategy $A_1$ (or study different approximation methods).

# Chapter 6

# Minimum Vertex $k$-Modem Set Problem

In the previous chapters it was used the classical visibility definition, which ensures that two points $x$ and $y$ on a polygon $P$ are visible to each other if the line segment $\overline{xy}$ does not intersect the exterior of $P$, that is, if $\overline{xy} \cap P = \overline{xy}$. Nevertheless, the development of the Internet and of the wireless networks inspire further research in the visibility field of computational geometry, as shown in [33, 53, 131].

Recently in [9] was defined a new variant of the original Art Gallery Problem that arises from the following everyday and practical problem: *How to place wireless modems in a building in such a way that a computer, with a wireless card, placed anywhere within the building receives a signal strong enough to have a stable connection to navigate in the Web?* There are two key limiting factors to connect a computer to a wireless network: its distance to the wireless modem and the number of walls that separate it from the modem. However, experience says that, in most buildings, the most significant limiting factor is the number of walls that separate the computer from the wireless modem and not its distance to the modem.

This was the practical motivation that encouraged Aichholzer *et al.* [9] to study the $k$-modem Art Gallery Problem: *Given a polygon $P$ with $n$ vertices, what is the minimum number of $k$-modems (placed on points of $P$) sometimes necessary and always sufficient to cover $P$?* It is said that a point $y$ in a polygon $P$ is covered or illuminated by a $k$-modem placed on a point $x \in P$ if the line segment $\overline{xy}$ crosses at most $k$ walls (edges) of $P$. It is easy to observe that $(i)$ for $k = 0$ this problem is reduced to the original Art Gallery problem and $(ii)$ for $k = n$ would be enough only one $n$-modem placed on any point of $P$ to cover $P$ (trivial solution). Combinatorial bounds for this problem were obtained for arbitrary monotone and orthogonal monotone polygons, remaining open the problem for general arbitrary and orthogonal polygons [9]. Later, Fabila-Monroy, Vargas and Urrutia extended the notion of covering with $k$-modems to other geometric configurations, such as families of line segments, families of lines and sets of horizontal or vertical disjoint segments, or sets of lines [56].

As always, in this dissertation the geometrical configurations were confined to polygons. As for the MVFS($P$), it is strongly believed that the problem of finding the minimum number of $k$-modems, needed to cover a given polygon, is $\mathcal{NP}$-hard, both for arbitrary and orthogonal polygons [9]. So, it makes sense makes sense to tackle it by applying approximate resolution methods. In the next section the problem will be formalized. In section 6.2 it will be presented an algorithm to calculate the region covered by a $k$-modem located on a point of a polygon with $n$ edges, for all the possible values of $k$ ($0 \leq k \leq n$). In section 6.3 it will be discussed a metaheuristic method designed to solve approximately the problem of minimizing the number of $k$-modems, based on a hybrid approach that uses both the genetic algorithms and simulated annealing metaheuristics. Finally, section 6.4 will present the experimental results obtained with this method, for $k = 2$ and $k = 4$, on randomly generated arbitrary and orthogonal polygons, and in section 6.5 some conclusion are presented.

## 6.1 Problem Description

In this chapter it is going to be studied the problem of covering a polygon $P$ with a set of wireless modems. As stated above, there are two key limiting factors to connect a computer to a wireless network: its distance to the wireless modem and the number of walls that separate it from the modem. In a first approach, only the number of walls that separates the computer from the modem will be considered, while the distance to the modem will be ignored in this work. So, first of all, it is necessary to define when a computer located on point $y \in P$ is covered or illuminated by a wireless modem placed on a point $x \in P$.

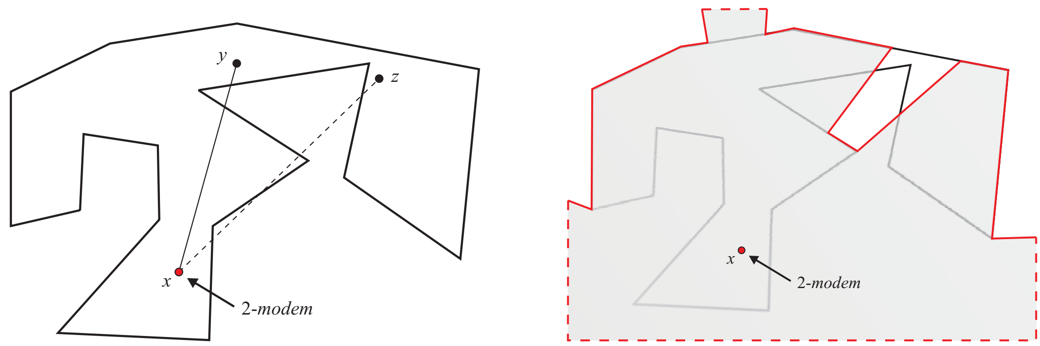
**Definition 6.1** *Let $P$ be a $n$-vertex polygon. A wireless modem, located on a point $x \in P$, which transmits a stable signal through at most $k$ edges (walls) of $P$ along a straight line is denoted by **k-modem**.* [9].

**Definition 6.2** *Let $P$ be a $n$-vertex polygon and $k \in \{0, \ldots, n\}$. A point $y \in P$ is **covered** by a $k$-modem placed on $x \in P$ if the line segment $\overline{xy}$ crosses at most $k$ edges (walls) of $P$ (see Figure 6.1(a)), that is, $y$ is **covered** by a $k$-modem placed on $x$ if the line segment $\overline{xy}$ intersects the relative interior of the edges of $P$ at most $k$ times.*

**Definition 6.3** *Let $P$ be a $n$-vertex polygon. The **k-modem visibility region** of a $k$-modem placed on $x \in P$ is the set of all points $y \in \mathbb{R}^2$ that is covered by $x$ and it is denoted by $Vis_k(x, P)$, that is, $Vis_k(x, P) = \{y \in \mathbb{R}^2 : x \text{ covers } y\}$, where $x$ is a $k$-modem*[1].

[1]Being $x \in P$, to simplify the presentation, sometimes the expression "$x$ is a $k$-modem" is used to mean "a $k$-modem placed on $x$".

Figure 6.1 (b) illustrates the visibility region of a 2-modem. Note that, this region can be, in some cases, unlimited.




(a) (b)

Figure 6.1: (a) The 2-modem placed on $x$ covers $y$ but it does not cover $z$; (b) $Vis_2(x, P)$.

Now, the following problem, $k$-modem Art Gallery Problem, can be posed: *Given a polygon $P$ with $n$ vertices, what is the minimum number of $k$-modems (placed on points of $P$) sometimes necessary and always sufficient to cover $P$?* [9].

Let $P$ be a polygon with $n$ vertices. Let $g_{km}(P)$ be the smallest number of $k$-modems needed to cover $P$:

$$g_{km}(P) = \min\{|S| : S \subset P,\ P \subset \bigcup_{x \in S} Vis_k(x, P)\}.$$

Denote by $G_{km}$ the maximum of $g_{km}(P)$ over all polygons with $n$ vertices:

$G_{km}(n) = \max\{g_{km}(P) : P \in P_n\}$, where $P_n$ denotes the set of all polygons with $n$ vertices.

Thus, $G_{km}(n)$ $k$-modems always suffice to cover any $n$-vertex polygon, and are necessary to cover at least one $n$-vertex polygon. This will be rewrite as: $G_{km}(n)$ $k$-modems are always sufficient and occasionally necessary, or just sufficient and necessary. So, the above established problem asks for $G_{km}(n)$.

Aichholzer et al. [9] studied this problem for monotone polygons, so some useful definitions concerning these class of polygons will follow.

**Definition 6.4** *A polygonal chain $p_0, \ldots, p_k$ is called **monotone with respect to a line l** if the projections of $p_0, \ldots, p_k$ onto $l$ are ordered in the same way as in the chain. Two adjacent vertices $p_i$ and $p_{i+1}$ may project to the same point on $l$ without destroying monotonicity* [101].

**Definition 6.5** *A polygonal chain is called **monotone** if it is monotone with respect to at least one line* [101]. *It will be used the convention that that the line of monotonicity is the $x$-axis.*

**Definition 6.6** *A polygon is* ***monotone*** *if it can be partitioned in two monotone polygonal chains with respect to the same line* [101]*. These two polygonal chains will be designated by the bottom and top chains (see Figure* 6.2*).*

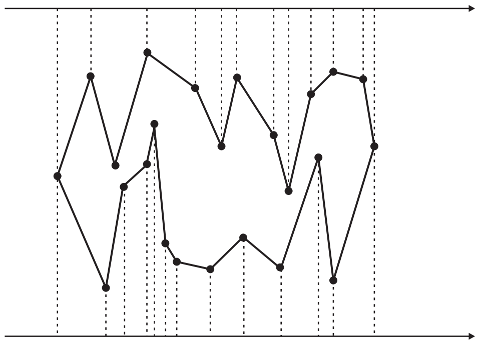


Figure 6.2: The vertices of a monotone polygon projected onto a line.

Concerning monotone polygons Aichholzer et al. [9] proved that:

**Proposition 6.1** *Every monotone polygon with n vertices can be covered with* $\lceil \frac{n}{2k} \rceil$ *$k$-modems, and there is a monotone $n$-vertex polygon that requires at least* $\lceil \frac{n}{2k+2} \rceil$ *$k$-modems to be covered.*

So, $\lceil \frac{n}{2k+2} \rceil \leq G_{km}(n) \leq \lceil \frac{n}{2k} \rceil$. A polygon achieving the lower bound is shown in Figure 6.3. For $k = 1, 2, 3$, they obtained a better upper bound: $G_{km}(n) \leq \lceil \frac{n}{k+4} \rceil$.

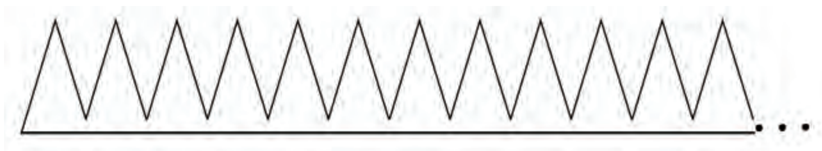


Figure 6.3: A $n$-vertex monotone polygon requiring $\lceil \frac{n}{2k+2} \rceil$ $k$-modems [9].

These authors also proved that:

**Proposition 6.2** *Every monotone orthogonal polygon with $n$ vertices can be covered with* $\lceil \frac{n-2}{2k+4} \rceil$ *$k$-modems.*

If $k$ is even, the bound established in the previous proposition is tight. So, if $k$ is even $G_{km}(n) = \lceil \frac{n}{2k+4} \rceil$. If $k$ is odd, they proved that $\lceil \frac{n-2}{2k+6} \rceil \leq G_{km}(n) \leq \lceil \frac{n}{2k+4} \rceil$.

Summing up, Aichholzer et al. [9], obtained combinatorial bounds for the $k$-modem Art Gallery Problem for monotone (arbitrary and orthogonal) polygons, remaining open the following problems: (1) closing the gaps between the obtained lower and upper bounds; and (2) determining bounds for general arbitrary and orthogonal polygons (which is rather challenging).

If the $k$-modems are restricted to the vertices of $P$ (*vertex $k$-modems*), the combinatorial bounds established by the above theorems remain valid. Besides, it is strongly believed that the algorithmic problem of finding the minimum number of vertex $k$-modems needed to cover a given polygon is $\mathcal{NP}$-hard. This variant of the $k$-modem Art Gallery problem will be designated by MINIMUM VERTEX $k$-MODEM SET PROBLEM.

**Definition 6.7** *A given set $G_{km}$ of vertices of $P$ is a* ***covering vertex k-modem set*** *for $P$ if they cover $P$, i.e., if $P \subset \bigcup_{v \in G_{km}} Vis_k(v, P)$. A covering vertex $k$-modem set for $P$ is denoted by $G_{km}$ and its cardinality by $|G_{km}|$.*

The MINIMUM VERTEX $k$-MODEM SET problem will be denoted by MVkMS$(P, k)$ and can be stated as follows:

| |
|---|
| **MVkMS$(P, k)$** |
| **Input**: A polygon $P$ with $n$ vertices and a number $k$ of walls. |
| **Question**:What is the minimum number of vertex $k$-modems necessary to cover $P$? |

Based on the assumption that this problem is $\mathcal{NP}$-hard, it makes sense to tackle it by applying approximate resolution methods. So, in this chapter it is proposed an approximation method to tackle it. Nevertheless, no algorithm is known to determine $Vis_k(x, P)$. Consequently, the first step to solve the MVkMS$(P, k)$ problem is to develop an algorithm that calculates the region covered by a $k$-modem located on a point $x$ of a polygon with $n$ edges, that is, an algorithm that determines $Vis_k(x, P)$.

## 6.2 $k$-Modem Visibility Polygon

Let $P$ be a polygon with $n$ vertices and $x$ a point on $P$ where a $k$-modem is placed. In this section it will be presented an algorithm to construct the region covered by the $k$-modem placed on $x$. This region will have zones of the interior of $P$ and zones of its exterior. For simplicity reasons, it is considered that $P$ is contained in a rectangular box $R$ and the visibility region is constructed inside $R$. In this way, the region covered by $x$ will be always limited and will be called *$k$-modem visibility polygon* and, abusing a bit of the terminology, it will be denoted by $Vis_k(x, P)$. A vertex $v_i \in V_P$ is called a *critical vertex* for $x$ if the vertices $v_{i-1} \in V_P$ and $v_{i+1} \in V_P$ are on the same half-plane regarding the ray $\overrightarrow{xv_i}$ (see Figure 6.4).

Now it will be described an algorithm to construct the $k$-modem visibility polygon $Vis_k(x, P)$ for all possible values of $k$, that is, $0 \le k \le n$. In a first approach it will be ignored the following "degenerated" cases: (i) $x$ is collinear with two (or more) critical ver-

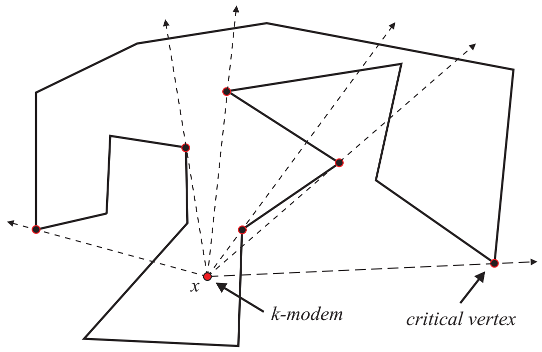



Figure 6.4: Rays and one of the critical vertices of $P$.

tices of $P$ and (ii) $x$ belongs to a line passing through an edge of $P$. The main steps of the algorithm will follow:

(1) Draw all the rays with source $x$ passing through the critical vertices of $P$. Identify, from the critical points, the intersection points of the rays with each edge of $P$ (see Figure 6.5 (a)). Sort the rays angularly about $x$.

(2) The rays divide each edge of the polygon in one or more segments. Label each segment with the number of edges (walls) crossed by the ray from $x$ to the segment (see Figure 6.5 (b)).

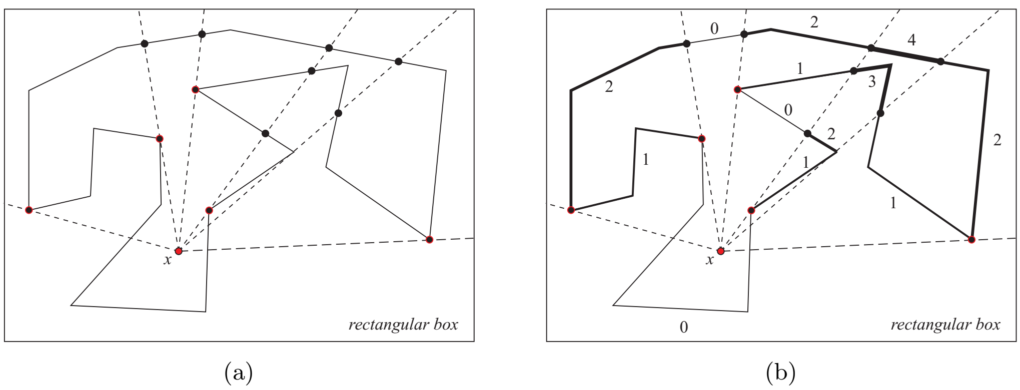


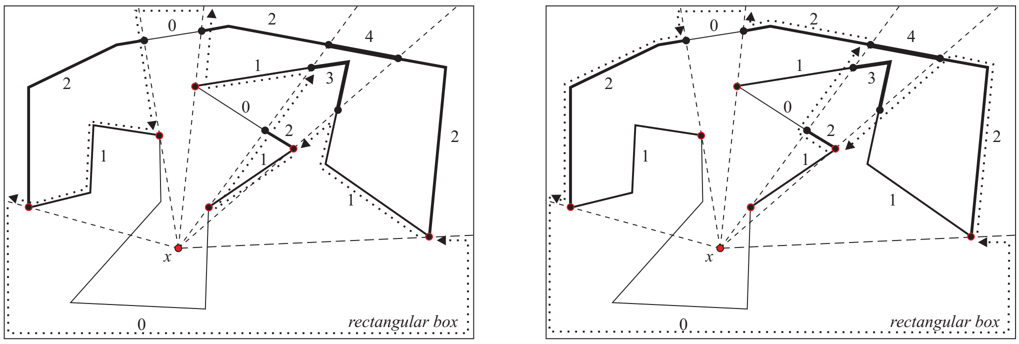

Figure 6.5: (a) Rays and intersection points; (b) Labelled segments.

(3) $Vis_k(x, P)$ is constructed by connecting the segments with label $k$, using for that the incident rays at its endpoints. Determine the first ordered ray to which the source $s_1$ of a segment with label $k$ belongs. At this point $s_1$ began to draw the boundary of $Vis_k(x, P)$, in CCW direction. Advance by $\partial P$ until reach the source $s_2$ of a segment with a different label (if $k$ is even advance in CCW direction; otherwise advance in CCW direction). If on the ray to which $s_2$ belongs there is another endpoint $p$ of a segment with label $k$ make the connection between $s_2$ and $p$ by the ray; otherwise make the

connection by the edge of the box containing the polygon.

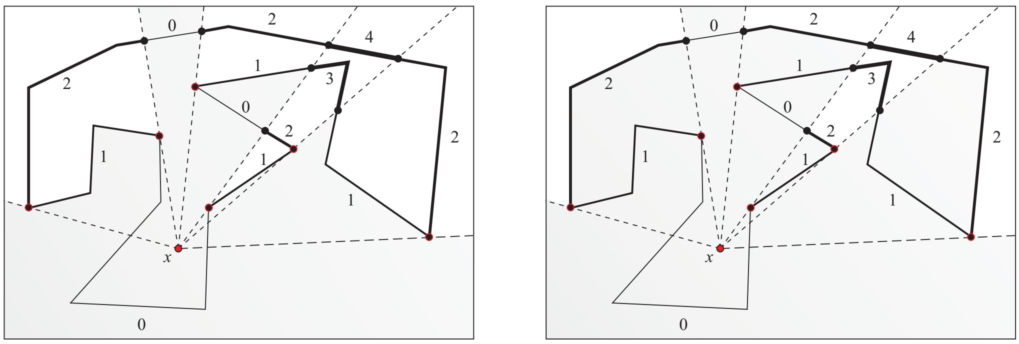
(4) Repeat the previous step until $Vis_k(x, P)$ is closed.

Figure 6.6 (a) illustrates the connection of the segments labelled with 1, to construct the polygon covered by a 1-modem and Figure 6.6 (b) shows the connection of the segments labelled with 2, to construct the polygon covered by a 2-modem. Figure 6.7 illustrates $Vis_1(x, P)$ and $Vis_2(x, P)$.




(a) (b)

Figure 6.6: Route for the construction of: (a) $Vis_1(x, P)$ and (b) $Vis_2(x, P)$.



(a) (b)

Figure 6.7: (a) $Vis_1(x, P)$ and (b) $Vis_2(x, P)$.

*Algorithm Complexity.* Since each ray can intersect all the edges of the polygon, the total number of labels of the segments is quadratic. Thus, labelling step runs in $\mathcal{O}(n^2)$ time. The construction of the visibility polygon for each fixed value of $k$ is done in linear time, as each side of $P$ only intervenes a constant number of times. To conclude, the construction of all $k$-visibility polygons is done in quadratic time.

The only step that needs a more detailed explanation is Step (2). The labelling of each segment $s$ with the number of edges crossed by the ray from $x$ to $s$, is done as follows:

1. Label all the critical vertices with "+1" or "$-1$". Being $v_i$ a critical vertex there are four rules to label it (see Figure 6.8):

   1.1 if $v_{i-1}v_iv_{i+1}$ is a left-turn and:

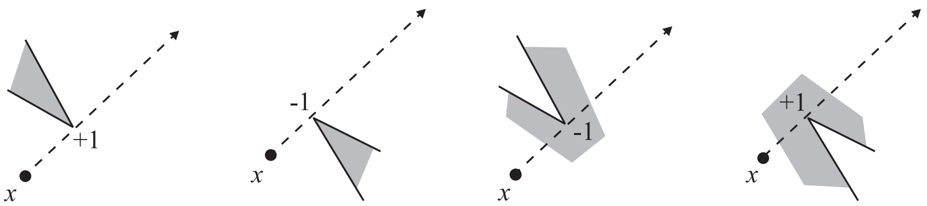
   ($i$) $v_{i-1}$ is on the positive side of the ray $\overrightarrow{xv_i}$ label $v_i$ with "+1" (**Rule 1**);

   ($ii$) $v_{i-1}$ is on the negative side of the ray $\overrightarrow{xv_i}$ label $v_i$ with "$-1$" (**Rule 2**);

   1.2 if $v_{i-1}$ is a right-turn and:

   ($i$) $v_{i-1}$ is on the positive side of the ray $\overrightarrow{xv_i}$ label $v_i$ with "$-1$" (**Rule 3**);

   ($ii$) $v_{i-1}$ is on the negative side of the ray $\overrightarrow{xv_i}$ label $v_i$ with "+1" (**Rule 4**).




(a) Rule 1 (b) Rule 2 (c) Rule 3 (d) Rule 4

Figure 6.8: Rules to label the critical vertices (shaded zones represent $int(P)$).

In Figure 6.9 it is illustrated a polygon with the critical vertices labelled according to the previous rules.

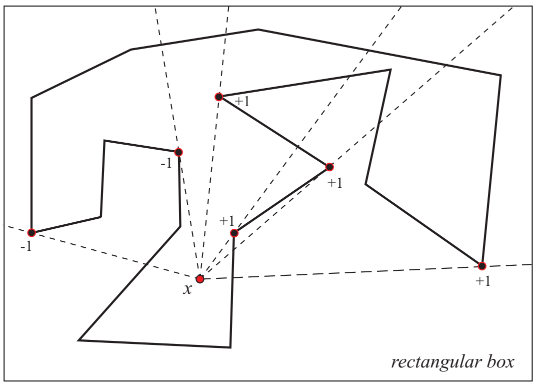



Figure 6.9: Labelled critical vertices.

2. Label all the intersection points $p_j$ identified in Step (1) with "+2" or "$-2$":

   2.1 If $p_j$ is an endpoint of an edge of $P$, that is, if $p_j = v_j$, for some $v_j \in V_P$. Then, if $v_{j-1}$ and $v_{i-1}$ are on the same side regarding the ray $\overrightarrow{xv_i}$ label $p_j$ with "$-2$", else label $p_j$ with "+2".

   2.2 If $p_j$ is a relative interior point of an edge of $P$. Then, if the source of the edge to which $p_j$ belongs and $v_{i-1}$ are on the same side regarding the ray $\overrightarrow{xv_i}$ label $p_j$ with "$-2$", else label $p_j$ with "+2". See Figure 6.10, for illustration.

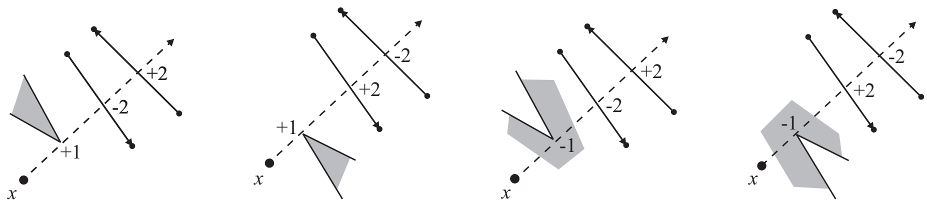



Figure 6.10: Rule to label the intersection points, which are relative interior points of edges of $P$ (shaded zones represent $int(P)$).

Figure 6.11 illustrates a polygon with the intersection points labelled according to the previous rules.

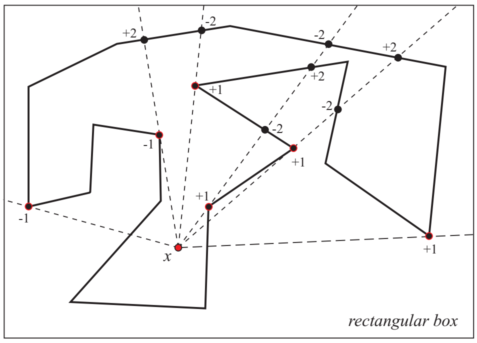



Figure 6.11: Labelled intersection points.

3. Draw the horizontal ray (to the right of $x$) and detect the first intersection point $z$ with $\partial P$. Label with 0 the edge to which $z$ belongs (from $z$ to the next vertex/point with a label), see Figure 6.12.
4. Advance by $\partial P$ in CCW direction until the next vertex/point $p$ with a label is found. Label the built segments in the following way: "label of $p$ + label of the previous segment", until $z$ is reached.

   Figure 6.13 illustrates a polygon with all the segments labelled.

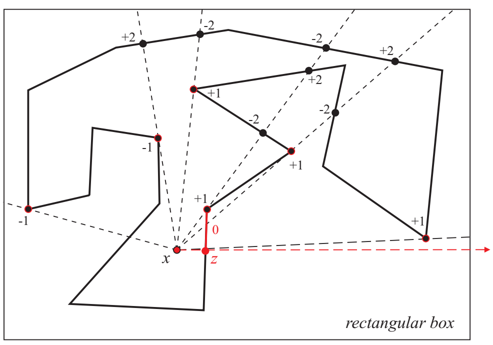



Figure 6.12: First labelled edge.

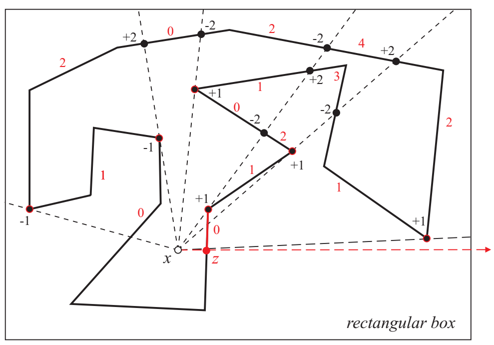



Figure 6.13: Labelled segments.

**Adaptation of the algorithm, to treat the "degenerated" case when $x$ is collinear with two or more critical vertices.**

Here it continues to be assumed that $x$ does not belong to any line passing through an edge of the polygon. This option was made because in definition 6.2 this case was not contemplated and, for now, it does not make sense to focus on it.

The differences for the previous algorithm are in Steps (1) and (2), the other steps remain equal. In the previous algorithm there is only one critical vertex on each ray, here each ray has at least one critical point (see Figure 6.14). So in Step (1), the critical vertices belonging to the same ray are ordered along the ray, according to its distance of the modem, from the nearest to the farthest. Then, from the nearest critical vertex of the modem, it is identified the intersection points of the rays with each edge of $P$ (see Figure 6.15).

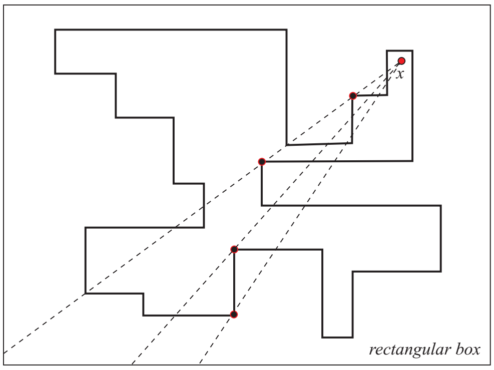



Figure 6.14: Three rays, one ray with two critical vertices and two rays with one critical vertex.

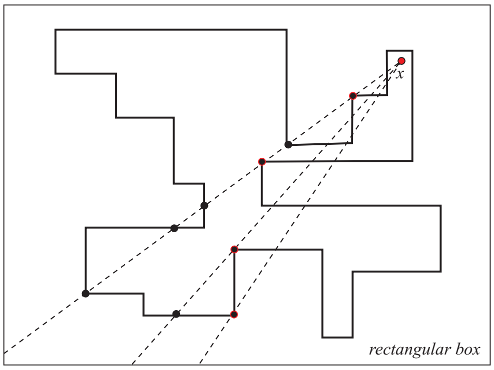



Figure 6.15: Intersection Points.

Concerning Step (2), the labelling of the critical vertices is done in the same way. The difference from the previous algorithm is in labelling the intersection points $p_j$ (not critical vertices) identified in Step (1). Here this is done as follows:

- Label all the intersection points $p_j$ identified in Step (1) with "$+n$" or "$-n$" ($n \in \mathbb{N}_0$):
    1. Initialize $n$ with 0.
    2. For each critical vertex $v_i$ nearer of the modem than $p_j$ do:
        (a) If $p_j$ is an endpoint of an edge of $P$, that is, if $p_j = v_j$, for some $v_j \in V_P$. Then "$n = n + (-2)$", if $v_{j-1}$ and $v_{i-1}$ are on the same side regarding the ray $\overrightarrow{xv_i}$; otherwise, "$n = n + (+2)$".
        (b) If $p_j$ is a relative interior point of an edge of $P$. Then "$n = n + (-2)$", if the source of the edge to which $p_j$ belongs and $v_{i-1}$ are on the same side regarding the ray $\overrightarrow{xv_i}$; otherwise, "$n = n + (+2)$". See Figure 6.16, for illustration.

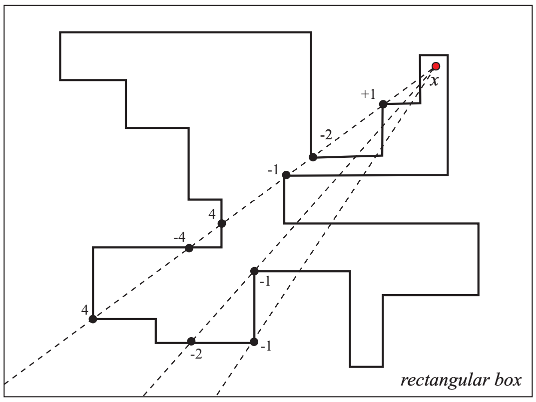
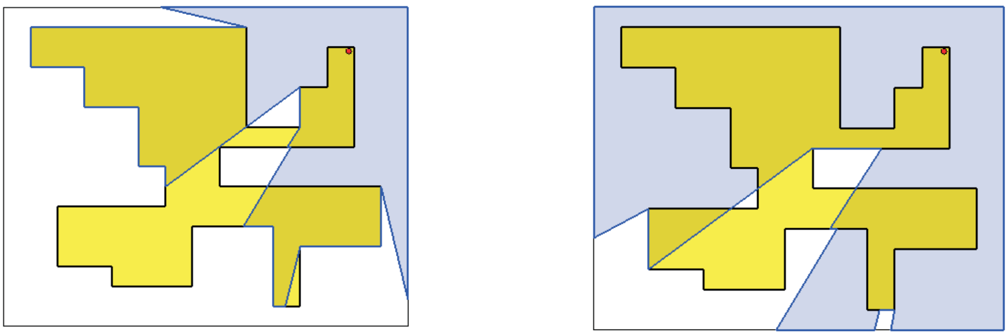



Figure 6.16: Rule to label the intersection points.

Figure 6.17 shows snapshots obtained with our software, which illustrates the region covered by a 2-modem and a 4-modem in the orthogonal polygon illustrated in the previous figures.



(a) (b)

Figure 6.17: Region covered by a k-modem in a orthogonal 30-vertex polygon: (a) $k = 2$ and (b) $k = 4$.

In the implementation of this algorithm only even values of $k$ were considered because, for now, the interest is to cover the interior of the polygon. However, for odd values the implementation can be done in a similar way. Figures 6.18, 6.19 and 6.20 show some snapshots obtained with our software.

As stated before, it is strongly believed that the MVkMS$(P, k)$ problem is $\mathcal{NP}$-hard. So, in this work the study proceeds developing an approximation method to tackle it. This method is described in the next section.

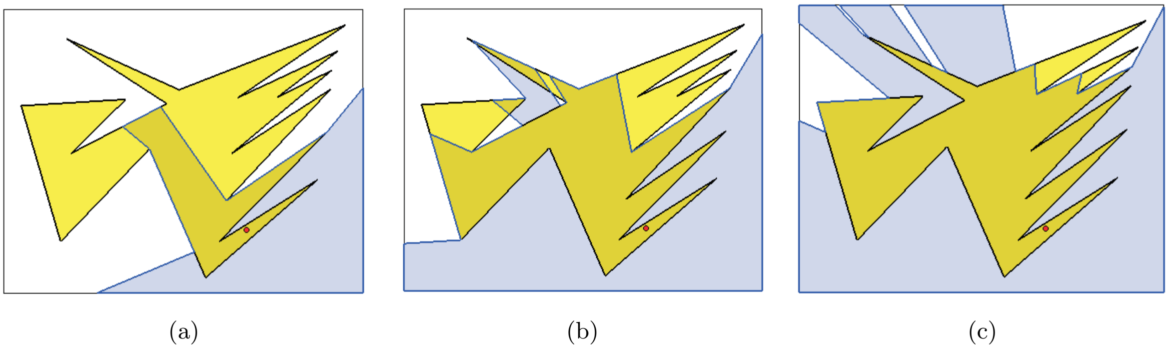



Figure 6.18: A 20-vertex arbitrary polygon $P$ and: (a) $Vis_2(x,P)$; (b) $Vis_4(x,P)$; (c) $Vis_6(x,P)$.

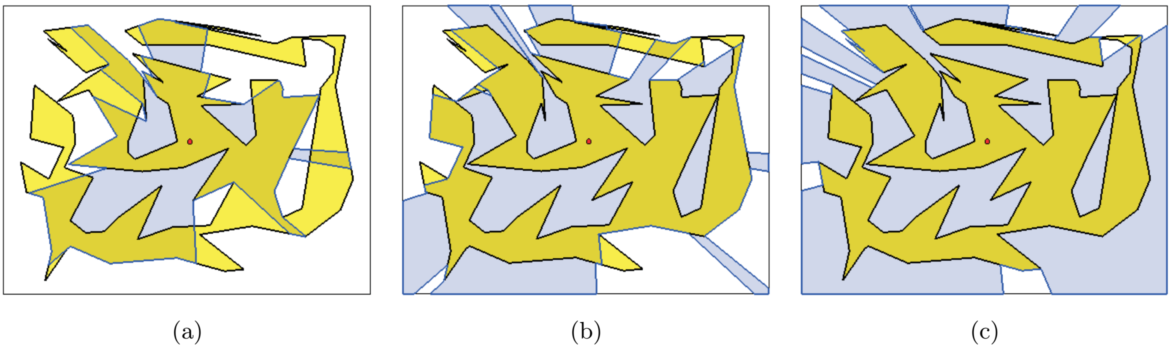



Figure 6.19: A 100-vertex arbitrary polygon $P$ and: (a) $Vis_2(x,P)$; (b) $Vis_4(x,P)$; (c) $Vis_6(x,P)$.

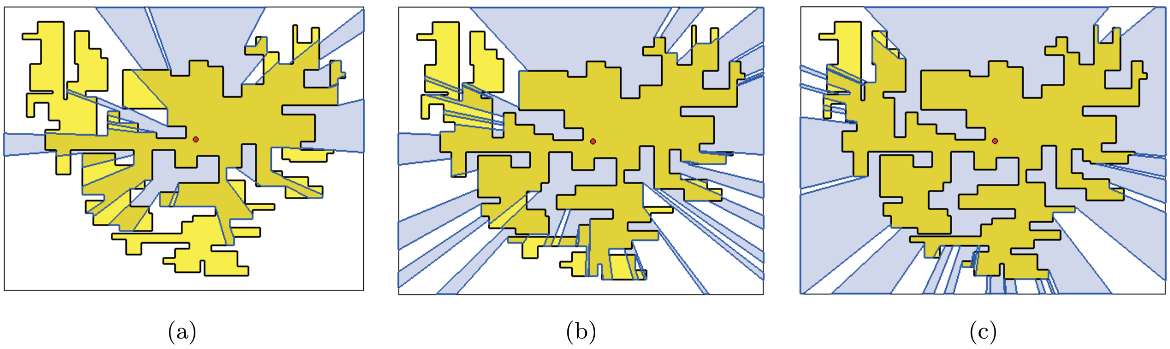



Figure 6.20: A 100-vertex arbitrary polygon $P$ and: (a) $Vis_2(x,P)$; (b) $Vis_4(x,P)$; (c) $Vis_6(x,P)$.

## 6.3 Approximation Method

Remember that, a set $G_{km}$ of vertices of $P$ is a covering vertex $k$-modem set for $P$ if $P \subset \bigcup_{v \in G_{km}} Vis_k(v,P)$ (see definition 6.7). In this work it was developed an approximation algorithm to determine a covering vertex $k$-modem set $G_{km}$, whose cardinality approximates the minimal number of vertex $k$-modems needed to cover a given polygon $P$. This approximation algorithm is an hybrid metaheuristic technique that combines the GAs and SA metaheuristics. This technique is similar to the methods $M_5$ and $M_4$ to solve the MVGP($P$) and MVFS($P$) problems, respectively (see Chapters 4 and 5). So, fundamentally it uses a GA, where, in addition to the classical operators crossover and mutation, it was added a new genetic operator based on the SA metaheuristic. Basically the process consists of applying the SA after the crossover operator and after this operation the mutation operator is applied. This strategy was chosen because it was selected as the best strategy to solve both the problem MVGS($P$) and the problem MVFS($P$).

First of all, a pre-processing step is performed to compute and store the $k$-modem visibility polygons of $v_i$, $Vis_k(v_i, P)$, for all $v_i \in V_P$. This information will decrease the algorithm runtime, since each time a vertex $k$-modem visibility polygon is required it is not necessary to calculate it again. To compute $Vis_k(v_i, P)$ it was used the algorithm described in subsection 6.2. Note that, after defining the SA and GA parameters to suit the MVkMS($P, k$) problem, a hybrid strategy is obtained that allows to get a $k$-modem vertex set $G_{km}$. However, as described in chapter 4, it may be possible that some elements of $G_{km}$ are redundant, that is, it may be possible to find a set $U \subset G_{km}$ such that $P \subset \bigcup_{v \in G_{km} \setminus U} Vis_k(v, P)$. Thus, to refine the obtained solution, the final step of the hybrid strategy is the iteratively removal of those elements.

The adaptation of the simulated annealing parameters to suit the MVkMS($P, k$) problem and the description of how the genetic algorithm parameters were defined will follow.

**Simulated Annealing** The *solution space*, set $S$, to the MVkMS($P, k$) problem is the set of all covering vertex $k$-modem sets for $P$. Thus, $S$ is a finite set and can be represented by $S = \{S_1, S_2, \ldots, S_m\}$, where $S_i = v_0^i v_1^i \ldots v_{n-1}^i$ for $i = 1, \ldots, m$. This way, each candidate solution $S_i$ is represented by a chain of length $n$, where $v_j^i$, with $j \in \{0, \ldots, n-1\}$, represents the vertex $v_j \in P$ and its value is 0 or 1. If $v_j^i = 1$ then the vertex $v_j$ is a $k$-modem; otherwise the vertex $v_j$ is not a $k$-modem. Besides, $S_i$ is a valid solution if the $k$-modems covers $P$. The *objective function* $f : S \to \mathbb{N}$ assigns to each element of $S$ the cardinality of the corresponding $k$-modem set. To generate a *neighbour* $S_j$ of $S_i = v_0^i \ldots v_{n-1}^i$ it is randomly generated a natural number, uniformly distributed, $t \in [0, n-1]$ and then if: (a) $v_t^i = 1$ then $v_t^j$ is set to 0, accepting the new solution if it is valid and rejecting it otherwise; (b) $v_t^i = 0$ then $v_t^j$ is set to 1, accepting this new solution with probability, since this is worsening the previous solution. The *initial solution* $S_0$ is the first covering vertex $k$-modem set to be analyzed. It was taken the solution obtained after applying the crossover operator in the genetic algorithm, as it will be described below. For the *initial temperature* and the *temperature decrement rule* it was considered $T_0 = \frac{n}{4}$ (value dependent on the number of vertices of the polygon) and

$T_{k+1} = \frac{T_0}{e^k}$ (very fast simulated annealing (VFSA) decrease), respectively. It was considered $N(T_k) = \lceil T_k \rceil$ *iterations for each temperature* $T_k$. Finally, the *termination condition* consists of finishing the search when the temperature is less than or equal to 0.005 or when during the last $l = 3000$ consecutive series of temperatures no new best solution is obtained and the percentage of accepted solutions is less than $\varepsilon = 2\%$.

**Genetic Algorithm** An *individual* $I$ is represented by a chain $I = m_0 m_1 \ldots m_{n-1}$, where each $m_i$ represents the vertex $v_i \in P$ and its value can be either 0 or 1. If $m_i = 1$ then $v_i$ is a $k$-modem; being $m_i = 1$, otherwise. The *population size* is the number of reflex vertices of the polygon $r$. To create the *initial population* it is considered the set of reflex vertices of $P$, $R = \{u_0, u_1, \ldots, u_{r-1}\}$, and then each of the $r$ individuals are generated as follows: $\forall i \in \{0, \ldots, r-1\}$, if placing a $k$-modem in every vertex of $R \backslash \{u_i\}$ the polygon is covered, $R \backslash \{u_i\}$ is admitted as an individual of the population; otherwise $R$ is taken as an individual. The *fitness function* is defined by $f(I) = \sum_{j=0}^{n-1} m_j$ and for the genetic operators *selection* and *crossover* it is used the tournament selection method and the variant of the single point crossover, where the generated children cannot be clones of the parents, with a probability of $p_c = 0.8$, respectively. After applying the crossover operator to two individuals, the SA strategy is applied with a probability of $p_{sa} = 0.1$. Then, with a probability of $p_m = 0.05$ the *mutation* operator is applied as follows: the value of each binary gene is flipped from zero to one or vice versa, with a probability of $p_m = 0.05$. The *evaluation of the population* is obtained by taking the lowest value obtained by the objective function $f$ in each of the individuals, finishing the algorithm when this value does not improve in 500 generations.

## 6.4 Experiments and Results

To perceive how this hybrid strategy behaves, it was implemented and it were performed several computational experiments. Note that, the implementation of this strategy implies the implementation of the algorithm to determine $Vis_k(x, P)$, $x \in P$, described in section 6.2. The computational experiments were done on a large set of randomly generated polygons. In the next two subsections, subsections 6.4.1 and 6.4.2, it will be presented the results and the conclusions from the accomplished experiments on arbitrary and orthogonal polygons, respectively.

### 6.4.1 Arbitrary Polygons

The computational experiments described in this section were performed on sets of randomly generated arbitrary and monotone arbitrary polygons, each one with 40 polygons of 30, 50, 70, 100, 110, 130, 150 and 200-vertex polygons. As previously mentioned, the arbitrary polygons were generated using the CGAL's function *random_polygon_2*. To generate the monotone arbitrary polygons it was used the algorithm developed by Snoeyink and Zhu [120].

For every set of polygons it was studied the average number of vertex $k$-modems, with $k = 2$ and $k = 4$, that the algorithm provides as a solution, as well as the average response time in seconds (pre-processing time *PP* and runtime *Time*) and the average number of iterations, *Iterations*. Tables 6.1 and 6.2 present the obtained results on arbitrary polygons (general and monotone), for $k = 2$ and $k = 4$.

| $n$ | PP (sec.) | 2-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.50 | 1.90 | 7.77 | 532.72 |
| 50 | 1.40 | 2.67 | 37.35 | 541.57 |
| 70 | 3.57 | 3.45 | 114.55 | 577.30 |
| 100 | 9.05 | 4.55 | 343.50 | 613.20 |
| 110 | 11.55 | 4.87 | 448.82 | 653.10 |
| 130 | 18.77 | 5.72 | 723.57 | 629.70 |
| 150 | 27.70 | 6.65 | 1091.50 | 613.70 |
| 200 | 63.77 | 8.35 | 2664.30 | 702.75 |

(a)

| $n$ | PP (sec.) | 2-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.60 | 2.32 | 6.40 | 527.77 |
| 50 | 1.95 | 3.17 | 26.00 | 555.70 |
| 70 | 4.87 | 4.82 | 71.97 | 561.62 |
| 100 | 14.27 | 6.92 | 189.60 | 594.90 |
| 110 | 18.20 | 7.20 | 247.80 | 587.45 |
| 130 | 31.05 | 8.87 | 417.35 | 640.17 |
| 150 | 46.30 | 10.10 | 509.62 | 605.20 |
| 200 | 104.82 | 13.37 | 1184.50 | 655.12 |

(b)

Table 6.1: Results obtained for $k = 2$ on: (a) arbitrary polygons and (b) monotone arbitrary polygons.

| $n$ | PP (sec.) | 4-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.32 | 1.07 | 4.47 | 538.65 |
| 50 | 1.27 | 1.65 | 30.97 | 517.67 |
| 70 | 3.40 | 2.00 | 114.97 | 539.62 |
| 100 | 8.95 | 2.65 | 417.45 | 582.17 |
| 110 | 11.65 | 2.82 | 550.40 | 543.82 |
| 130 | 18.75 | 3.05 | 1014.90 | 603.40 |
| 150 | 27.80 | 3.57 | 1527.00 | 572.10 |
| 200 | 63.85 | 4.37 | 4207.40 | 613.30 |

(a)

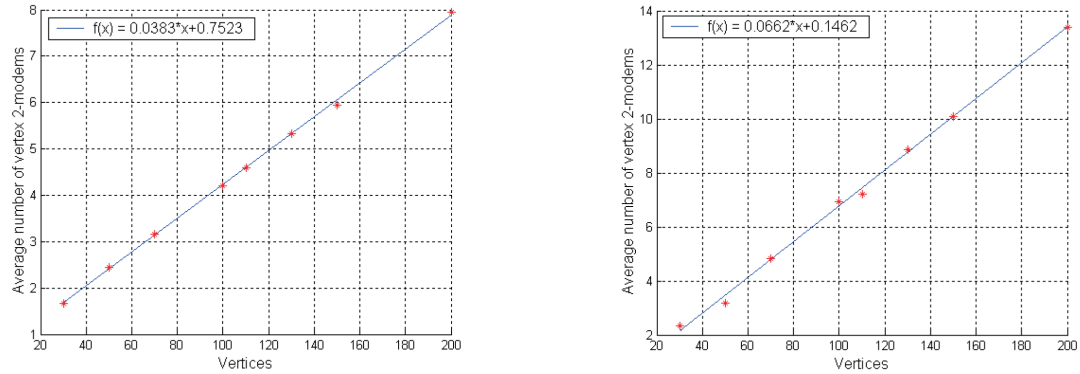
| $n$ | PP (sec.) | 4-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.52 | 1.75 | 5.87 | 530.82 |
| 50 | 1.80 | 2.15 | 20.17 | 536.00 |
| 70 | 4.87 | 3.02 | 66.27 | 525.22 |
| 100 | 14.37 | 4.00 | 184.45 | 562.95 |
| 110 | 18.17 | 4.70 | 226.65 | 551.52 |
| 130 | 31.42 | 5.45 | 425.00 | 585.90 |
| 150 | 46.30 | 6.15 | 515.95 | 581.45 |
| 200 | 105.10 | 7.85 | 1232.60 | 588.85 |

(b)

Table 6.2: Results obtained for $k = 4$ on: (a) arbitrary polygons and (b) monotone arbitrary polygons.

To infer about the average number of the minimum number of vertex 2-modems and vertex 4-modems, that are necessary to cover an arbitrary (general and monotone), it was used the least squares method. The obtained results are presented below.

**Results for 2-modems.** The linear functions that "best" fit the number of 2-modems with the number of vertices $n$ of arbitrary and monotone arbitrary polygons are $f(n) = 0.0383n + 0.7523 \approx \frac{n}{26.10} + 0.7523 \approx \frac{n}{26.10}$, with a correlation factor of 0.9988 and $f(n) = 0.0662n + 0.1462 \approx \frac{n}{15.10} + 0.1462 \approx \frac{n}{15.10}$, with a correlation factor of 0.9978 (see Figure 6.21).

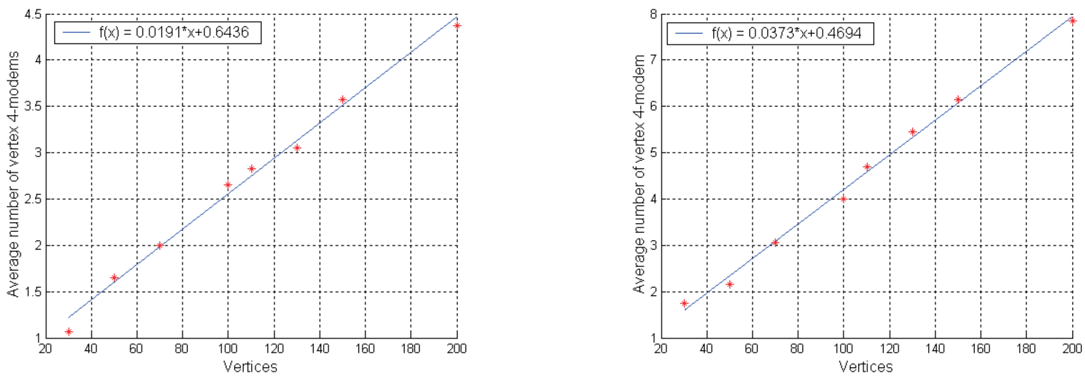



(a) (b)

Figure 6.21: Linear adjustment $k = 2$: (a) arbitrary polygons; (b) monotone arbitrary polygons.

Therefore, it can be concluded that on average, and approximately, the number of vertex 2-modems needed to cover a $n$-vertex arbitrary polygon is observed to be $\lceil \frac{n}{26.10} \rceil$. It can be also concluded that $\lceil \frac{n}{15.10} \rceil$ is the average the number of vertex 2-modems needed a $n$-vertex monotone arbitrary polygon.

**Results for 4-modems.** The curve fitted for the data presented on Table 6.2 (a) (concerning the average number of 4-modem) is $f(n) = 0.0191n + 0.6436 \approx \frac{n}{52.35} + 0.6436 \approx \frac{n}{52.35}$, with a correlation factor of 0.9926 (see Figure 6.22 (a)). The linear function that "best" fits the data presented on Table 6.2 (b) (concerning the average number of 4-modem) is $f(n) = 0.0373n + 0.4694 \approx \frac{n}{26.80} + 0.4694 \approx \frac{n}{26.80}$, with a correlation factor of 0.9951 (see Figure 6.22 (b)).




(a) (b)

Figure 6.22: Linear adjustment $k = 4$: (a) arbitrary polygons; (b) monotone arbitrary polygons.

These results allow to conclude that, approximately, the average number of 4-modems needed to cover a $n$-vertex arbitrary polygon is $\lceil \frac{n}{52.35} \rceil$. It can be also concluded that to cover a $n$-vertex arbitrary monotone polygon is $\lceil \frac{n}{26.80} \rceil$.

### 6.4.2 Orthogonal Polygons

The computational experiments described in this section were performed on sets of randomly generated orthogonal and monotone orthogonal polygons, each one with 40 polygons of 30, 50, 70, 100, 110, 130, 150 and 200-vertex polygons. As previously mentioned, the orthogonal polygons were generated using the polygon generator developed by Joseph O'Rourke. The monotone orthogonal polygons were generated using the algorithm proposed in [125]. According to this algorithm the generated polygons are placed on an $\frac{n}{2} \times \frac{n}{2}$ unit square grid and have no collinear edges (see Chapter 7, section 7.1), and that is the reason why they are designated by *grid monotone orthogonal polygons*. As for arbitrary polygons, for every set of polygons it was studied the average number of $k$-modems, with $k = 2$ and with $k = 4$, that the algorithm provides as a solution, as well as the average response time in seconds (pre-processing time *PP* and runtime *Time*) and the average number of iterations, *Iterations*. Tables 6.3 and 6.4 presents the obtained results on orthogonal polygons (general and monotone), for $k = 2$ and $k = 4$.

| $n$ | PP (sec.) | 2-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.50 | 1.67 | 6.10 | 523.07 |
| 50 | 1.80 | 2.45 | 32.25 | 559.27 |
| 70 | 4.75 | 3.15 | 101.37 | 547.65 |
| 100 | 13.02 | 4.20 | 318.17 | 618.60 |
| 110 | 16.97 | 4.60 | 409.67 | 585.37 |
| 130 | 27.05 | 5.32 | 708.17 | 604.60 |
| 150 | 41.05 | 5.95 | 1091.90 | 692.70 |
| 200 | 94.10 | 7.95 | 2541.30 | 656.45 |

(a)

| $n$ | PP (sec.) | 2-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.50 | 1.97 | 5.87 | 537.67 |
| 50 | 1.95 | 3 | 27.90 | 583.10 |
| 70 | 4.90 | 4.12 | 72.77 | 569.02 |
| 100 | 13.35 | 5.65 | 212.57 | 640.67 |
| 110 | 17.37 | 6.22 | 254.90 | 614.47 |
| 130 | 28.47 | 7.15 | 426.15 | 610.60 |
| 150 | 43.20 | 8.60 | 612.90 | 644.77 |
| 200 | 100.37 | 11.25 | 1391.00 | 695.65 |

(b)

Table 6.3: Results obtained for $k = 2$ on: (a) orthogonal polygons and (b) grid monotone orthogonal polygons.

| $n$ | PP (sec.) | 4-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.37 | 1.00 | 2.72 | 517.60 |
| 50 | 1.95 | 1.35 | 22.82 | 555.50 |
| 70 | 4.80 | 1.80 | 87.60 | 536.15 |
| 100 | 13.07 | 2.22 | 359.15 | 594.20 |
| 110 | 17.05 | 2.45 | 481.47 | 603.62 |
| 130 | 27.20 | 2.75 | 827.47 | 567.70 |
| 150 | 41.22 | 3.15 | 1535.00 | 625.77 |
| 200 | 94.50 | 3.97 | 4088.89 | 649.32 |

(a)

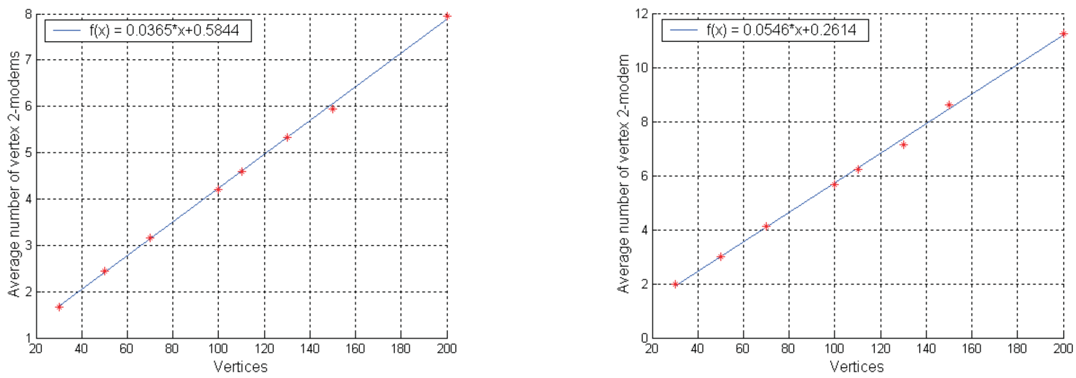
| $n$ | PP (sec.) | 4-modems | Time (sec.) | Iterations |
|---|---|---|---|---|
| 30 | 0.57 | 1.17 | 4.02 | 553.57 |
| 50 | 1.97 | 1.97 | 22.22 | 530.27 |
| 70 | 4.87 | 2.30 | 66.22 | 569.05 |
| 100 | 13.32 | 3.20 | 212.950 | 564.92 |
| 110 | 17.45 | 3.50 | 272.10 | 595.35 |
| 130 | 28.45 | 3.97 | 456.35 | 596.07 |
| 150 | 43.10 | 4.65 | 693.85 | 643.05 |
| 200 | 100.55 | 6.00 | 1468.30 | 622.40 |

(b)

Table 6.4: Results obtained for $k = 4$ on: (a) orthogonal polygons and (b) grid monotone orthogonal polygons.

Similar to arbitrary polygons, to infer about the average number of the minimum number of vertex 2-modems and vertex 4-modems that are necessary to cover an orthogonal (general and grid monotone), it was used the least squares method. The obtained results are presented below.

**Results for 2-modems.** The linear functions that "best" fits the number of vertex 2-modems with the number of vertices $n$ of orthogonal and grid monotone orthogonal polygons are $f(n) = 0.0365n + 0.5844 \approx \frac{n}{27.39} + 0.5844 \approx \frac{n}{27.39}$, with a correlation factor of 0.9988 and $f(n) = 0.0546n + 0.2614 \approx \frac{n}{18.31} + 0.2614 \approx \frac{n}{18.31}$, with a correlation factor of 0.9987 (see Figure 6.23).



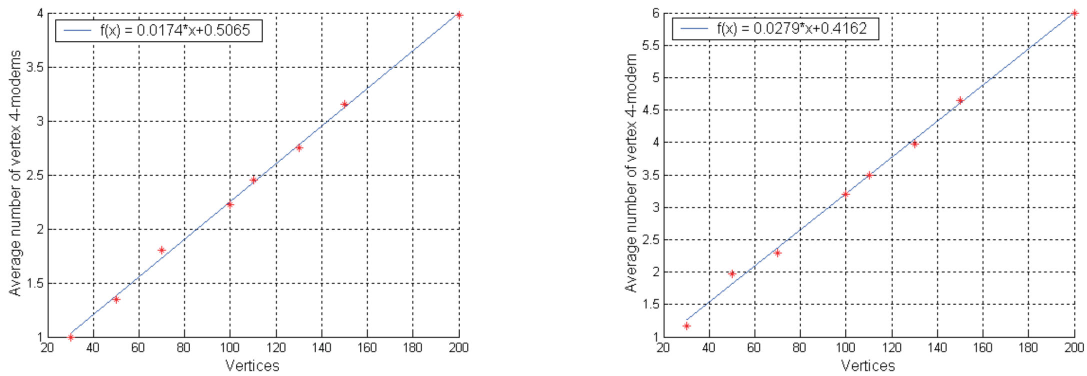

(a) (b)

Figure 6.23: Linear adjustment $k = 2$: (a) orthogonal polygons; (b) grid monotone orthogonal polygons.

Thus, it can be concluded that on average, and approximately, the number of vertex 2-modems needed to cover a $n$-vertex orthogonal polygon is observed to be $\lceil \frac{n}{27.39} \rceil$. It can be also concluded that $\lceil \frac{n}{18.31} \rceil$ is the average the number of vertex 2-modems needed a $n$-vertex grid monotone $\lceil \frac{n}{18.31} \rceil$ polygon.

**Results for 4-modems.** The curve fitted to the data of Table 6.4 (a) (concerning the average number of 4-modem) is $f(n) = 0.0174n + 0.5065 \approx \frac{n}{57.47} + 0.5065 \approx \frac{n}{57.47}$, with a correlation factor of 0.9985 (see Figure 6.24 (a)). The linear function that "best" fits the data presented on Table 6.4 (b) (concerning the average number of 4-modem)is $f(n) = 0.0279n + 0.4162 \approx \frac{n}{35.84} + 0.4162 \approx \frac{n}{35.84}$, with a correlation factor of 0.9973 (see Figure 6.24 (b)).

These results allow to conclude that, approximately, the average number of 4-modems needed to cover a $n$-vertex orthogonal polygon is $\lceil \frac{n}{57.47} \rceil$. It can be also concluded that to cover a $n$-vertex grid monotone orthogonal polygon is $\lceil \frac{n}{35.84} \rceil$ if the polygon is grid monotone.




(a) (b)

Figure 6.24: Linear adjustment $k = 4$: (a) orthogonal polygons; (b) grid monotone orthogonal polygons.

## 6.5 Concluding Remarks

The computational experiments showed that, approximately and on average, the number 2-modems needed to cover an arbitrary polygon $P$ with $n$ vertices was observed to be $\lfloor \frac{n}{26.10} \rfloor$, being $\lceil \frac{n}{15.10} \rceil$ if $P$ is monotone. If the polygons are orthogonal or grid monotone orthogonal, the obtained values are $\lceil \frac{n}{27.39} \rceil$ and $\lceil \frac{n}{18.31} \rceil$, respectively. Concerning, the number of 4-modems needed to cover a polygon $P$ with $n$ vertices, the results show that , approximately and on average, the number 2-modems needed to cover an arbitrary polygon $P$ with $n$ edges was observed to be $\lceil \frac{n}{52.35} \rceil$, being $\lceil \frac{n}{26.80} \rceil$ if $P$ is monotone. If the polygons are orthogonal or grid monotone orthogonal, the obtained values are $\lceil \frac{n}{57.47} \rceil$ and $\lceil \frac{n}{35.84} \rceil$, respectively. As it can be observed, the obtained values for monotone polygons are much less than the theoretical bounds $\lceil \frac{n}{2k} \rceil$ and $\lceil \frac{n-2}{2k+4} \rceil$ for monotone arbitrary and monotone orthogonal polygons, respectively.

For this problem it was not explored the value of the various parameters associated with to the used metaheuristics. As future work it is intended not only to study these parameters, but also to use different metaheuristics. It also intended to develop a method that allows to obtain the approximation ration of the developed algorithms, since it is strongly believed that this problem is $\mathcal{NP}$-hard.

# Part II

# Visibility Problems on Special Classes of Polygons

## Introduction

Since many of the visibility problems are $\mathcal{NP}$-hard or they are strongly supposed to be $\mathcal{NP}$-hard, the Part I of this dissertation is devoted to the study of approximation algorithms to deal with them. However, it is also important to identify classes of polygons for which it is possible to determine exact solutions and/or combinatorial bounds. In this sense it is intended to study the Minimum Vertex Guard Set, MVGS($P$), Maximum Hidden Set, MHS($P$), and Maximum Hidden Vertex Set, MHVS($P$), problems where some of these classes were identified.

Thus, the Part II of the dissertation addresses the determination of exact solutions and/or combinatorial bounds on special classes of polygons, in this way, following the second line of investigation proposed in Chapter 1. This part is divided in two chapters. In the first chapter the MVGS($P$) and the MHVS($P$) problems are studied on a subclass of orthogonal polygons, the grid $n$-ogons. In this chapter, in order to simplify the study of these problems firstly it is studied some structural properties of the grid $n$-ogons. In the second chapter the MHVS($P$) and the MHS($P$) problems are studied on spiral and histogram polygons.

# Chapter 7

# A Subclass of Orthogonal Polygons: the grid $n$-ogons

In this chapter a subclass of orthogonal polygons, the grid $n$-ogons, is studied. These polygons were defined by Tomás and Bajuelos [24,124] and they appear to exhibit sufficiently interesting characteristics that are studied and formalized. Besides, they were used experimentally to evaluate some approximated methods of resolution of some illumination problems [36,37,126]. In this chapter, the first vertex of a polygon is denoted by $v_1$ instead of $v_0$ to be congruent with the notation used by Tomás and Bajuelos in [24, 124].

This chapter is divided in four sections. In the first one, section 7.1, some definitions and already known results related to grid $n$-ogons are briefly presented (for more details refer to [24,124]). Section 7.2 is devoted to the study of new results related to this class of polygons, mainly related to structural properties. In section 7.3, for the Fat, Min-Area and Spiral grid $n$-ogons, special subclasses of grid $n$-ogons, the optimal solution of the following problems is determined: MVGS($P$), where $P$ is a Fat grid $n$-ogon, a Min-Area grid $n$-ogon and a Spiral grid $n$-ogon and MHVS($P$), where $P$ is a Thin grid $n$-ogon. Finally, in section 7.4, some conclusions and open problems are established.

Let us mention that some of the results appearing in this chapter have been published in [16, 18, 19, 91–94].

## 7.1 Conventions, Definitions and Results

Remember that, for every $n$-vertex orthogonal polygon ($n$-*ogon*, for short), $n = 2r + 4$, where $r$ denotes the number of reflex vertices, e.g. [101]. So, orthogonal polygons have an even number of vertices.

**Definition 7.1** *A partition of a polygon $P$ is a division of $P$ into sub-polygons* (*named pieces*) *that do not overlap except on their boundaries.*

**Definition 7.2** *A* ***rectilinear cut r-cut*** *of a $n$-ogon $P$ is obtained by extending each edge incident on a reflex vertex of $P$ towards $int(P)$ until it hits $\partial P$. By drawing all r-cuts, $P$ is partitioned into rectangles, called* ***r-pieces****. This partition is denoted by $\Pi(P)$ and the number of its elements (pieces) as $|\Pi(P)|$ (see Figure 7.1).*

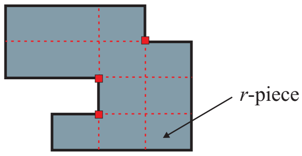



Figure 7.1: A $n$-ogon $P$ and its $\Pi(P)$ partition.

**Definition 7.3** *A $n$-ogon $P$ is in* ***general position*** *if it has no collinear edges. A $n$-ogon in general position defined in a $(\frac{n}{2}) \times (\frac{n}{2})$ square grid is called* ***grid $n$-ogon****.*

It is assumed that the grid is defined by the horizontal lines $y = 1, \ldots, y = \frac{n}{2}$ and the vertical lines $x = 1, \ldots, x = \frac{n}{2}$ and that its northwest corner is $(1, 1)$. Each grid $n$-ogon has exactly one edge in every line of the grid. A correct and complete method to generate grid $n$-ogons, well described in [124] and briefly explained here, is the INFLATE-PASTE.

Let $v_i = (x_i, y_i)$, for $i = 1, \ldots, n$, be the vertices of a grid $n$-ogon $P$, in CCW order.

### Inflate

Inflate takes $P$ and a pair of integers with $(p, q)$ with $p, q \in \{0, 1, \ldots, \frac{n}{2}\}$, and yields a new $n$-ogon $\widetilde{P}$ with vertices $\widetilde{v}_i = (\widetilde{x}_i, \widetilde{y}_i)$ given by $\widetilde{x}_i = x_i$, if $x_i \leq p$ and $\widetilde{x}_i = x_i + 1$, if $x_i > p$; and $\widetilde{y}_i = x_i$, if $y_i \leq q$ and $\widetilde{y}_i = y_i + 1$, if $y_i > q$ , for $i = 1, \ldots, n$. INFLATE augments the grid creating two free lines, namely $x = p + 1$ and $y = q + 1$.

### Inflate-Paste

First imagine $P$ merged in a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid, with top, bottom, leftmost, and rightmost grid free lines. The top-line is now $x = 0$ and the leftmost is $y = 0$. Now, the northwest corner of this extended grid is the point $(0, 0)$. Let $e_h(v_i)$ be the horizontal edge of $P$ to which $v_i$ belongs.

**Definition 7.4** *Given a grid $n$-ogon merged into a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid and a convex vertex $v_i$ of $P$, the* ***free staircase neighborhood of $v_i$****, denoted by $\mathrm{FSN}(v_i)$, is the largest staircase polygon in this grid that has $v_i$ as vertex, does not intersect the interior of $P$ and its base edge contains $e_H(v_i)$, the horizontal edge of $P$ to which $v_i$ belongs (see Figure 7.2).*

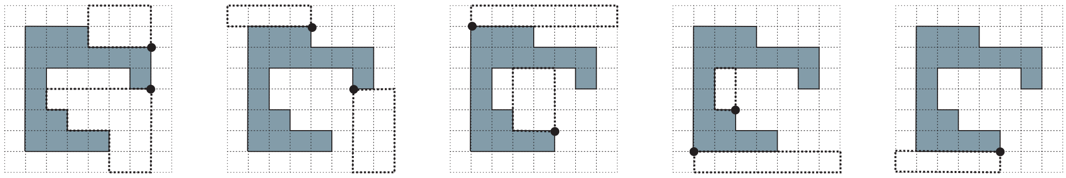


Figure 7.2: A grid $n$-ogon merged into a $(\frac{n}{2}+2)\times(\frac{n}{2}+2)$ square grid and the free staircase neighborhood for each of its convex vertices [124].

To transform $P$ by INFLATE-PASTE, first take a convex vertex $v_i$ of $P$, then select a cell $C$ in $FSN(v_i)$, with center $c$ and northwest corner $(p,q)$, and apply INFLATE to $P$ using $(p,q)$. As stated before, the center of $C$ is mapped to $\widetilde{c}=(p+1,q+1)$, which will be now a convex vertex of the new polygon. PASTE glues the rectangle defined by $\widetilde{v_i}$ and $\widetilde{c}$ to $\widetilde{P}$, increasing the number of vertices by two. If $e_H(v_i)=\overline{v_iv_{i+1}}$ then PASTE removes $\widetilde{v_i}=(\widetilde{x_i},\widetilde{y_i})$ and inserts the chain $(\widetilde{x_i},q+1),\widetilde{c},(p+1,\widetilde{y_i})$. If $e_H(v_i)=\overline{v_{i-1}v_i}$, PASTE replaces $\widetilde{v_i}$ by the chain $(p+1,\widetilde{y_i}),\widetilde{c},(\widetilde{x_i},q+1)$. Figure 7.3 illustrates this transformation.

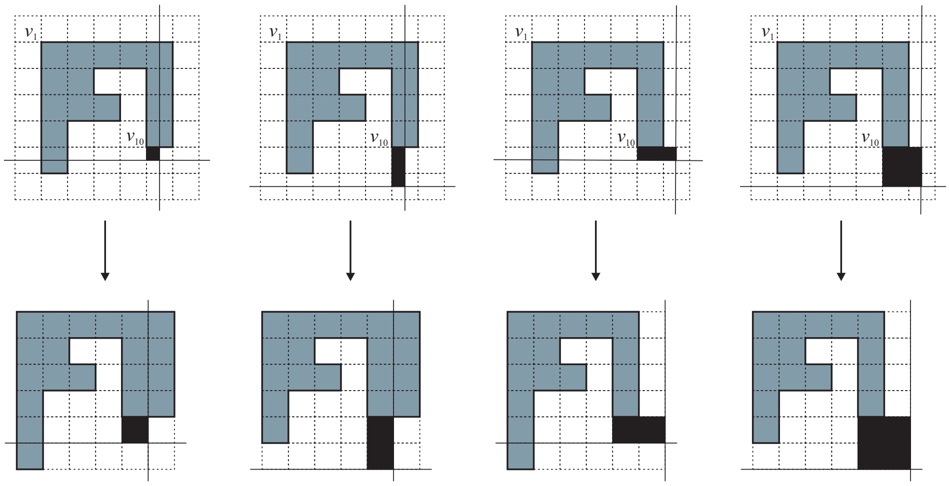
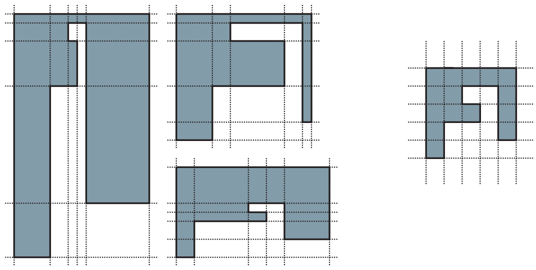



Figure 7.3: The four grid 14-ogon that may be constructed if INFLATE-PASTE is applied to the given 12-ogon, extending the vertical edge that ends at vertex $v_{10}$ [124].

Note that, each $n$-ogon in general position is mapped to an unique grid $n$-ogon through top-to-bottom and left-to-right sweeping. And, reciprocally, given a grid $n$-ogon a $n$-ogon may be created that is an instance of its class by randomly spacing the grid lines in such a way that their relative order is kept (see Figure 7.4). Each $n$-ogon that is not in general position may be mapped to a $n$-ogon in general position by $\epsilon$-perturbations, for a sufficiently small $\epsilon>0$ constant. Thus, $n$-ogons in general position are addressed in a first approach.



(a) (b)

Figure 7.4: The three 12-ogons on the left are mapped in the grid 12-ogon on the right. And the three 12-ogons on the left can be obtained from the grid 12-ogon on the right [124].

Grid $n$-ogons that are symmetrically equivalent are grouped, in the same classes. In this way, the grid $n$-ogons in Figure 7.5 represent the same class.

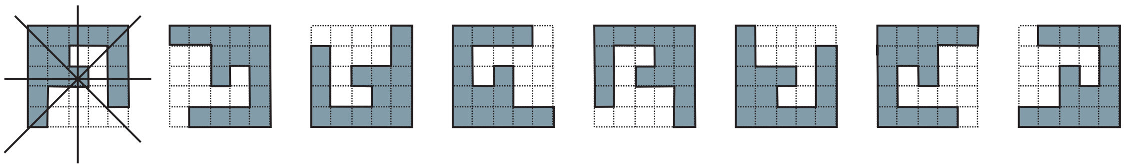


Figure 7.5: Eight grid$n$-ogons that are symmetrically equivalent. From left to right, we see images by clockwise rotations of 90°, 180° and 270°, by flips wrt horizontal and vertical axes and flips wrt positive and negative diagonals [124].

Given a $n$-ogon $P$ in general position, FREE($P$) represents any grid $n$-ogon in the class that contains the grid $n$-ogon to which $P$ is mapped by the sweep procedure. For all $n$-ogons $P$ in general position, $|\Pi(P)| = |\Pi(\text{FREE}(P))|$.

**Definition 7.5** *A grid $n$-ogon $Q$ is called* **Fat** *if, and only if, $|\Pi(Q)| \geq |\Pi(P)|$, for all grid $n$-ogons $P$. Similarly, a grid $n$-ogon $Q$ is called* **Thin** *if, and only if, $|\Pi(Q)| \leq |\Pi(P)|$, for all grid $n$-ogons $P$.*

Let $P$ be a grid $n$-ogon and $r$ the number of its reflex vertices. In [24] it was proven that, if $P$ is FAT then

$$|\Pi(P)| = \begin{cases} \frac{3r^2+6r+4}{4} & \text{for } r \text{ even} \\ \\ \frac{3(r+1)^2}{4} & \text{for } r \text{ odd} \end{cases}$$

and if $P$ is THIN then $|\Pi(P)| = 2r + 1$.

There is a single FAT $n$-ogon (except for symmetries of the grid) and its form is illustrated in Figure 7.6. However, the THIN $n$-ogons are not unique (see Figure 7.7).

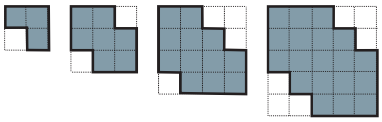


Figure 7.6: The unique FAT $n$-ogons, for $n = 6, 8, 10$ and 12.

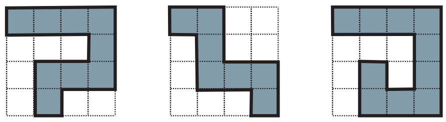


Figure 7.7: Three THIN 10-ogons.

The area of a grid $n$-ogon $P$ is the number of grid cells in its interior and is denoted by $A(P)$. In [24] it was proven that for all grid $n$-ogon $P$, with $n \geq 8$, $2r + 1 \leq A(P) \leq r^2 + 3$.

**Definition 7.6** *A grid $n$-ogon $P$ is denoted by* **Max-Area** *grid $n$-ogon if, and only if, $A(P) = r^2 + 3$ and it is called a* **Min-Area** *grid $n$-ogon if, and only if, $A(P) = 2r + 1$.*

In [24] it has been shown that there are MAX-AREA grid $n$-ogons for all $n$, but they are not unique (see Figures 7.8 and 7.9). However, there is a single MIN-AREA grid $n$-ogon (except for symmetries of the grid) and its form is illustrated in Figure 7.10.

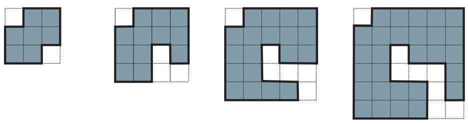


Figure 7.8: A family of grid $n$-ogons with MAX-AREA, for $r = 2, 3, 4$ and 5.

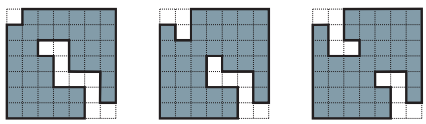


Figure 7.9: A sequence of MAX-AREA $n$-ogons, for $r = 6$.

As we can see in Figure 7.11 the FAT $n$-ogons are not the MAX-AREA. Regarding MIN-AREA $n$-ogons, it is obvious that they are THIN grid $n$-ogons, because $|\Pi(P)| = 2r + 1$ holds only for THIN grid $n$-ogons. However, this condition is not sufficient for a grid $n$-ogon to be a MIN-AREA grid $n$-ogon, i.e., not all the THIN grid $n$-ogons are the MIN-AREA grid $n$-ogon, as we can see in Figure 7.12.

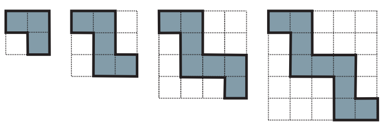


Figure 7.10: The unique grid MIN-AREA grid $n$-ogons, for $r = 1, 2, 3$ and 4.

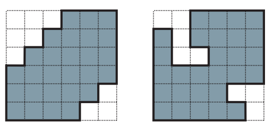


Figure 7.11: On the left is the FAT grid 14-ogon, it has area 27. On the right is a 14-ogon with area 28, which is the maximum for $n = 14$.

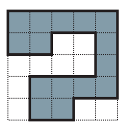


Figure 7.12: THIN grid 12-ogon with area 15, where the area of the MIN-AREA grid 12-ogon is equal to 9.

## 7.2 More Results on grid $n$-ogons

Given a $n$-ogon $P$, the partition $\Pi(P)$ is already defined. A different partition can be obtained by extending each horizontal edge incident on a reflex vertex of $P$ towards $int(P)$ until it hits $\partial P$, this partition is denoted by $\Pi_H(P)$. Both partitions decompose $P$ into rectangles (see Figure 7.13).

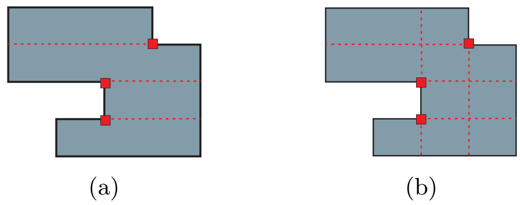



Figure 7.13: A $n$-ogon $P$ and: (a) partition $\Pi_H(P)$; (b) partition $\Pi(P)$.

To each partition of a polygon can be associated a graph, designated by *dual graph* of the partition, which captures the adjacency relation between the partition pieces. The nodes of the dual graph represent the pieces of the partition and two nodes are adjacent if their corresponding pieces share a line segment in their common boundary. The dual graph of $\Pi(P)$ is denoted by $G_{\Pi(P)}$ and the dual graph of $\Pi_H(P)$ by $G_{\Pi_H(P)}$ (see Figure 7.14).

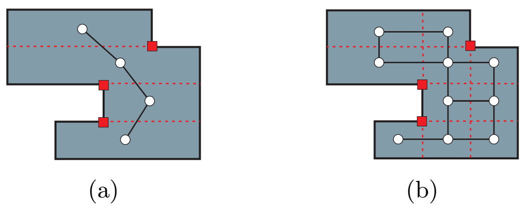



Figure 7.14: A $n$-ogon $P$ and: (a) $G_{\Pi_H(P)}$; (b) $G_{\Pi(P)}$.

Some necessary definitions, for this work, related to graphs will follow. Let $G = (V, E)$ be a graph. The number of nodes adjacent to a node $v \in V$ is called the *degree* of node $v$. A sequence of nodes $v_1, \ldots, v_k$, $k \geq 2$, is called a *walk* if $v_i v_{i+1} \in E$ for $i = 1, \ldots, k-1$. A walk $v_1, \ldots, v_k$ is said to be *closed* if $v_1 = v_k$. A closed walk is said to be a *cycle* if $k \geq 3$ and $v_1, \ldots, v_k$ is a walk with no node repetitions. A graph is called *connected* if for every pair of nodes $u$ and $v$ of $V$, there is a walk with no node repetitions $u = v_1, \ldots, v_k = v$; otherwise it is called *disconnected*. A graph is called a *tree* if it is connected and contains no cycles [99]. A graph is designated by *path graph* if it is a tree with two nodes of degree 1, called leaves, and the other nodes of degree 2. In [24] it was proved that $G_{\Pi_H(P)}$ is a tree, here it will be proved that if $P$ is a THIN grid $n$-ogon then $G_{\Pi(P)}$ is a path graph.

In $\Pi(P)$, each $r$-piece is defined by four vertices. Each vertex is either on $\partial P$ (*boundary* *vertex*) or in $int(P)$ (*internal vertex*). Similar definitions hold for the edges. An edge $e$ of a $r$-piece $R$ is called a *boundary edge* if $e \cap int(P) = \emptyset$; and it is called an *internal edge* if $e \cap int(P) \neq \emptyset$ and the only points of $e$ that could not be in $int(P)$ are its endpoints [24]. The total number of internal vertices of $\Pi(P)$ is denoted by $|V_i(P)|$. For all $P$, Thin grid $n$-ogon, $|V_i(P)| = 0$ [24].

**Lemma 7.1** *Let $P$ be a* Thin *$(n+2)$-ogon. Then every grid $n$-ogon that yields $P$ by* Inflate-Paste *is also a* Thin*.*

*Proof:* Suppose that there is a grid $n$-ogon $Q$ not Thin that yields $P$ by Inflate-Paste. Since $Q$ is not a Thin grid $n$-ogon, we have $|V_i(Q)| > 0$. The application of Inflate to $Q$ does not change $|V_i(Q)|$, because this operation does not add any reflex vertex and the relative position of the remainder is maintained. However, Paste adds a reflex vertex, $v_r$, to $Q$. In this way the number of interior vertices increases or is maintained. It is maintained if the extensions of the incident edges at $v_r$ do not intersect any $r$-cut of $Q$, and increases otherwise (see Figure 7.15).

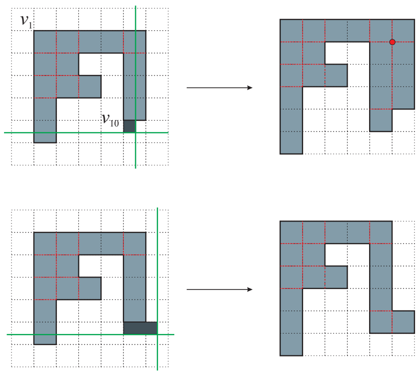



Figure 7.15: On the right we can see two grid 14-ogons that can result from the application of Inflate-Paste to the 12-ogon on the left, extending the vertical edge that ends at vertex $v_{10}$. In the top-right polygon the number of internal vertices increases in one unit and in the bottom-right polygon this number is maintained.

Thus, any grid $(n+2)$-ogon $P_1$ that is obtained from $Q$, by Inflate-Paste, will verify $|V_i(P_1)| \geq |V_i(Q)| > 0$. Therefore, $|V_i(P)| > 0$, in contradiction to the fact of $P$ being Thin.

□

**Proposition 7.1** *Let $P$ be a* Thin *grid $n$-ogon with $r = \frac{n-4}{2} \geq 1$ reflex vertices, then $G_{\Pi(P)}$ is a path graph (see examples in Figure 7.16).*

*Proof:* The demonstration will be done by induction on $r$.

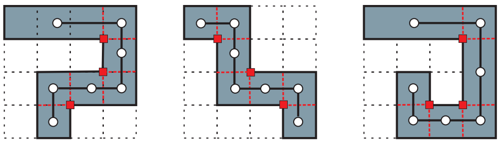


Figure 7.16: Three Thin grid 10-ogon and respective dual graphs.

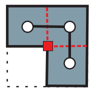


Figure 7.17: 6-ogon $P$, $\Pi(P)$ and $G_{\Pi(P)}$.

**Base Case, $r = 1$:** It can easily be checked that the proposition is true for $r = 1$ (see Figure 7.17).

**Inductive Step:** Let, $r \geq 1$. Assuming that the result is true for Thins with $r$ reflex vertices, it will be proven that it is, also, true for Thins with $r + 1$ reflex vertices.

Consider any leaf, $F$, of the tree $G_{\Pi_H(P)}$. In [24] it is proved that each leaf of this tree corresponds to a rectangle that could have been glued by Paste to yields $P$. Thus, let $R$ be the rectangle of $\Pi_H(P)$ that corresponds to $F$ and $\overline{vs_v}$ the chord that separates $R$ from the rest of $P$. $R$ can be one of two types: *Type 1* and *Type 2*, illustrated in the Figure 7.18 (see [24]). The vertex that is not adjacent to $s_v$ is $\widetilde{c} = (p + 1, q + 1)$, in Inflate-Paste, and $s_v$ is $\widetilde{v}_i = (\widetilde{x}_i, \widetilde{y}_i)$.

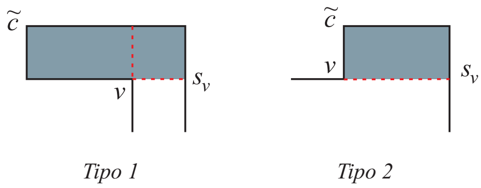



Figure 7.18: The two possible types of rectangles (shaded) that correspond to leaves in $G_{\Pi_H(P)}$, being $P$ a grid $n$-ogon.

**Case 1:** $R$ is of Type 1.

In this case, $R = R_1 \cup R_2$, where $R_1$ and $R_2$ are two adjacent $r$-pieces of $\Pi(P)$. If we remove $R$ we will obtain a $n$-ogon in general position $Q$ that, by lemma 7.1, is an "inflated" Thin $n$-ogon with $r$ reflex vertices. Thus, by induction hypothesis and Observation 1, $G_{\Pi(Q)}$ is a path graph.

**Observation 1:** If $Q$ is an "inflated" $n$-ogon then $G_{\Pi(Q)}$ and $G_{\Pi(P)}$ have the same "structure", where $P$ is the grid $n$-ogon that yields $Q$, given that the relative position of the vertices is maintained after Inflate.

Besides, $\overline{vs_v}$ belongs to a unique $r$-piece, $R_3$, of $\Pi(Q)$. In fact, suppose that $\overline{vs_v}$ belongs to more than a $r$-piece. Then it exists, at least, a point $p \in \overline{vs_v}$ that belongs to a vertical chord (extension of a vertical edge) in $Q$ (see Figure 7.19 (a)). Therefore, $p \in int(P)$, so we can conclude that $p \in V_i(P) \Rightarrow |V_i(P)| \neq 0$, in contradiction to the fact of $P$ being THIN grid $n$-ogon. Thus, $R_3$ has three boundary edges and one interior edge (see Figure 7.19 (b)). Consequently, the node $n_3 \in G_{\Pi(Q)}$, that corresponds to $R_3$, is a leaf.

Now, consider a path graph with two nodes $n_1$ and $n_2$, corresponding to $R_1$ and $R_2$, respectively. Connecting this graph to $G_{\Pi(Q)}$, through an edge that joins $n_2$ and $n_3$, we will obtain $G_{\Pi(P)}$, which is a path graph (see Figure 7.19 (c)).

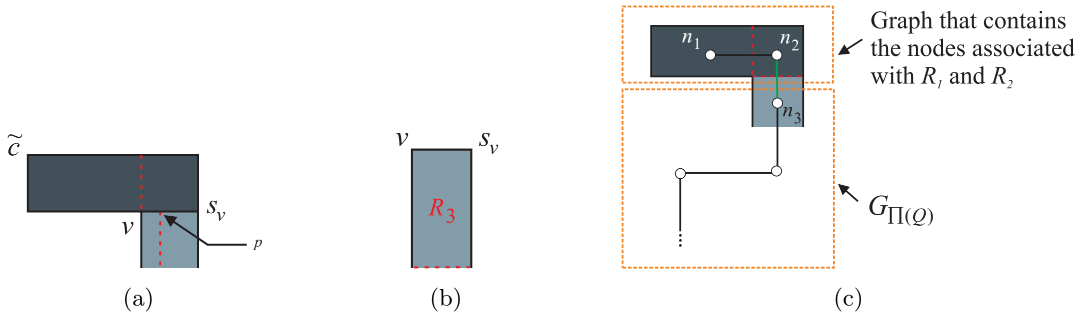



Figure 7.19: (a) $p \in \overline{vs_v}$; (b) $R_3$ with three boundary edges and one interior edge and (c) Construction of $G_{\Pi(P)}$.

**Case 2:** $R$ is of Type 2.

In this case, $R = R_1$, where $R_1$ is a $r$-piece of $\Pi(P)$ (see Figure 7.20 (a)). As in the previous case, if we remove $R$ we will obtain a $n$-ogon in general position $Q$ that, by lemma 7.1, is an "inflated" THIN $n$-ogon with $r$ reflex vertices. Thus, by induction hypothesis and Observation 1, $G_{\Pi(Q)}$ is a path graph. Note that $v$ is not a vertex of $Q$, moreover $v$ belongs to the interior of the edge $\overline{\widetilde{v}_i\widetilde{v}_{i+1}}$ (see Figure 7.20 (b)).

As in the previous case, we can prove that $\overline{vs_v}$ belongs to a unique $r$-piece $R_2 \in \Pi(Q)$. Besides, $v$ is not a vertex of $R_2$. In fact, if it was a vertex of $R_2$, then $P$ would have two collinear edges, in contradiction to the fact of $P$ being in general position (see Figure 7.20(c)).

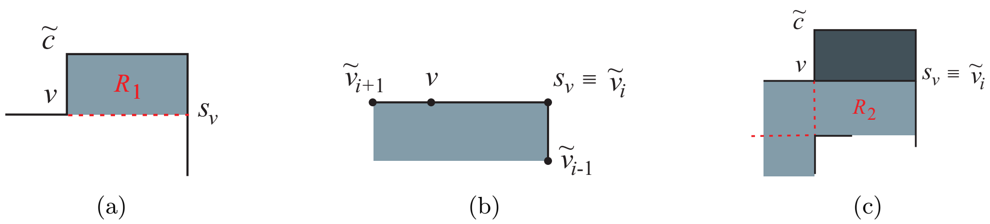



Figure 7.20: (a) $R = R_1$ is of Type 2; (b) Removal of $R = R_1$ and (c) $v$ is not a vertex of $R_2$.

So, $\overline{vs_v}$ is strictly contained in an edge of $R_2$. Denote by $e_{V_1}$ and $e_{H_1}$ the vertical and horizontal edges of $R_2$, incident on $s_v$, respectively; and denote by $e_{V_2}$ and $e_{H_2}$, the vertical and horizontal edges of $R_2$ not incident on $s_v$, respectively.

We know that $e_{V_1}$ and $e_{H_1}$ are boundary edges, so there are four hypotheses for $e_{V_2}$ and $e_{H_2}$:

a) $e_{V_2}$ is an interior edge and $e_{H_2}$ is a boundary edge (see Figure 7.21 (a));

b) $e_{V_2}$ and $e_{H_2}$ are interior edges (see Figure 7.21 (b));

c) $e_{V_2}$ is a boundary edge and $e_{H_2}$ is an interior edge (see Figure 7.21 (c));

d) $e_{V_2}$ and $e_{H_2}$ boundary edges (see Figure 7.21 (d));

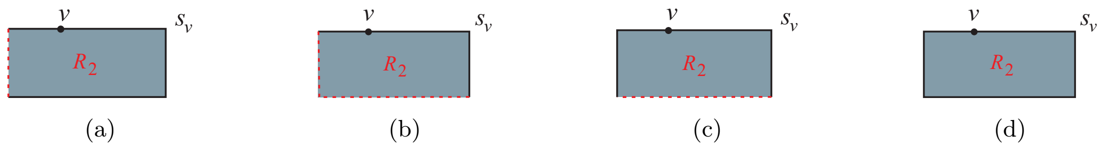



Figure 7.21: Four hypotheses to the edges of $R_2$.

Notice that the case b) cannot take place. In fact, suppose that case b) take place, then there would be a point $p \in V_i(P) \Rightarrow |V_i(P)| \neq 0$, in contradiction to the fact of $P$ being a THIN (see Figure 7.22 (a)). In an analogous way we can show that c) cannot take place. Case d) also cannot take place, since in this case we would have $r = 0$, given that $r \geq 1$, it would come $0 \geq 1$, absurdity!

As a result, only case a) can happen. Being so, the node $n_2 \in G_{\Pi(Q)}$, that corresponds to $R_2$, is a leaf (see Figure 7.22 (b)).


Figure 7.22: (a) Case b) cannot take place; (b) $R_2$ corresponds to a leaf in $G_{\Pi(Q)}$.

Therefore, the operation PASTE splits $R_2$ into adjacent $r$-pieces $R_2^{(1)}$ and $R_2^{(2)}$, i.e., $R_2 = R_2^{(1)} \cup R_2^{(2)}$ and it adds $R_1$ to $Q$ to give rise $P$ (see Figure 7.23 (a)).

Now, consider a path graph $C_3$ with three nodes $n_1$, $n_2^{(1)}$ and $n_2^{(2)}$, corresponding to $R_1$, $R_2^{(1)}$ and $R_2^{(2)}$, respectively. In other words, consider $C_3 = \{V, E\}$, where $V = \{n_1, n_2^{(1)}, n_2^{(2)}\}$ and $E = \{n_1 n_2^{(1)}, n_2^{(2)} n_2^{(1)}\}$. In $G_{\Pi(Q)}$ remove $n_2$. Connect $C_3$ to $G_{\Pi(Q)} \backslash \{n_2\}$, through an edge that joins the adjacent node to $n_2$ in $G_{\Pi(Q)}$ to $n_2^{(2)}$. We will obtain $G_{\Pi(P)}$, which is a path graph (see Figure 7.23 (b)).

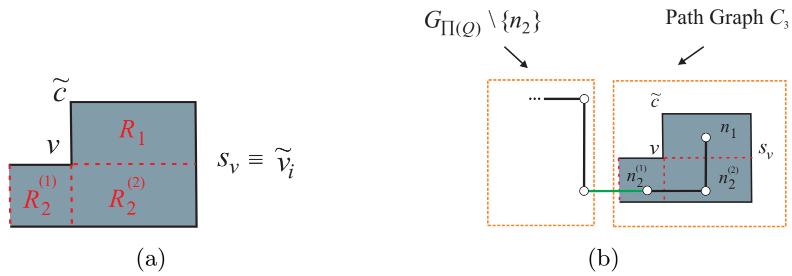



Figure 7.23: (a) PASTE operation; (b) Construction of $G_{\Pi(P)}$.

In both Cases, 1 and 2, we showed that $G_{\Pi(P)}$ is a path graph.

□

**Proposition 7.2** *Let $P$ be a grid $n$-ogon, with $n > 6$. If $P$ is not* THIN *then $G_{\Pi(P)}$ is not a tree (see example in Figure 7.24).*

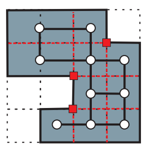


Figure 7.24: A grid 10-ogon and respective dual graph.

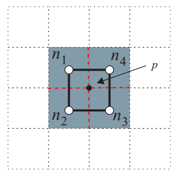



Figure 7.25: Subgraph of $G_{\Pi(P)}$.

*Proof:* If $P$ is a non THIN grid $n$-ogon then it exists at least one point $p \in V_i(P)$, i.e., one internal vertex of $\Pi(P)$. Consequently, $p$ belongs to 4 adjacent $r$-pieces, what allows us to conclude that in $G_{\Pi(P)}$ exists a cycle, hence $G_{\Pi(P)}$ is not a tree (see Figure 7.25)

□

Proposition 7.1 establishes that if $P$ is a THIN grid $n$-ogon, with $n \geq 6$, then $G_{\Pi(P)}$ is a path graph. So, each $r$-piece of $\Pi(P)$ is adjacent, at most, to two $r$-pieces. In fact, suppose that a $r$-piece is adjacent to more than two $r$-pieces, then $G_{\Pi(P)}$ would have a node with degree greater than 2, in contradiction to the definition of path graph. In this way, each $r$-piece has at most two interior edges (which are the common sides to other $r$-pieces). Consequently, we conclude that in $\Pi(P)$ there are three types of $r$-pieces:

- ***Type 1***: with one interior edge and three boundary edges;
- ***Type 2***: with two interior edges not adjacent and two boundary edges not adjacent;
- ***Type 3***: with two adjacent interior edges and two adjacent boundary edges.

The $r$-pieces of the Type 1 correspond to leaves of $G_{\Pi(P)}$ and those of the Type 2 and Type 3 correspond to nodes of degree 2 (see Figure 7.26)

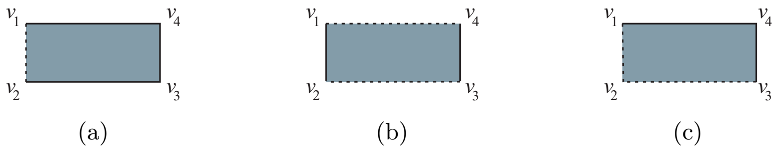



Figure 7.26: The $r$-pieces of a THIN: (a) *Type 1*; (b) *Type 2*; (c) *Type 3*.

Now, the vertices of each one of these $r$-pieces will be analyzed.

***Type 1***

In the $r$-pieces of the *Type 1*, $v_3$ and $v_4$ are convex vertices of $P$. Relatively to $v_1$ and $v_2$, or $v_1$ or $v_2$ it is a reflex vertex of $P$, unable to be both reflex vertices, since $P$ is in general position. Suppose, without loss of generality, that $v_1$ is a reflex vertex of $P$, in this case $v_2$ is an interior point of an edge of $P$ (see Figure 7.27 (a)).

***Type 2***

In the $r$-pieces of the *Type 2* none of four vertices can be a convex vertex of $P$. In fact, let us study $v_1$, for instance. For $v_1$ to be a vertex of $P$ there will be an horizontal edge of $P$ incident on $v_1$, denote this edge by $e_H(v_1)$. As the interior of $P$ is on right of $\overline{v_1v_2}$, $e_H(v_1)$ will be on the left of $\overline{v_1v_2}$, being formed, this way, a reflex vertex in $v_1$. So, $v_1$ cannot be a reflex vertex. In an analogous way, it can be concluded that none of the vertices $v_2$, $v_3$ and $v_4$ can be convex vertices of $P$. Besides, since $\overline{v_1v_4}$ is an interior edge of $P$, or $v_1$ or $v_4$ is a reflex vertex, but being impossible to be both, because $P$ is in general position. Suppose, without loss of generality, that $v_1$ is reflex, in this case, $v_4$ is an interior point of an edge of $P$ (see Figure 7.27 (b) (i)). Relatively to the vertices $v_2$ and $v_3$:

- $v_2$ is a reflex vertex of $P$ and $v_3$ is an interior point of an edge of $P$ (see Figure 7.27 (b) (ii)); or
- $v_3$ is a reflex vertex of $P$ and $v_2$ is an interior point of an edge of $P$ (see Figure 7.27 (b) (iii)).

***Type 3***

In the pieces of the *Type 3*, $v_4$ is a convex vertex of $P$. As $v_2$ cannot be an interior point of $P$, it has to belong to an edge of $P$, being an endpoint or an interior point of the edge. Moreover, as $\overline{v_1v_2}$ is an interior edge of $\Pi(P)$, or $v_1$ or $v_2$ is a reflex vertex of $P$, not being possible to be both, because $P$ is in general position. However, $v_1$ cannot be a reflex vertex. In fact, suppose that $v_1$ is reflex, in this case $v_2$ would be an interior point of a horizontal edge of $P$, what cannot happen since $\overline{v_2v_3}$ is one interior edge of $\Pi(P)$. So, it can be concluded that $v_2$ is a reflex vertex, $v_1$ is one interior point of a horizontal edge of $P$ and $v_3$ is an interior point of a vertical edge of $P$ (see Figure 7.27 (c)).

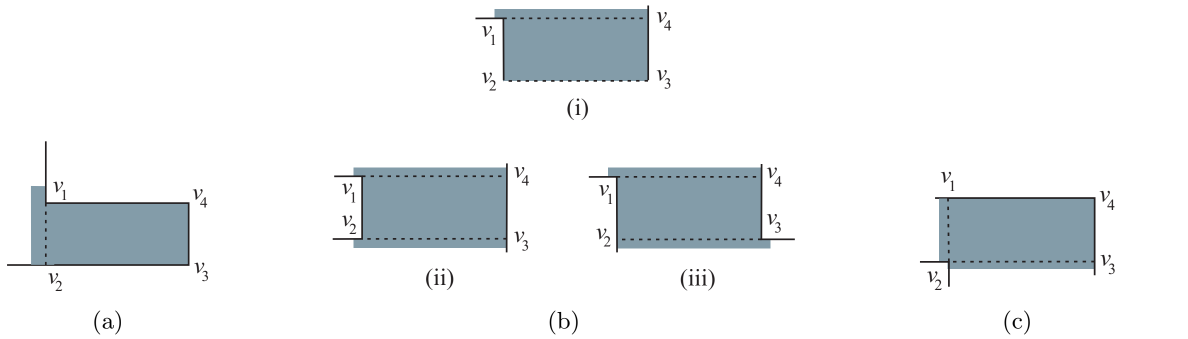



Figure 7.27: The $r$-pieces of a Thin, from left to right: (a) *Type 1*; (b) *Type 2*; (c) *Type 3*.

Summing up, the $r$-pieces of *Type 1* correspond to leaves of $G_{\Pi(P)}$ and those of the *Type 2* and *Type 3* correspond to nodes of degree 2. Of the four vertices of the $r$-pieces of *Type 1* three are vertices of $P$, two being convex and the other reflex, and the other vertex is an interior point of an edge of $P$. Of the four vertices of the $r$-pieces of *Type 2* two are reflex vertices of $P$ and the other two are interior points of edges of $P$. And finally, of the four vertices of the $r$-pieces of *Type 3* two are vertices of $P$, one being reflex and the other convex, and the other two are interior points of edges of $P$

**Proposition 7.3** *The unique convex vertices of a* Thin *grid $n$-ogon $P$ that could be used to yield a* Thin *grid $(n+2)$-ogon, by Inflate-Paste, are those which belong to the $r$-pieces associated with the leaves of $G_{\Pi(P)}$.*

*Proof:* Let $v_i = (x_i, y_i)$ be a convex vertex of $P$ that belongs to a $r$-piece $R$ that does not correspond to a leaf of $G_{\Pi(P)}$. As we saw, $R$ is of the *Type 3* and its form is illustrated in Figure 7.27 (c) (this form is unique except for symmetries of the grid). Denote by $v_r = (x_r, y_r)$ the reflex vertex that belongs to $R$. In the Inflate-Paste process, $FSN(v_i)$ is contained in the orange zone illustrated in Figure 7.28.

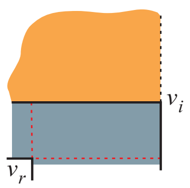



Figure 7.28: $FSN(v_i)$ (free staircase neighborhood of $v_i$).

Denote the horizontal edge of $P$, incident on $v_i$ by $e_H(v_i) = \overline{v_i v_{i+1}}$, where $v_{i+1} = (x_{i+1}, y_{i+1})$. Let $C \in FSN(v_i)$ the cell chosen in the Inflate-Paste process and $c = (x_c, y_c)$ its center. Three situations can take place:

(1) $x_r < x_c < x_i$ (see Figure 7.29 (a));

(2) $x_{i+1} < x_c < x_r$ (see Figure 7.29 (b));

(3) $x_c < x_{i+1}$ (see Figure 7.29 (c))

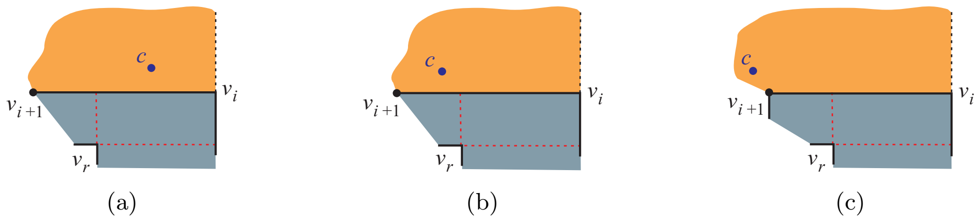



Figure 7.29: The three possibilities for the center of $C$, $c = (x_c, y_c)$: (a) situation (1); (b) situation (2); (c) situation (3).

Notice that the last case will only take place if $v_{i+1}$ is a convex vertex. INFLATE augments the grid creating two free lines $x = x_c$ and $y = y_c$, and it transforms the points $v_i$, $v_{i+1}$ and $c$ in points $\widetilde{v}_i$, $\widetilde{v}_{i+1}$ and $\widetilde{c}$, respectively (see Figure 7.30).

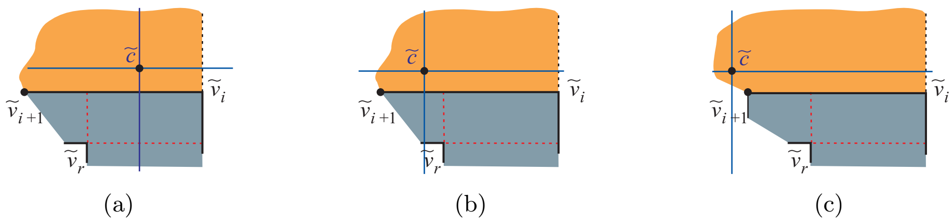



Figure 7.30: INFLATE operation.

Then, PASTE glues to $P$ the rectangle defined by $\widetilde{v}_i$ and $\widetilde{c}$ (see Figure 7.31).

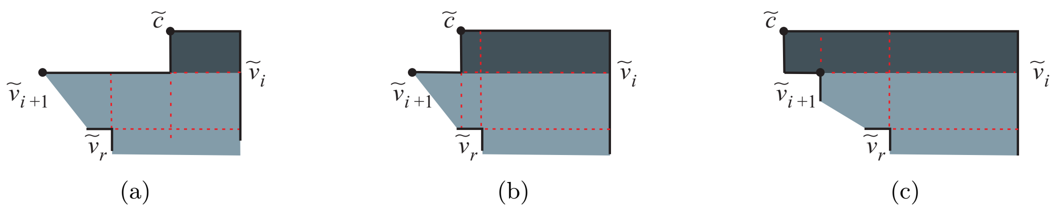



Figure 7.31: PASTE operation.

Note that in any of the three situations the obtained polygon is not a THIN.

□

Lemma 7.1 and proposition 7.3 can be very useful in the generation, by INFLATE-PASTE, of THIN grid $n$-ogons ($n \geq 8$). Lemma 7.1 says that a THIN grid $(n-2)$-ogon must be taken, and proposition 7.3 establishes that the only convex vertices that can "be used" are those which belong to the $r$-pieces associated with the leaves of $G_{\Pi(P)}$ (which are in number of 4). In this way it is not necessary to apply INFLATE-PASTE to all the convex vertices of a THIN and then to check which of the produced polygons are THINs. It is only necessary to apply INFLATE-PASTE to 4 convex vertices and then to check which of the produced polygons are THINs. So the number of case analysis is significantly reduced (see Figure 7.32).

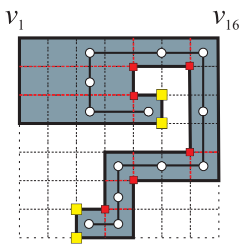



Figure 7.32: The only convex vertices that could yield, by INFLATE-PASTE, the illustrated THIN grid 14-ogons are $v_3$, $v_4$, $v_{11}$ and $v_{12}$.

Now, the *skeleton* of a THIN grid $n$-ogon is going to be defined. Being $P$ a THIN grid $n$-ogon, $G_{\Pi(P)}$ is a path graph, as a result it can be said that $P$ has two *"extremes"*: the $r$-pieces associated with the leaves of $G_{\Pi(P)}$. The extreme that has the horizontal edge with highest $y$-coordinate will be denoted by *kernel*.

Let $P$ be a THIN grid $n$-ogon and $G_{\Pi(P)} = \{V, E\}$, where $V = \{n_1, n_2, \ldots, n_{2r+1}\}$ and $E = \{n_1n_2, n_2n_3, \ldots, n_{2r}n_{2r+1}\}$, being $n_1$ the node associated with the kernel centroid. From $G_{\Pi(P)}$ an *orthogonal polygonal chain* (i.e., a polygonal chain with horizontal or vertical edges) can be obtained in the following way:

(1) For each node $n_i \in V$, associated with the $r$-piece $R_i$, take the centroid of $R_i$. That is, take the centroid of each $r$-piece.

(2) Connect each centroid with the centroids of the adjacent $r$-pieces. In this way, it is obtained an orthogonal polygonal chain, whose vertices are the centroids of the $r$-pieces of $\Pi(P)$.

(3) Remove from this polygonal chain all vertices $(x_i, y_i)$ such that $x_{i-1} = x_i = x_{i+1}$ or $y_{i-1} = y_i = y_{i+1}$. That is, remove the central vertices of each three aligned vertices.

With the process described above an orthogonal polygonal chain is obtained from the dual graph of a THIN grid $n$-ogon. For the first vertex of this orthogonal chain is chosen the the kernel centroid (see Figure 7.33, for an illustration).

Note that in step (3) $r-1$ vertices are removed from the polygonal chain. In fact, it can be easily checked that these vertices are associated with $r$-pieces of *Type 2*. And we know that the total number of $r$-pieces in a THIN grid $n$-ogon is $2r+1$, where 2 are of *Type 1* (the leaves), each one having 2 convex vertices. As a THIN grid $n$-ogon has $r+4$ convex vertices, each $r$-piece of *Type 2* has 0 convex vertices and each $r$-piece of *Type 3* has 1 convex vertex, it

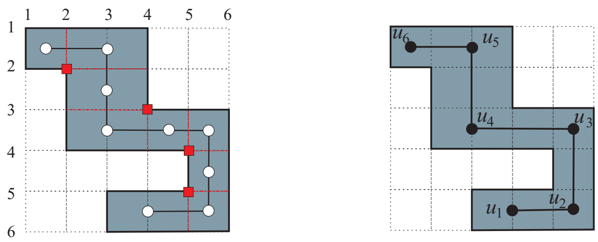



Figure 7.33: A Thin grid $n$-ogon with $r = 4$; on the left is represented its dual graph $G_{\Pi(P)}$ and on the right its skeleton.

can be easily concluded that there are $r$ $r$-pieces of *Type 3*, and consequently there are $r - 1$ $r$-pieces of *Type 2*. So it follows lemma 7.2.

**Lemma 7.2** *The skeleton of a* Thin *grid n-ogon is an orthogonal polygonal chain with* $r+2$ *vertices.*

### 7.2.1 Spiral grid $n$-ogons

Recall that the Thin grid $n$-ogons are not unique (see section 7.1). Besides, it seems that the number of Thin grid $n$-ogons grows exponentially with $n$ (by observation, it is known that there is 1 Thin 6-ogon, there are 2 Thin 8-ogons, there are 7 Thin 10-ogons, there are 30 Thin 12-ogons, there are 149 Thin 12-ogons, and so on...). Until now, the unique subclass of Thins well identified and characterized are the Min-Area grid $n$-ogons, that is: the subclass for which the number of grid cells is minimum. In this section another subclass of Thin grid $n$-ogons will be characterized: the Spiral grid $n$-ogons.

At first sight, spiral polygons are a highly restricted class of polygons that are of little general interest. However, this is not the case. Spiral polygons are a subclass of polygons that have been usefully distinguished in the literature. These polygons can be recognized in linear time and they have arisen in "practice". For instance, Feng and Pavlidis [58, 108] studied decomposition of polygons into spiral pieces for its application on character recognition. Besides, spiral polygons can be seen as the first level of a hierarchy that contains all polygons, the so called $k$-spiral polygons, that is, the polygons having $k$ reflex chains. This hierarchy contains all polygons, therefore, viewed in this light the results presented in this section can be seen as a first step in understanding polygons from the $k$-spiral viewpoint [100].

To characterize the Spiral grid $n$-ogons subclass, it will be firstly defined what is a Spiral grid $n$-ogon. Then, it will be proven that for all $n \geq 6$ there is, at least, a Spiral grid $n$-ogon. Finally, it will be shown that all Spiral grid $n$-ogons are Thin grid $n$-ogons.

**Definition 7.7** *A grid n-ogon is called* **Spiral** ***grid n-ogon*** *if its boundary can be divided into a* ***reflex chain*** *and a* ***convex chain****.*

Remember that, a polygonal chain is called *reflex* if its vertices are all reflex (all except the vertices at the end of the chain) with regard to the interior of the polygon. And, a polygonal chain is called *convex* if its vertices are all convex with regard to the interior of the polygon. Note that, a SPIRAL grid $n$-ogon $P$ can be expressed as an ordered sequence of vertices $u_1, u_2, ..., u_r, c_1, c_2, ..., c_{n-r}$ where the vertices $u_i$ are reflex and the vertices $c_i$ are convex. Thus, the reflex chain is the polygonal chain $c_{n-r}, u_1, ..., u_r, c_1$ and the convex chain is the polygonal chain $c_1, c_2, ..., c_{n-r}$ (see Figure 7.34).

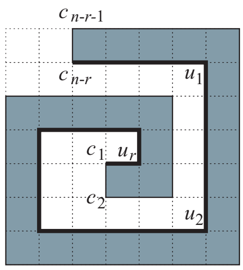



Figure 7.34: Reflex (in bold) and convex chains.

The edges of the convex chain will be denoted by $f_1, f_2, ..., f_{n-r-1}$, where $f_j \equiv \overline{c_j c_{j+1}}$, $1 \leq j \leq n-r-1$. And the edges of the reflex chain will be denoted by $e_0, e_1, ..., e_{r-1}, e_r$, where: $e_0 \equiv \overline{c_{n-r} u_1}$ (the first edge of the reflex chain); $e_i \equiv \overline{u_i u_{i+1}}$, $1 \leq i \leq r-1$; and $e_r \equiv \overline{u_r c_1}$ (the last edge of the reflex chain). Note that $n$ is therefore the total number of edges of $P$, where $r+1$ belongs to the reflex chain and $n-r-1$ to the convex chain.

**Proposition 7.4** *There is, at least, a* SPIRAL *grid $n$-ogon with $r$ reflex vertices, for all $r \geq 1$.*

*Proof:* This demonstration will be done by induction on $r$.

**Base Case, $r = 1$:** There is only one grid $n$-ogon with $r = 1$ and, as we can see in Figure 7.35, it is SPIRAL grid $n$-ogon.

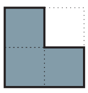


Figure 7.35: Grid $n$-ogon with $r = 1$.

**Inductive Step:** Let, $r \geq 1$. Assuming that the proposition is true for $r$, it will be proven that it is, also, true for $r+1$.

Let $u_1, u_2, ..., u_r, c_1, c_2, ..., c_{n-r}$ be the ordered sequence of vertices that define a SPIRAL grid $n$-ogon $P$, with $r$ reflex vertices. Consider the first vertex of the convex chain $c_1$ and the horizontal edge of $P$ to which $c_1$ belongs $e_H(c_1)$. There are two possible cases: $e_H(c_1) \equiv \overline{u_r c_1}$ (*Case 1*) or $e_H(c_1) \equiv \overline{c_1 c_2}$ (*Case 2*), see Figure 7.36.

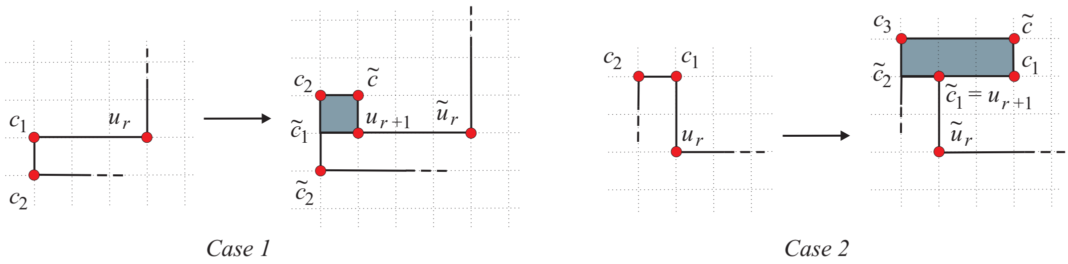



Figure 7.36: On the left it is illustrated *Case 1*, i.e., $e_H(c_1) \equiv \overline{u_r c_1}$; and on the right Case 2, that is, $e_H(c_1) \equiv \overline{c_1 c_2}$.

In Case 1, by taking $c_1 = (x_1, y_1)$, selecting any cell $C$ in $FSN(c_1)$ and applying INFLATE-PASTE to $P$, we will obtain a grid $n$-ogon $Q$ with $r+1$ reflex vertices. The orderly sequence of vertices of $Q$ is $\widetilde{u}_1, \widetilde{u}_2, ..., \widetilde{u}_r, u_{r+1}, \widetilde{c}, c_2, \widetilde{c}_2, ..., \widetilde{c}_{n-r}$, where the vertex $u_{r+1}$ is reflex and the vertices $\widetilde{c}$ and $c_2$ are convex (see Figure 7.36). As we can see $Q$ is a SPIRAL grid $n$-ogon with $r+1$ reflex vertices.

In Case 2, by taking $c_2 = (x_2, y_2)$, selecting a cell $C$ in $FSN(c_2)$ such that its center $c = (c_x, c_y)$ verifies $|c_x - x_2| > |x_1 - x_2|$ and applying INFLATE-PASTE to $P$, we will obtain a grid $n$-ogon $Q$ with $r+1$ reflex vertices, whose orderly sequence of vertices is $\widetilde{u}_1, \widetilde{u}_2, ..., \widetilde{u}_r, u_{r+1}, c_1, \widetilde{c}, c_3, \widetilde{c}_3, ..., \widetilde{c}_{n-r}$, where the vertex $u_{r+1}$ is reflex and the vertices $c_1, \widetilde{c}$ and $c_3$ are convex (see Figure 7.36). As we can check $Q$ is a SPIRAL grid $n$-ogon with $r+1$ reflex vertices.

In any case, a SPIRAL grid $n$-ogon with $r+1$ reflex vertices can always be obtained from $P$. Therefore, for all $r \geq 1$ there is, at least, a SPIRAL grid $n$-ogon with $r$ reflex vertices.

□

The previous proposition establishes that, for all $n \geq 6$ there is, at least, a SPIRAL grid $n$-ogon. However, they are not unique, as it can be seen in Figure 7.37.

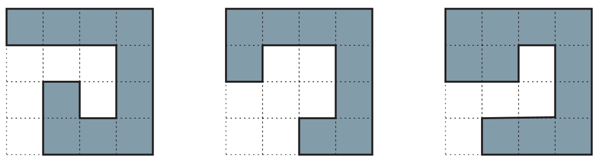


Figure 7.37: A sequence of SPIRAL grid 10-ogons.

Now it will be proven that all SPIRAL grid $n$-ogons are THIN grid $n$-ogons. To show this result, first it will be established that only SPIRAL grid $n$-ogons can yield, by INFLATE-PASTE, SPIRAL grid $(n+2)$-ogons.

**Lemma 7.3** *Only* SPIRAL *grid $n$-ogons can yield, by* INFLATE-PASTE, SPIRAL *grid $(n+2)$-ogons.*

*Proof:* Let $P$ be a grid $n$-ogon and $v_1, v_2, ..., v_n$ its vertices. Take a convex vertex $v_i =$

$(x_i, y_i)$ of $P$ and apply INFLATE-PASTE, this would yield a grid $(n+2)$-ogon $Q$. Suppose that $e_H(v_i) \equiv \overline{v_i v_{i+i}}$, there are two possibilities for $v_{i+1}$: it is either a reflex vertex or it is a convex vertex.

If $v_{i+1}$ is a reflex vertex the form of the rectangle glued by PASTE to yield $Q$ is illustrated in Figure 7.38 (*Case 1*); otherwise the rectangle glued by PASTE to yield $Q$ is of one of the two forms illustrated in Figure 7.38 (*Case 2.1* and *Case 2.2*).

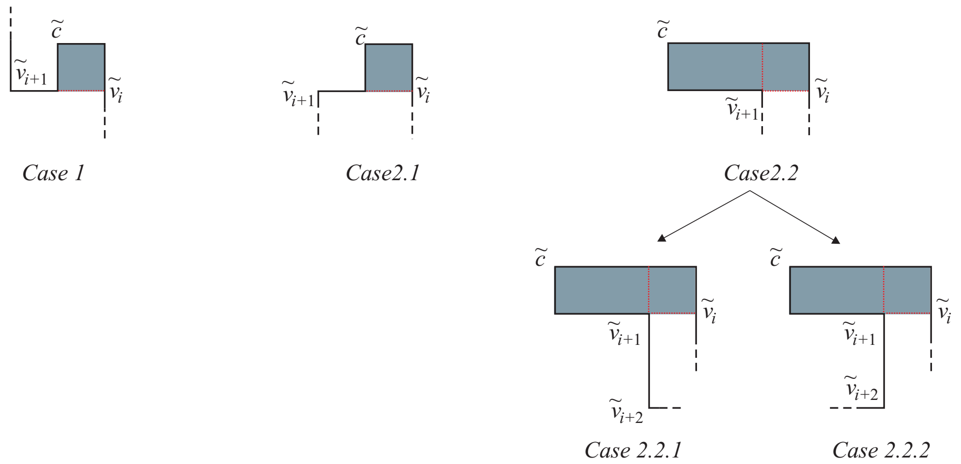



Figure 7.38: Rectangles that might be glued by PASTE to yield $Q$.

In *Case 1*, it is easy to verify that $Q$ is a SPIRAL grid $(n+2)$-ogon only if $P$ is a SPIRAL grid $n$-ogon. In *Case 2.1*, $Q$ is never a SPIRAL grid $(n+2)$-ogon, independently of $P$ being a SPIRAL grid $n$-ogon or not, since a reflex vertex is inserted between two convex ones. In *Case 2.2*, in order to draw conclusions about $Q$, it has to be splited in two cases: *Case 2.2.1*, when $v_{i+2}$ is convex, and *Case 2.2.1*, when $v_{i+2}$ is reflex (see Figure 7.38).

In *Case 2.2.1*, $Q$ is never a SPIRAL grid $(n+2)$-ogon, independently of $P$ being a SPIRAL grid $n$-ogon or not, since a reflex vertex is inserted between two convex ones. In *Case 2.2.2*, it is easy to verify that $Q$ is a SPIRAL grid $(n+2)$-ogon only if $P$ is a SPIRAL grid $n$-ogon.

In conclusion, if $e_H(v_i) \equiv \overline{v_i v_{i+i}}$ and $P$ is not a SPIRAL grid $n$-ogon then $Q$ is never a SPIRAL grid $(n+2)$-ogon in any of the 4 cases. If $e_H(v_i) \equiv \overline{v_i v_{i+i}}$ and $P$ is a SPIRAL then $Q$ is a SPIRAL grid $(n+2)$-ogon in *Cases 1* and *2.2.2* and it is not a SPIRAL in *Cases 2.1* and *2.2.1*.

If $e_H(v_i) \equiv \overline{v_{i-1} v_i}$ we can prove, in an analogous way, that $Q$ is SPIRAL grid $(n+2)$-ogon only if $P$ is a SPIRAL grid $n$-ogon and: either $v_{i-1}$ is a reflex vertex and we select any cell $C$ in $FSN(v_i)$ (see Figure 7.39 (a)) or $v_{i-1} = (x_{i-1}, y_{i-1})$ is convex, $v_{i-2}$ is reflex and we select a cell $C$ in $FSN(v_i)$ such that its center $c = (c_x, c_y)$ verifies $|c_x - x_i| > |x_{i-1} - x_i|$ (see Figure 7.39 (b)).

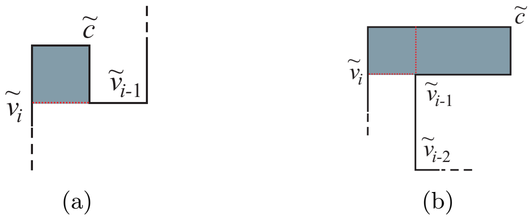



Figure 7.39: Rectangles glued to $P$.

Consequently, only SPIRAL grid $n$-ogons can yield SPIRAL grid $(n+2)$-ogons, by INFLATE-PASTE.

□

**Proposition 7.5** *Every* SPIRAL *grid $n$-ogon, with $r \geq 1$ reflex vertices, is a* THIN *grid $n$-ogon.*

*Proof:* It is easy to see that this proposition holds for $r = 1$. Let $r \geq 1$, it will be proven that the proposition is true for $r+1$, assuming that it is true for $r$. Let $Q$ be a SPIRAL grid $n$-ogon with $r+1$ reflex vertices. By lemma 7.3, $Q$ can only have been generated from a SPIRAL grid $n$-ogon $P$ with $r$ reflex vertices. Moreover, the convex vertex $v_i \in P$ taken to yield $Q$ has to be in such a way that:

- if $e_H(v_i) \equiv \overline{v_i v_{i+1}}$, then: $v_{i+1} \in P$ is reflex (*Case a*)) or $v_{i+1} \in P$ is convex and $v_{i+2} \in P$ is reflex (*Case b*));
- if $e_H(v_i) \equiv \overline{v_{i-1} v_i}$, then: $v_{i-1} \in P$ is reflex (*Case c*)) or $v_{i-1} \in P$ is convex and $v_{i-2} \in P$ is reflex (*Case d*)).

Since $P$ is a SPIRAL grid $n$-ogon it comes that: in *Case a*) $v_i = c_{n-r}$ (the last vertex of the convex chain), in *Case b*) $v_i = c_{n-r-1}$ (the penultimate vertex of the convex chain), in *Case c*) $v_i = c_1$ (the first vertex of the convex chain) and in *Case d*) $v_i = c_2$ (the second vertex of the convex chain). Furthermore, by induction hypothesis, $P$ is a THIN then $G_{\Pi(P)}$ is a path graph, and so it has two leafs. Each has three adjacent vertices of $P$: one reflex vertex preceded or followed by two convex vertices. Thus, it can concluded that $u_r, c_1, c_2$ and $c_{n-r-1}, c_{n-r}, u_1$ belong to the leaves, since they are the only vertices of $P$ in the above stated condition. Therefore, the four *Cases a*), *b*), *c*), and *d*) are illustrated in Figure 7.40.

In lemma 7.3 it has, also, been proven that the rectangle glued to $P$, by PASTE, to yield $Q$ is one of the four forms illustrated in Figure 7.41.

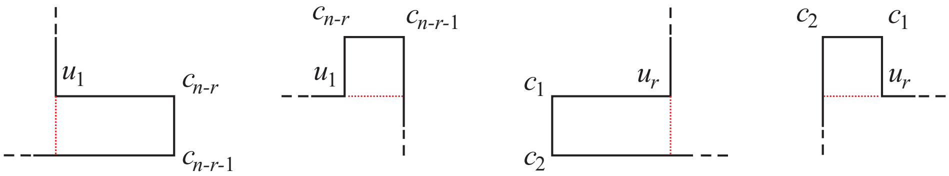


Figure 7.40: From left to right: *Case a*), *Case b*), *Case c*) and *Case d*).

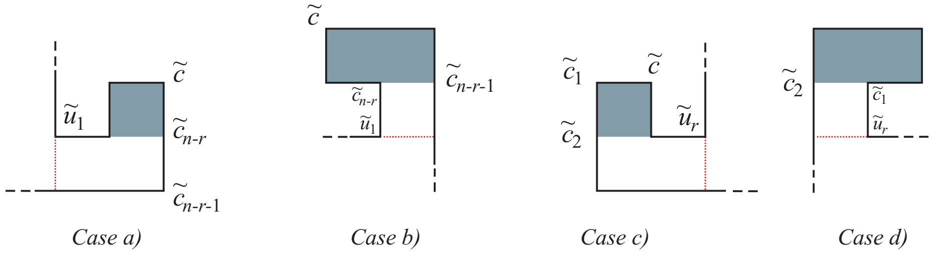



Figure 7.41: Rectangles glued by PASTE to yield $Q$.

In any case, by induction hypothesis, $P$ is THIN then $|\Pi(P)| = 2r+1$. And we can easily check that only two $r$-pieces are added to yield $Q$, thus $|\Pi(Q)| = |\Pi(P)| + 2 = 2(r+1)+1$. Therefore, $Q$ is a THIN grid $n$-ogon.

□

### 7.2.2 Some Problems related to Thin grid $n$-ogons

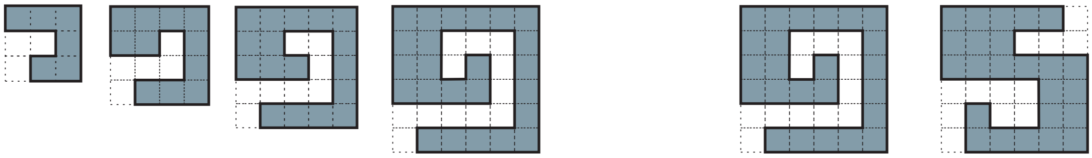
As we saw in section 7.1, as opposed to what happens with the FATs the THIN grid $n$-ogons are not unique. By observation, it is known that there is 1 THIN 6-ogon, there are 2 THIN 8-ogons, there are 7 THIN 10-ogons, there are 30 THIN 12-ogons, there are 149 THIN 12-ogons, and so on... Thus, it is interesting to evidence that the number of THIN grid $n$-ogons ($|\text{THIN}(n)|$) seems to grow exponentially. Will it exist some expression that relates $n$ to $|\text{THIN}(n)|$? Also, we can question on the value of the area of the THIN grid $n$-ogon with maximum area (MAX-AREA-THIN grid $n$-ogon) and if the MAX-AREA-THIN grid $n$-ogon is unique.

#### 7.2.2.1 Max-Area-Thin grid $n$-ogon

Denoting as $MA_r$ the value of the area of "the" MAX-AREA-THIN $n$-ogon with $r$ reflex vertices, by observation it was concluded that $MA_2 = 6$, $MA_3 = 11$, $MA_4 = 17$ and $MA_5 = 24$ (see Figure 7.42 (a)). Note that, $MA_2 = 6$, $MA_3 = MA_2 + 5$, $MA_4 = MA_3 + 6 = MA_2 + 5 + 6$ and $MA_5 = MA_4 + 7 = MA_2 + 5 + 6 + 7$. From these observations it follows:

**Conjecture 7.1** $MA_r = MA_2 + 5 + 6 + 7 + \ldots + (r+2) = \frac{r^2+5r-2}{2}$.

If conjecture 7.1 is true it can be said that the THIN grid $n$-ogon with maximum area is not unique (see Figure 7.42 (b)).



(a) (b)

Figure 7.42: (a) From left to right $MA_2 = 6, MA_3 = 11, MA_4 = 17, MA_5 = 24$; (b) Two THIN 14-ogons with area 24, $MA_5 = 24$.

From left to right in Figure 7.42 (a), the second SPIRAL grid $n$-ogon can be obtained from the first by INFLATE-PASTE, the third SPIRAL grid $n$-ogon can be obtained from second, and so on. So we believe that a MAX-AREA-THIN grid $(n+2)$-ogon can always be obtained from a MAX-AREA-THIN grid $n$-ogon. We intend to use the SPIRAL grid $n$-ogons illustrated in Figure 7.42 (a), lemma 7.1 and propositions 7.1, 7.2 and 7.3 to prove conjecture 7.1.

#### 7.2.2.2 Classifying Thin grid $n$-ogons

As a step for the resolution of the problem placed at the beginning of this section: Will it exist some expression that relates $n$ to $|\text{THIN}(n)|$? First the THINs will be grouped into classes. For this, it will be used the concept of skeleton of a THIN grid $n$-ogon (see lemma 7.2).

From the skeleton of a THIN grid $n$-ogon, it is always possible to represent it by a chain of 0's and 1's, with length $r$. So proceed as follows: (i) transverse the skeleton, starting at vertex $u_1$; (ii) represent each left turn by 1 and each right turn by 0. For example, the chain that represents the THIN illustrated in Figure 7.43 is 1101.

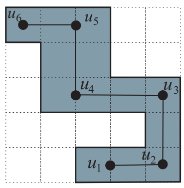



Figure 7.43: THIN grid $n$-ogon with $r = 4$ and its skeleton.

Now, two operations on these chains will be defined: the *complementary operation* and the *inversion operation*.

**Definition 7.8** *Let $c$ be a chain of 0's and 1's, with length $r$, that is, $c = b_1b_2\ldots b_r$, where $b_i = 0$ or $b_i = 1$, for $i = 1, 2, \ldots, r$. The complementary operation is an operation which takes $c$ as argument and returns its complementary $c^* = b_1^*b_2^*\ldots b_r^*$, where $b_i^* = 1$ if $b_i = 0$ and $b_i^* = 0$ if $b_i = 1$, for $i = 1, 2, \ldots, r$.*

**Definition 7.9** *Let $c$ be a chain of 0's and 1's, with length $r$, i.e., $c = b_1b_2\ldots b_r$, where*

*$b_i = 0$ or $b_i = 1$, for $i = 1, 2, \ldots, r$. The inversion operation is an operation which takes $c$ as argument and returns its inverse $c^{-1} = b_r b_{r-1} \ldots b_2 b_1$.*

For example, the complementary of the chain $c = 100011$ is the chain $c^* = 011100$ and its inverse is $c^{-1} = 110001$.

It is easy to check that, $(c^*)^{-1} = (c^{-1})^*$, $(c^*)^* = c$ and $(c^{-1})^{-1} = c$.

**Proposition 7.6** *Let $\mathcal{C}_r$ be the set of all chains, of 0's and 1's, with length $r$. The relation $\sim$ defined on $\mathcal{C}_r$ by*

$$c_1 \sim c_2 \Leftrightarrow c_2 = c_1 \vee c_2 = c_1^{-1} \vee c_2 = c_1^* \vee c_2 = (c_1^*)^{-1},$$

*is an equivalence relation.*

*Proof:* It has to be shown that the relation $\sim$ is reflexive, symmetric, and transitive. The first property it is obvious.

For symmetry: Take $c_1 \sim c_2$, i.e. $c_2 = c_1$ or $c_2 = c_1^{-1}$ or $c_2 = c_1^*$ or $c_2 = (c_1^*)^{-1}$.

1. $c_2 = c_1$. In this case, it is obvious that $c_2 \sim c_1$.
2. $c_2 = c_1^{-1} \Rightarrow c_2^{-1} = (c_1^{-1})^{-1} \Rightarrow c_2^{-1} = c_1$.
3. $c_2 = c_1^* \Rightarrow c_2^* = (c_1^*)^* \Rightarrow c_2^* = c_1$.
4. $c_2 = (c_1^*)^{-1} \Rightarrow c_2^{-1} = c_1^* \Rightarrow (c_2^{-1})^* = c_1$.

In any of the four cases $c_2 \sim c_1$, therefore the relation is symmetric.

For transitivity: Take $c_1 \sim c_2$, that is, $c_2 = c_1$ or $c_2 = c_1^{-1}$ or $c_2 = c_1^*$ or $c_2 = (c_1^*)^{-1}$; and $c_2 \sim c_3$, that is, $c_3 = c_2$ or $c_3 = c_2^{-1}$ or $c_3 = c_2^*$ or $c_3 = (c_2^*)^{-1}$.

1. $c_2 = c_1$ and $c_3 = c_2$. In this case, it is obvious that $c_1 \sim c_3$.
2. $c_2 = c_1$ and $c_3 = c_2^{-1}$.
   If $c_2 = c_1$ then $c_2^{-1} = c_1^{-1}$. Therefore, $c_1 \sim c_3$.
3. $c_2 = c_1$ and $c_3 = c_2^*$.
   If $c_2 = c_1$ then $c_2^* = c_1^*$. Thus, $c_1 \sim c_3$.
4. $c_2 = c_1$ and $c_3 = (c_2^*)^{-1}$.
   $c_2 = c_1 \Rightarrow c_2^* = c_1^* \Rightarrow (c_2^*)^{-1} = (c_1^*)^{-1}$. As a result, $c_1 \sim c_3$.
5. $c_2 = c_1^{-1}$ and $c_3 = c_2$. In this case, it is obvious that $c_1 \sim c_3$.
6. $c_2 = c_1^{-1}$ and $c_3 = c_2^{-1}$.
   If $c_2 = c_1^{-1}$ then $c_2^{-1} = c_1$. Therefore, $c_1 \sim c_3$.
7. $c_2 = c_1^{-1}$ and $c_3 = c_2^*$.
   If $c_2 = c_1^{-1}$ then $c_2^* = (c_1^{-1})^*$. Hence, $c_1 \sim c_3$.

8. $c_2 = c_1^{-1}$ and $c_3 = (c_2^*)^{-1}$.
   If $c_2 = c_1^{-1}$ then $c_2^{-1} = c_1$. Since, $c_3 = (c_2^*)^{-1} = (c_2^{-1})^*$, it can be concluded that $c_3 = c_1^*$. Thus, $c_1 \sim c_3$.
9. $c_2 = c_1^*$ and $c_3 = c_2$. In this case, it is obvious that $c_1 \sim c_3$.
10. $c_2 = c_1^*$ and $c_3 = c_2^{-1}$.
    If $c_2 = c_1^*$ then $c_2^{-1} = (c_1^*)^{-1}$. Therefore, $c_1 \sim c_3$.
11. $c_2 = c_1^*$ and $c_3 = c_2^*$.
    If $c_2 = c_1^*$ then $c_2^* = c_1$. Thus, $c_1 \sim c_3$.
12. $c_2 = c_1^*$ and $c_3 = (c_2^*)^{-1}$.
    If $c_2 = c_1^*$ then $c_2^* = c_1$. Hence, $c_1 \sim c_3$.
13. $c_2 = (c_1^*)^{-1}$ and $c_3 = c_2$. In this case, it is obvious that $c_1 \sim c_3$.
14. $c_2 = (c_1^*)^{-1}$ and $c_3 = c_2^{-1}$.
    If $c_2 = (c_1^*)^{-1}$ then $c_2^{-1} = c_1^*$. Accordingly, $c_1 \sim c_3$.
15. $c_2 = (c_1^*)^{-1}$ and $c_3 = c_2^*$.
    $c_2 = (c_1^*)^{-1} \Rightarrow c_2 = (c_1^{-1})^* \Rightarrow c_2^* = c_1^{-1}$. Therefore, $c_1 \sim c_3$.
16. $c_2 = (c_1^*)^{-1}$ and $c_3 = (c_2^*)^{-1}$.
    $c_2 = (c_1^*)^{-1} = (c_1^{-1})^* \Rightarrow c_2^* = c_1^{-1} \Rightarrow (c_2^*)^{-1} = c_1$. Thus, $c_1 \sim c_3$.

Summing up, in any of the sixteen cases $c_1 \sim c_3$, as a result it can be concluded that the relation is symmetric.

□

Consider, now, the quotient set of $\mathcal{C}_r$ by $\sim$:

$$\mathcal{C}_r/_\sim = \{[c_1]_\sim : c_1 \in \mathcal{C}_r\}, \text{ where } [c_1] = \{c_2 \in \mathcal{C}_r : c_1 \sim c_2\}.$$

Note that, each equivalence class has more than one representant. Here it is assumed that the representant of each equivalence class always starts by 1.

**Proposition 7.7** *Let $\mathcal{P}_r$ be the set of all* THIN *grid $n$-ogons, with $r$ reflex vertices. The relation $\equiv$ defined on $\mathcal{P}_r$ by $P_1 \equiv P_2 \Leftrightarrow c_1 \sim c_2$, where $c_1$ and $c_2$ are the chains that represent $P_1$ and $P_2$, respectively, is an equivalence relation.*

The proof of this proposition is trivial. Consider $\mathcal{P}_r/_\equiv = \{[P_1]_\equiv : P_1 \in C_r\}$. Let $P_1, P_2 \in \mathcal{P}_r$ and $c_1, c_2 \in \mathcal{C}_r$ the chains that represent them, respectively. Notice that, $P_1$ and $P_2$ belong to the same class (i.e., $P_1$ and $P_2$ are *equivalent*) if one of the following conditions is true: (i) $c_1 = c_2$; (ii) $c_2 = c_1^{-1}$; (iii) $c_2 = c_1^*$ or (iv) $c_2 = (c_1^*)^{-1}$. Observe that, geometrically, (ii) can correspond to an horizontal reflection and (iii) to a vertical reflection. In this way, the THINS with 4 reflex vertices in Figure 7.44 represent the same class.

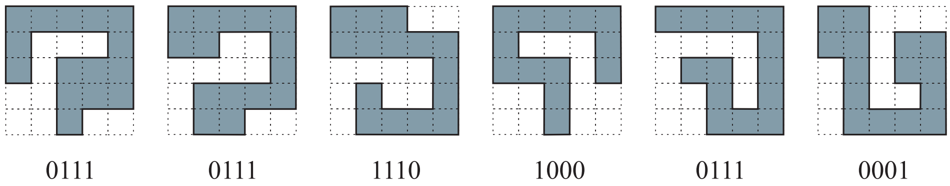



Figure 7.44: THIN grid $n$-ogons with 4 reflex vertices and respective chains.

At this point the following question can be put: Let $c$ be chain of 0's and 1's with length $r$, started by 1. Is it always possible to construct a THIN, with $r$ reflex vertices, whose chain is $c$? To answer this question, the main steps of an algorithm to construct a THIN from a chain of 0's and 1's, started by 1 and with length $r$, are as follows:

(1) From the chain $c$ draw a skeleton ignoring collinearities.

(2) Move a vertical sweep line from left to right to eliminate vertical collinearities.
Repeat the previous step until there are no more collinear vertical edges.

(3) Move a horizontal sweep line from bottom to top to eliminate horizontal collinearities.
Repeat the previous step until there are no more collinear horizontal edges.

Figures 7.45 (a) and 7.45 (b) illustrate the main steps of this algorithm for two chains: 1001 and 1110.

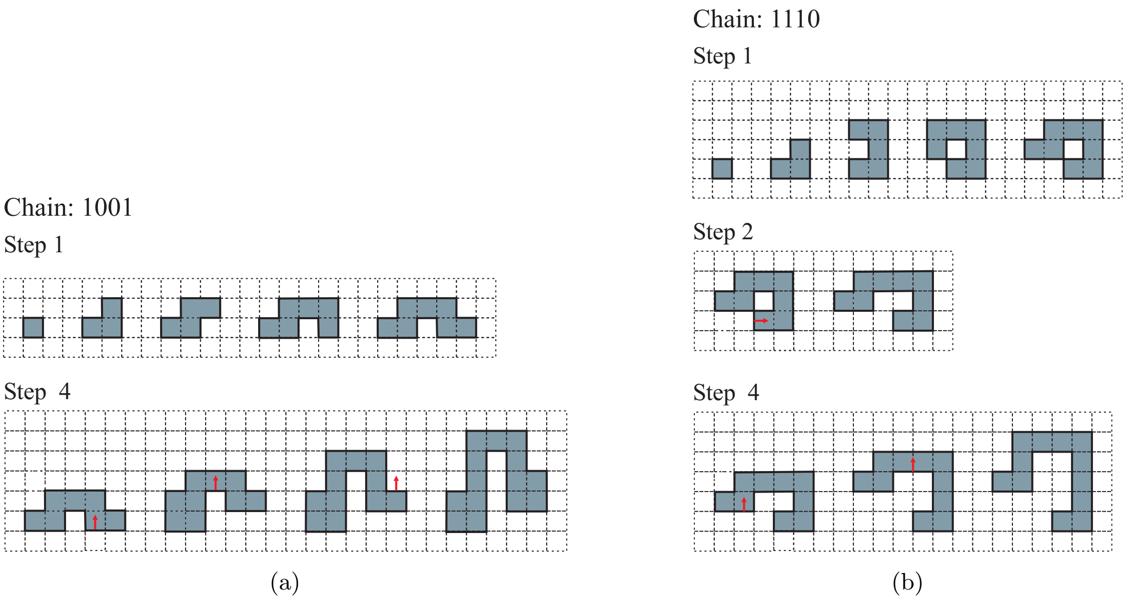



Figure 7.45: Constructing the THIN grid 12-ogon from the chains:(a) $c = 1001$ and (b) $c = 1110$.

**Important Remarks:**

(i) In step (1) if a skeleton is constructed, ignoring the collinearities, a skeleton that does not correspond to the given chain can be obtained, for example, in Figure 7.46 (a), the chain that represents the constructed THIN is 0010000 (complementary followed by inversion of the given chain).

(ii) To eliminate collinearities, in steps (2) and (3), it is necessary to modify the edge corresponding to the beginning of the polygon. If two edges correspond to the beginning of the polygon, or no edge corresponds to the beginning of the polygon, it does not matter which one is modified. Nevertheless, when the polygon has its beginning in two collinear edges, the choice of the edge is not always indifferent, for example, in Figure 7.46 (b), the chain that represents the constructed THIN is 0110 (complementary of the given chain).

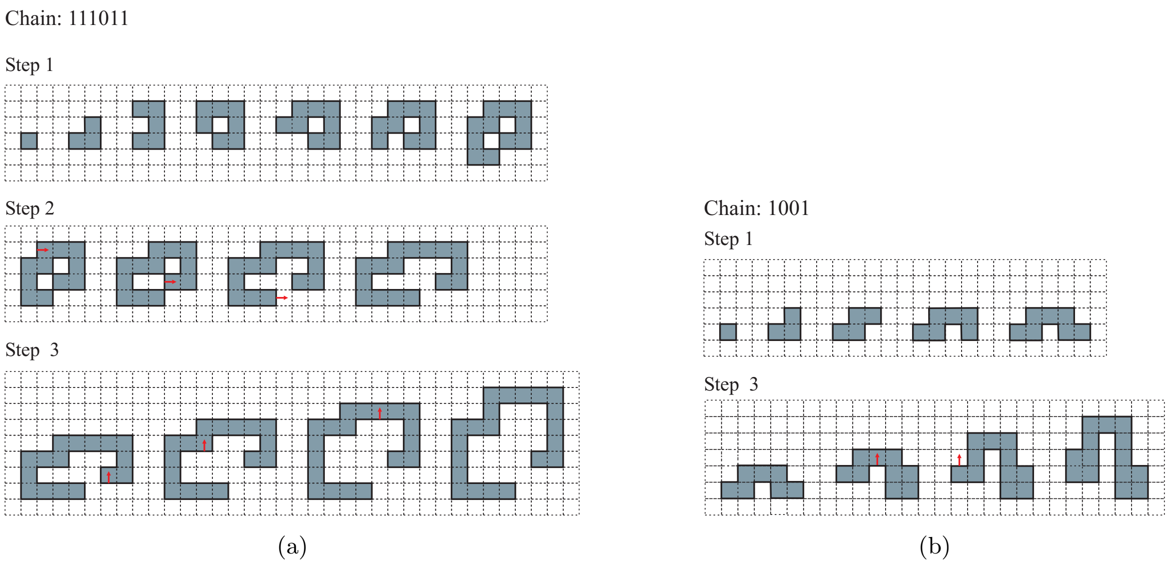



Figure 7.46: Constructing a THIN grid 12-ogon from the chains:(a) $c = 1001$ and (b) $c = 1110$.

Anyhow, this algorithm always generates a THIN grid $n$-ogon whose chain, that represents it is equivalent to $c$. Thus, if the chain that represents the THIN, generated by this algorithm, is $c^*, c^{-1}$ or $(c^*)^{-1}$, it is enough to make a vertical reflection, an horizontal reflection or a vertical reflection followed by a horizontal reflection, respectively, so that the chain, that represents it, may be exactly $c$ (see Figure 7.47 for illustration). Thus, this algorithm proves that for each chain of 0's and 1's, with length $r$, started by 1, there is a THIN grid $n$-ogon with $r$ reflex vertices represented by it.

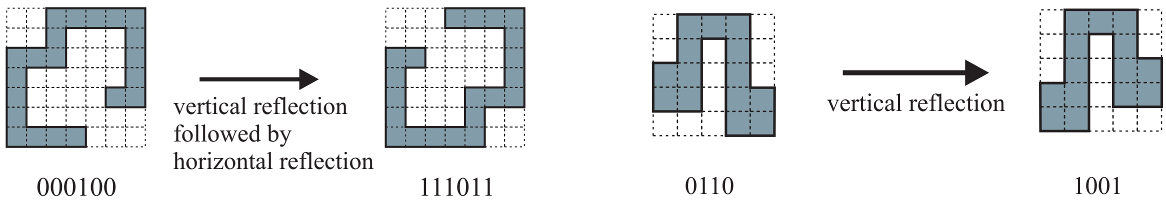



Figure 7.47: (a) The chain that represents the THIN after the horizontal and vertical reflections is $c = 111011$; (b) The chain that represents the THIN after the vertical reflection is $c = 1001$.

Based on the previous reasoning and definitions it is not difficult to prove the following proposition.

**Proposition 7.8** *The correspondence $f : \mathcal{P}_r/_{\equiv} \to \mathcal{C}_r/_{\sim}$ defined by $f([P_1]) = [c_1]$, where $c_1 \in \mathcal{C}_r$ is the chain that represents $P_1 \in \mathcal{P}_r$, which is a representant of the class $[P_1]$, is a bijective function.*

*Proof:* It has to be shown that $f$ is well-defined, is injective and is surjective.

1. $f$ is well-defined.

   (a) Let $[P_1] \in \mathcal{P}_r/_{\equiv}$. We know that there is a chain $c_1 \in \mathcal{C}_r$ that represents $P_1$, thus there is a $[c_1] \in \mathcal{C}_r/_{\sim}$ such that $f([P_1]) = [c_1]$.

   (b) Assume $[P_1] = [P_2]$. Then $P_1 \equiv P_2$. In turn, this implies $c_1 \sim c_2$, where $c_1$ and $c_2$ are the chains that represent $P_1$ and $P_2$, respectively. Therefore $f(P_1) = f(P_2)$

   From (a) and (b) we can conclude that $f$ is well-defined.

2. $f$ is injective.
   Assume $f([P_1]) = f([P_2])$. Then $[c_1] = [c_2]$, where $c_1$ and $c_2$ are the chains that represent $P_1$ and $P_2$, respectively. $[c_1] = [c_2] \Rightarrow c_1 \sim c_2 \Rightarrow P_1 \equiv P_2 \Rightarrow [P_1] = [P_2]$. Therefore, $f$ is injective.

3. $f$ is surjective.
   Let $[c_1] \in \mathcal{C}_r/_{\sim}$. Here there are two cases: $c_1$ starts with 1 or $c_1$ starts with 0.

   (a) If $c_1$ starts with 1, by the algorithm previously presented, we know that there is a $P_1 \in \mathcal{P}_r$ such that is represented by $c_1$. Thus, there is a $[P_1] \in \mathcal{P}_r/_{\equiv}$ such that $f([P_1]) = [c_1]$.

   (b) If $c_1$ starts with 0, we consider $c_1^*$ and, by the algorithm described above, we know that there is a $P_1 \in \mathcal{P}_r$ such that $c_1^*$ represents. Thus, there is a $[P_1] \in \mathcal{P}_r/_{\equiv}$ such that $f([P_1]) = [c_1^*] = [c_1]$.

   From (a) and (b) it can be concluded that $f$ is surjective.

$\square$

The next result allows to count the number of classes of THIN grid $n$-ogons with $r$ reflex vertices.

**Proposition 7.9** *The number of classes of* THIN *grid $n$-ogons with $r$ reflex ($r \geq 2$) is equal to*

$$\begin{cases} 2^{r-2} + 2^{\frac{1}{2}(r-3)} & \textit{if } r \textit{ is odd} \\ 2^{r-2} + 2^{\frac{1}{2}(r-2)} & \textit{if } r \textit{ is even} \end{cases}$$

*Proof:* By Proposition 7.8 we can conclude that $|\mathcal{P}_r/_{\equiv}| = |\mathcal{C}_r/_{\sim}|$, so we just have to determine $|C_r/_{\sim}|$.

1. The cardinal of $C_r$ is $2^r$.
2. The number of symmetrical chains ($c = c^{-1}$), with length $r$, is $2^{\lceil \frac{r}{2} \rceil}$.
3. If a chain $c$ is symmetrical, then its equivalence class is constituted by two chains, $c$ and $c^*$.
4. If a chain $c$ is not symmetrical, to find the cardinal of its class we have to distinguish two cases: $r$ odd and $r$ even.
   (a) $r$ odd. All the chains have 4 equivalent chains: $c$, $c^{-1}$, $c^*$ and $(c^*)^{-1}$.
   For example: $c = 11010, 01011, 00101$ and $10100$.
   (b) $r$ even. In this case there are chains that have 4 equivalent chains , for example $c = 1110$. And chains that only have 2 equivalent chains, this case happens when $c^* = c^{-1}$.
   For example: for the chain $c = 1100$, $c^* = c^{-1} = 0011$.

Let us count, now, the number of equivalence classes.

- **Case $r$ odd**

  Equivalence classes of symmetrical chains:

  $\frac{1}{2}\#(\text{symmetrical chains}) = \frac{1}{2}\ 2^{\frac{r+1}{2}} = 2^{\frac{r-1}{2}}$

  Equivalence classes of non symmetrical chains:

  $\frac{1}{4}\#(\text{non symmetrical chains}) = \frac{1}{4}\ (2^r - 2^{\frac{r+1}{2}}) = 2^{r-2} - 2^{\frac{r-3}{2}}$

  On the whole, for $r$ odd, the number of equivalence classes is:

  $2^{r-2} + 2^{\frac{r-1}{2}} - 2^{\frac{r-3}{2}} = 2^{r-2} + 2^{\frac{1}{2}(r-3)}$

- **Case $r$ even**

  Equivalence classes of symmetrical chains:

$\frac{1}{2}\#(\text{symmetrical chains}) = \frac{1}{2}\ 2^{\frac{r}{2}} = 2^{\frac{1}{2}(r-2)}$

Equivalence classes of non symmetrical chains constituted by **two** chains (for example the classes of the chains 101010, 1100, 110100,.....):

$\frac{1}{2}\ 2^{\frac{r}{2}} = 2^{\frac{1}{2}(r-2)}$

In fact, to obtain that $c^* = c^{-1}$, the second half of the chain is completely determined by the first half. Therefore, the cardinal of these classes is half of the number of chains of this type.

Equivalence classes of non symmetrical chains constituted by **four** chains:

$\frac{1}{4}\#(\text{All} - \text{Symmetric} - (\text{Chains with } c^* = c^{-1})) = \frac{1}{4}\ (2^r - 2^{\frac{r}{2}} - 2^{\frac{r}{2}}) = 2^{r-2} - 2^{\frac{1}{2}(r-2)}$

On the whole, for $r$ even,, the number of equivalence classes is: $2^{r-2} + 2^{\frac{1}{2}(r-2)}$

□

Some problems related to Thins were solved. However, there are still some open problems to solve, such as:

**Open Problem 7.1** *How many elements* Thin *grid $n$-ogons does each class have?*

**Open Problem 7.2** *Will it be possible to find an algorithm that generates all* Thin *grid $n$-ogons of the same class?*

**Open Problem 7.3** *Is there any expression that relates $n$ to* $|\text{Thin}(n)|$*?*

Note that, solving the first problem, the initial one (i.e., the problem posed in the beginning of this subsection), which is the open problem 9.2, is also solved.

## 7.3 Visibility Problems on grid $n$-ogons

Of the problems related to grid $n$-ogons, the guarding and hiding problems are the ones that motivate us more, particularly the problem of finding the minimum number of vertex guards needed to guard a given simple polygon (i.e., the MVGS($P$) problem) and the problem of determining the maximum number of vertices of a given polygon, such that no two of these vertices see each other (i.e., the MHVS($P$) problem). Since, the Thin and the Fat $n$-ogons are the classes for which the number of $r$-pieces is minimum and maximum, respectively, one can think that they can be representative of extremal behaviour. Besides, they were used experimentally to evaluate approximation methods of resolution of the MVGS($P$) problem [36,37,126], so the study started with them.

In section 7.3.1 it will be presented results related to the MVGS($P$) problem, where $P$ is a FAT, a MIN-AREA or a SPIRAL grid $n$-ogon. And, in section 7.3.2 the MHVS($P$) problem will be studied, being $P$ a Thin grid $n$-ogon.

### 7.3.1 Minimum Vertex Guard Set Problem on grid n-ogons

#### 7.3.1.1 Fat grid $n$-ogons

Here it is assumed that the rooms of the art gallery have the form of a FAT grid $n$-ogon and we want to know how many vertex guards are sufficient to cover them. In the context of this problem it will be necessary to determine: how many guards are enough to cover a FAT grid $n$-ogon? and, where must these guards be placed?

**Proposition 7.10** *To cover completely any* FAT *grid $n$-ogon it is always sufficient two vertex guards.*

The proof of this proposition is trivial. It is enough to decompose the FAT grid $n$-ogon in two staircase polygons[1] (with not disjoint interiors) and to place a vertex guard at the vertices that are in the intersection of the height edge and the base edge of the respective staircase polygons (see Figure 7.48).

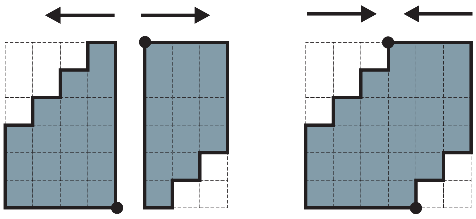


Figure 7.48: Guarded FAT grid 14-ogon.

**Conjecture 7.2** *The only way to cover a* FAT *grid $n$-ogon with two vertex guards is illustrated in Figure 7.48.*

As we can see, the problem of guarding a FAT grid $n$-ogon, with vertex guards, is very simple.

#### 7.3.1.2 Thin grid $n$-ogons

The THIN grid $n$-ogons are much more difficult to cover, contrary to one might think once they have much fewer $r$-pieces than the FATS. Besides, they are not unique, so in the previous sections we tried to characterize structural properties of THINS classes that allow to simplify the problem study. Up to now, the only quite characterized subclasses are the MIN-AREA and the SPIRAL grid $n$-ogons. The study of the MVGS($P$) problem on these two subclasses of THIN grid $n$-ogons will follow.

[1] A staircase polygon is an orthogonal polygon with an horizontal edge $h$, whose length is equal to the sum of the lengths of the remaining horizontal edges, and a vertical edge $v$, whose height is equal to the sum of the lengths of the remaining vertical edges. The horizontal edge $h$ is called *base edge* and the vertical edge $v$ is designated by *height edge*.

**Min-Area grid $n$-ogons**

It is already known that $\lfloor \frac{n}{4} \rfloor$ vertex guards are always sufficient to cover any Min-Area grid $n$-ogon, since a Min-Area is a $n$-ogon. We will start by improving this result, proving that $\lceil \frac{r+2}{3} \rceil = \lceil \frac{n}{6} \rceil$ vertex guards are always sufficient to cover any Min-Area grid $n$-ogon. After that, it will be shown that not only this number of guards is sufficient, but also it is necessary to cover any Min-Area grid $n$-ogon.

**Sufficiency.** The following lemma will be used in the proof of proposition 7.11. This proposition establishes that $\lceil \frac{r+2}{3} \rceil$ vertex guards are always sufficient to cover any Min-Area grid $n$-ogon with $r$ reflex vertices.

**Lemma 7.4** *Let $P$ be a* Min-Area *grid $n$-ogon with $r \geq 4$ reflex vertices. If the "line 3" is removed, then two* Min-Area *grid $n$-ogons wil be obtained (see Figure 7.49 (a)).*

*Proof:* Let $P$ be a Min-Area grid $n$-ogon with $r \geq 4$ reflex vertices. If the "line 3" is removed then two grid $n$-ogons will be obtained, $P_1$ and $P_2$ (see Figure 7.49 (a)). Clearly, $P_1$ is a Min-Area grid $n$-ogon with one reflex vertex.

The construction of a Min-Area grid $n$-ogon with $r \geq 2$ reflex vertices, is done by an iterative method that builds the Min-Area from the unit square by applying $r$ times the Inflate-Paste process. Each time the Inflate-Paste is applied just two cells are glued to the polygon being constructed (see [24], for details).

So, applying this method to the unit square $Q_0 = (\frac{n}{2}-1, \frac{n}{2}-1)(\frac{n}{2}-1, \frac{n}{2})(\frac{n}{2}, \frac{n}{2})(\frac{n}{2}, \frac{n}{2}-1)$, see Figure 7.49 (b), the polygon $P$ is obtained.

Notice that, in the $(r-3)^{th}$ iteration we have a Min-Area grid $n$-ogon with $r-3$ reflex vertices, being this polygon $P_2$. Therefore, $P_2$ is a Min-Area grid $n$-ogon.

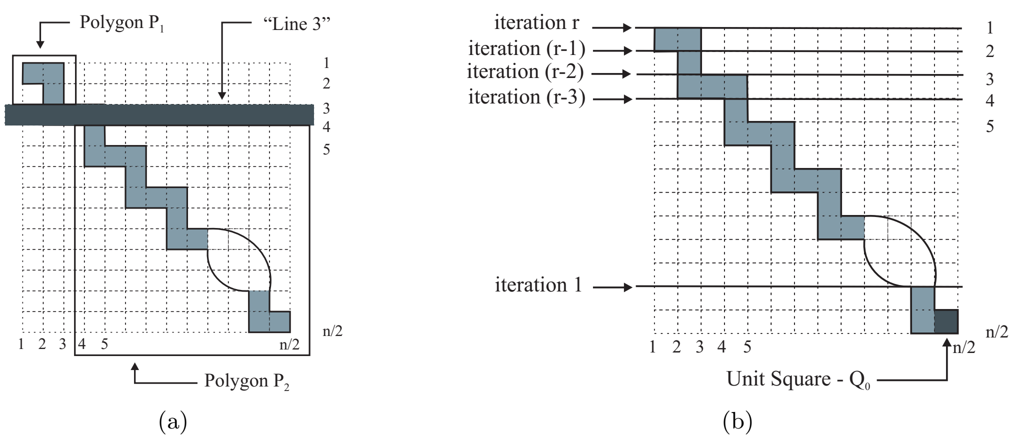



Figure 7.49: (a) Removing "line 3"; (b) "Constructing" $P$.

In conclusion, removing the "line 3" two polygons, $P_1$ and $P_2$, are obtained, being both MIN-AREA grid $n$-ogons.

□

In the forthcoming propositions and lemmas (proposition 7.11, lemma 7.5, proposition 7.12 and proposition 7.13), due to practical reasons, the vertices of a MIN-AREA grid $n$-ogon, will be denoted by $v_{i,j} = (i,j)$, with $i,j \in \{1,\ldots,\frac{n}{2}\}$, where $(i,j)$ are the coordinates of the vertex $v_{i,j}$ on the grid.

**Proposition 7.11** $\lceil\frac{r+2}{3}\rceil$ *vertex guards are always sufficient to cover a* MIN-AREA *grid $n$-ogon with $r$ reflex vertices.*

*Proof:* Let $P$ be a MIN-AREA grid $n$-ogon, with $r$ reflex vertices, being $v_{i,j} = (i,j)$, with $i,j \in \{1,\ldots,\frac{n}{2}\}$, its vertices. Consider the $\lceil\frac{r+2}{3}\rceil$ vertex guards placed on the following vertices:

$$\begin{cases} v_{2+3i,2+3i},\, i = 0,1,\ldots,\frac{r-1}{3} & \text{if} \quad r \equiv 1\ (mod\ 3) \\ v_{2+3i,2+3i},\, i = 0,1,\ldots,\frac{r-2}{3} \quad \textit{and} \quad v_{r+1,r+1} & \text{if} \quad r \equiv 2\ (mod\ 3) \\ v_{2+3i,2+3i},\, i = 0,1,\ldots,\frac{r-3}{3} \quad \textit{and} \quad v_{r+1,r+1} & \text{if} \quad r \equiv 0\ (mod\ 3) \end{cases} \tag{7.1}$$

It will be shown, by induction on $r$, that these vertex guards are sufficient to cover $P$.

Can be checked easily that this is true for $r \leq 4$ (see Figure 7.50). Note that, the placement of the $\lceil\frac{r+2}{3}\rceil$ vertex guards, in some cases, is not unique (see Figure 7.51). Let $r \geq 5$.

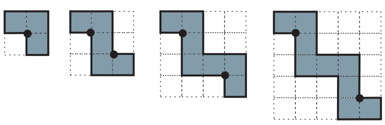


Figure 7.50: MIN-AREA grid $n$-ogons with $r = 1, 2, 3$ and 4.

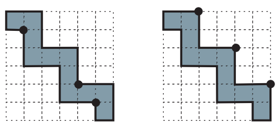


Figure 7.51: MIN-AREA grid $n$-ogon with $r = 5$.

*Induction Hypothesis:* The vertex guards established in (7.1) are sufficient to cover any MIN-AREA grid $n$-ogon $P$ with $1 \leq m < r$ reflex vertices.

*Induction Thesis:* The vertex guards established in (7.1) are sufficient to cover a MIN-AREA grid $n$-ogon $P$ with $r$ reflex vertices.

Let $P$ be a MIN-AREA grid $n$-ogon with $r \geq 5$ reflex vertices. Remove the "line 3" in $P$, as illustrated in Figure 7.49 (a). By lemma 7.4, two MIN-AREA grid $n$-ogons, $P_1$ and $P_2$, are obtained. $P_1$ has $r_1 = 1$ reflex vertex and $P_2$ has $r_2 = r - 3$ reflex vertices, thus $r = r_1 + r_2 + 2$.

Denote by $v_{i,j}^{(1)}$ the vertices of $P_1$ and by $v_{i,j}^{(2)}$ the vertices of $P_2$. Therefore, $v_{i,j}^{(1)} = v_{i,j}$, for $i, j \in \{1, 2, 3\}$ and $v_{i,j}^{(2)} = v_{i+3,j+3}$, for $i, j \in \{1, 2, ..., \frac{n-6}{2}\}$. By induction hypotheses, the vertex guard $v_{2,2}^{(1)} = v_{2,2}$ is sufficient to cover $P_1$.

Now, consider the polygon $\tilde{P}_2$, symmetric of $P_2$ relative to the positive diagonal , i.e., the diagonal that contains the reflex vertices, and denote by $\tilde{v}_{i,j}^{(2)}$ its vertices (see Figure 7.52 (a)).

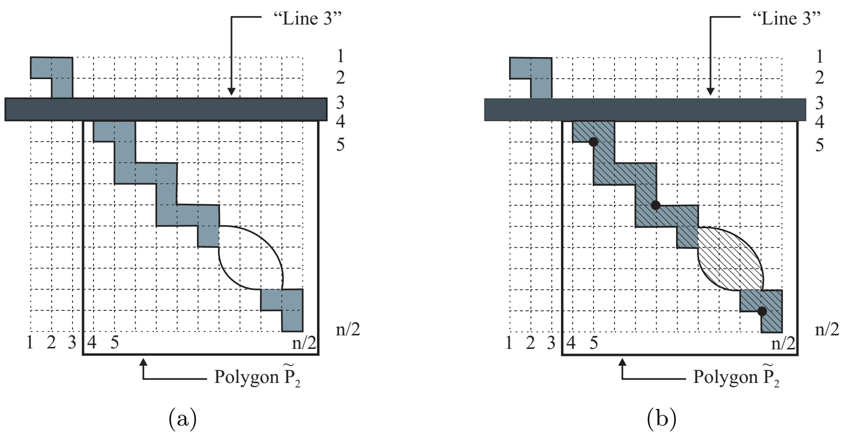



Figure 7.52: (a) Polygon $\tilde{P}_2$; (b) Applying induction hypotheses to $\tilde{P}_2$.

By induction hypotheses, the following $\lceil \frac{r_2+2}{3} \rceil$ vertex guards are sufficient to cover $\tilde{P}_2$ (see Figure 7.52 (b)):

$$\begin{cases} v_{2+3i,2+3i}^{(2)},\ i = 0, 1, ..., \frac{r_2-1}{3} & \text{if} \quad r \equiv 1\ (mod\ 3) \Leftrightarrow r_2 \equiv 1\ (mod\ 3) \\ v_{2+3i,2+3i}^{(2)},\ i = 0, 1, ..., \frac{r_2-2}{3} \quad and \quad v_{r_2+1,r_2+1}^{(2)} & \text{if} \quad r \equiv 2\ (mod\ 3) \Leftrightarrow r_2 \equiv 2\ (mod\ 3) \\ v_{2+3i,2+3i}^{(2)},\ i = 0, 1, ..., \frac{r_2-3}{3} \quad and \quad v_{r_2+1,r_2+1}^{(2)} & \text{if} \quad r \equiv 0\ (mod\ 3) \Leftrightarrow r_2 \equiv 0\ (mod\ 3) \end{cases}$$

Take into account, now, the symmetric of $\tilde{P}_2$ relative to the positive diagonal, i.e. $P_2$, is covered with the same $\lceil \frac{r_2+2}{3} \rceil$ vertex guards (see Figure 7.53 (a)).

Note that, $P = P_1 \cup R \cup P_2$, where $R$ is the rectangle $R = (2,3)(2,4)(5,4)(5,3)$. Thus, $P$ is all covered except in the rectangle $R$ (see Figure 7.53 (b)). However, the vertex guard $v_{2,2}^{(1)}$ of $P_1$ (i.e., the vertex guard $v_{2,2}$ of $P$) covers the quadrilateral $Q_1 = (2,3)(2,4)(4,4)(3,3)$ and the vertex guard $v_{2,2}^{(2)}$ of $P_2$ (i.e., the vertex guard $v_{5,5}$ of $P$) cover the quadrilateral $Q_2 = (3,3)(4,4)(5,4)(5,3)$ (see Figure 7.53 (c)). But being $R = Q_1 \cup Q_2$, so $R$ is covered. Consequently, $P$ is completely covered (see Figure 7.53 (c)).

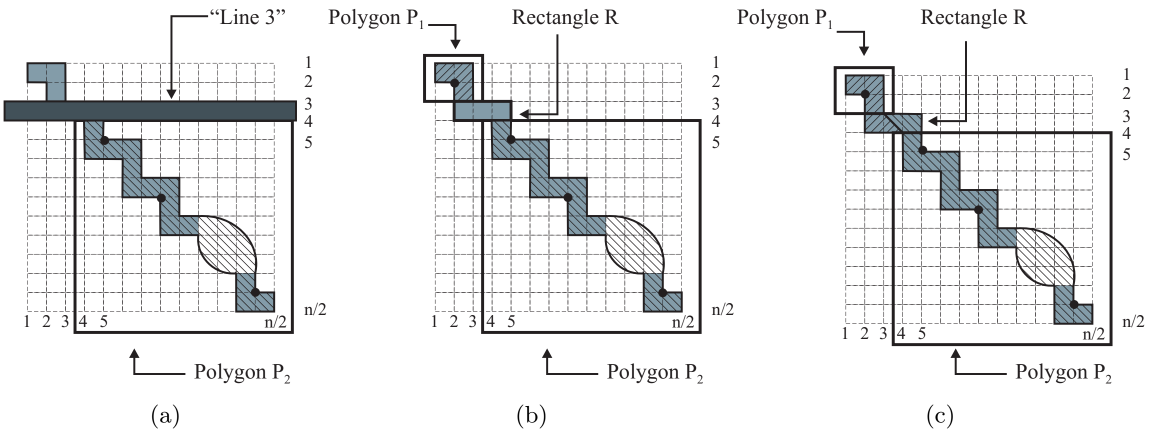



Figure 7.53: (a) Polygon $P_2$ completely covered; (b) Rectangle $R$; (c) Quadrilaterals $Q_1$ and $Q_2$.

Summing up, $\lceil \frac{r_2+2}{3} \rceil + 1 = \lceil \frac{r+2}{3} - 1 \rceil + 1 = \lceil \frac{r+2}{3} \rceil$ vertex guards are enough to cover $P$. □

Proposition 7.11 not only gives the guarantee of that $\lceil \frac{r+2}{3} \rceil$ vertex guards are always sufficient to cover a MIN-AREA grid $n$-ogon with $r$ reflex vertices, but also establishes where the vertices should be placed.

**Necessity.** At this point, it will be shown that $\lceil \frac{n}{6} \rceil = \lceil \frac{r+2}{3} \rceil$ vertex guards are necessary to cover any MIN-AREA grid $n$-ogon. In other words, it will be established that less than $\lceil \frac{n}{6} \rceil$ vertex guards are not enough to cover a MIN-AREA grid $n$-ogon. First, it will be proven that this number of vertex guards is required to cover any Min-Area *grid n-ogon* with $r \equiv 1 \ (mod\ 3)$ reflex vertices. Besides, the only possible positioning for those guards will be established (lemma 7.5 and proposition 7.12). Then, using these results, it will be shown that this number of vertex guards is, also, necessary to cover any MIN-AREA grid $n$-ogon with $r \equiv 0 \ (mod\ 3)$ or $r \equiv 2 \ (mod\ 3)$ reflex vertices (proposition 7.13).

**Lemma 7.5** *Two vertex guards are necessary to cover the Min-Area* grid 12-ogon*. Moreover, the only way to do so is with the vertex guards $v_{2,2}$ and $v_{5,5}$.*

*Proof:* Let $P$ be the Min-Area *grid* 12-*ogon*. The unit square $Q_0 = (1,1)(1,2)(2,2)(2,1)$ has to be guarded. The only vertex guards that can do it are: $v_{1,1}$, $v_{1,2}$ , $v_{2,2}$ and $v_{3,1}$ (see Figure 7.54).

As we can see in Figure 7.55, $Vis(v_{1,2}, P) \subset Vis(v_{1,1}, P) \subset Vis(v_{3,1}, P) \subset Vis(v_{2,2}, P)$. Since we intend to minimize the number of guards that cover $P$ the vertex $v_{2,2}$ is chosen. Observing Figure 7.55, we can conclude that it is necessary more than a vertex guard to cover $P$. By proposition 7.11 we known that we do not need more than two vertex guards to cover

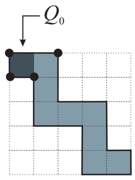



Figure 7.54: Min-Area grid 12-ogon.

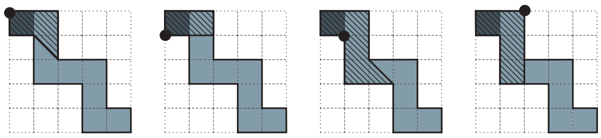


Figure 7.55: Visibility Regions.

$P$. Thus, we can conclude that exactly two vertex guards are needed to cover $P$. This ends the proof of the first part of the proposition.

Let us see where the second vertex guard must be placed. The unit square $Q_1 = (5,6)(6,6)(6,5)(5,5)$ must be guarded. The only vertex guards that can do it are: $v_{4,6}$, $v_{6,6}$ , $v_{6,5}$ and $v_{5,5}$ (see Figure 7.56).

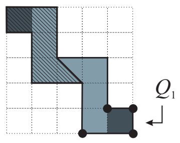



Figure 7.56: Min-Area grid 12-ogon.

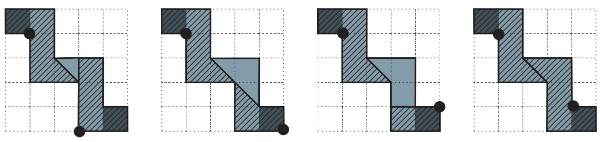


Figure 7.57: Visibility Regions.

We can easily see that, of these vertex guards, the only one that "works for" is $v_{5,5}$, since the choice of any other would left parts of $P$ not covered (see Figure 7.57).

So it can be concluded that 2 vertex guards are necessary to cover the Min-Area *grid* 12-*ogon* and the only way to do so is with the vertex guards $v_{2,2}$ and $v_{5,5}$.

□

**Proposition 7.12** *If* $k \geq 2$ Min-Area *grid* 12-*ogons are "merged", then the* Min-Area *grid* $n$*-ogon with* $r = 3k+1$ *is obtained. Moreover,* $k+1$ *vertex guards are necessary to cover it, and the only way to do so is with the vertex guards:* $v_{2+3i,2+3i}, i = 0, 1, \ldots, k$.

*Proof:* Let $P$ be the Min-Area *grid* $n$*-ogon* with $r = 7$ reflex vertices. $P$ can be obtained from two Min-Area *grid* 12-*ogons*, as we can see in see Figure 7.58.

The resulting polygon has $r = 7$ reflex vertices and not $r = 8$, once, by construction, vertices $v_{5,5}$ of the first polygon and $v_{1,1}$ of the second polygon are over lapped. Besides, by lemma 7.5 and as we can see, 3 vertex guards are necessary to cover $P$, and the only way to do that is with the vertex guards placed on the vertices: $v_{2,2}$, $v_{5,5}$ and $v_{8,8}$.

Thus, for $k = 2$, the proposition is true. Let $k \geq 2$, it will be shown that the proposition is true for $k+1$, assuming that it is true for $k$.

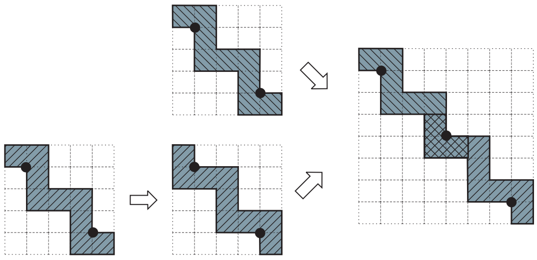


Figure 7.58: Construction of the Min-Area grid 18-ogon from two Min-Area grid 12-ogons.

First, it has to be proven that if $k+1$ MIN-AREA grid 12-ogons are "merged", then the MIN-AREA grid $n$-ogon with $r = 3(k+1)+1 = 3k+4$ reflex vertices is obtained.

By induction hypothesis, "merging" $k$ MIN-AREA grid 12-ogons the MIN-AREA grid $n$-ogon $Q$, with $r_q = 3k+1$ reflex vertices, is obtained. If $Q$ is "merged" with the MIN-AREA grid 12-ogon, a polygon $P$ will be obtained (see Figure 7.59).

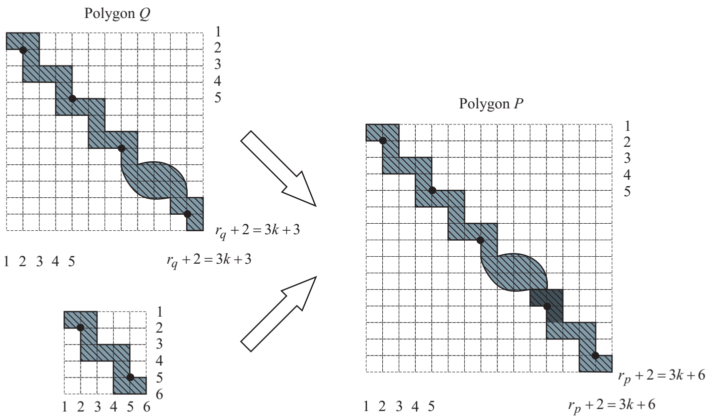



Figure 7.59: Polygon $P$ ("merging" $Q$ with the Min-Area grid 12-ogon).

$P$ has $r_p = r_q+3 = 3k+4$ reflex vertices. Besides, $A(P) = A(Q)+6$. As $Q$ is a MIN-AREA, $A(Q) = 2r_q+1$. Consequently, $A(P) = 2r_q+1+6 \Leftrightarrow A(P) = 2(r_p-3)+7 \Leftrightarrow A(P) = 2r_p+1$. Therefore, $P$ is the MIN-AREA grid $n$-ogon with $r = 3(k+1)+1$ reflex vertices.

Besides, by induction hypotheses, and as we can observe in Figure 7.59, it can be concluded that $\lceil \frac{r_q+2}{3} \rceil + 1 = k+2$ vertex guards are necessary to cover $P$. Moreover, the only way to do so is with the vertex guards placed on the following vertices: $v_{2,2}$, $v_{5,5}, \ldots, v_{2+3k,2+3k}$ and $v_{5+3k,5+3k}$.

$\square$

**Proposition 7.13** $\lceil \frac{r+2}{3} \rceil$ *vertex guards are always necessary to cover any* MIN-AREA *grid $n$-ogon with $r$ reflex vertices.*

*Proof:* Let $P_n$ be a MIN-AREA grid $n$-ogon with $r_n = \frac{n-4}{2}$ reflex vertices. We may easily check that $1, 2$ and $2$ vertex guards are necessary to guard MIN-AREA grid $n$-ogons with $r_n = 1, 2, 3,$

respectively (see Figure 7.60).

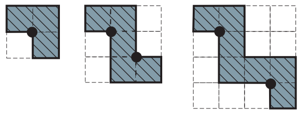


Figure 7.60: Min-Area grid $n$-ogons with $r = 1, 2$ and 3.

Let $r_n \geq 4$. If $r_n \equiv 1$ ($mod$ 3) then, by proposition 7.12, the $\lceil \frac{r_n+2}{3} \rceil$ vertex guards placed on the vertices: $v_{2+3i,2+3i}$, $i = 0, 1, \ldots, \frac{r_n-1}{3}$, are necessary to cover $P_n$. Thus, it is just necessary to prove the following cases: $r_n \equiv 2$ ($mod$ 3) and $r_n \equiv 0$ ($mod$ 3).

In any case, $P_n$ can be obtained, by INFLATE-PASTE, from a Min-Area $Q_m$ with $r_m = \frac{m-4}{2}$ and such that $r_m = 3k_m + 1$ (see Figure 7.61). The first case corresponds to polygon $Q_{m+2}$, in Figure 7.61, and $r_n = r_m + 1$. The second case corresponds to polygon $Q_{m+4}$, in Figure 7.61, and $r_n = r_m + 2$.

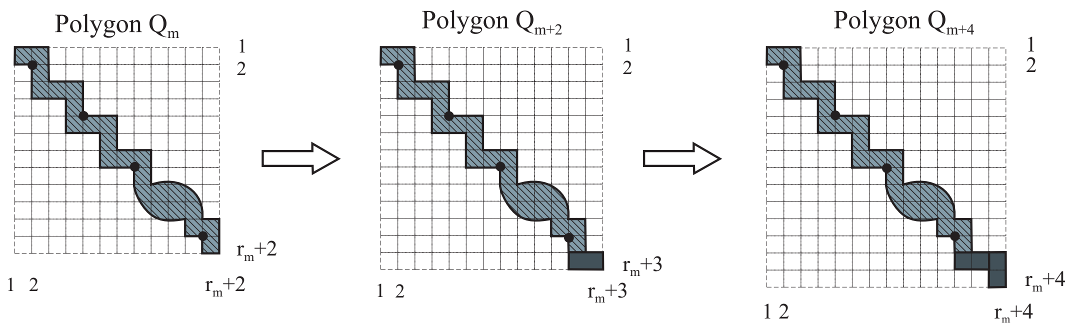



Figure 7.61: Min-Area grid $n$-ogons $Q_m$, $Q_{m+2}$ and $Q_{m+4}$.

As we can see, in any case, it is always necessary one more vertex guard, which can be placed on the vertex $v_{r_n+1,r_n+1}$. Thus, $\lceil \frac{r_m+2}{3} \rceil + 1 = \lceil \frac{r_n+2}{3} \rceil$ vertex guards are necessary to guard $P_n$.

□

Proposition 7.13 not only gives the guarantee that $\lceil \frac{r+2}{3} \rceil$ vertex guards are required to guard any MIN-AREA grid $n$-ogon with $r$ reflex vertices, but also establishes a possible positioning.

Thus, given a MIN-AREA grid $n$-ogon $P$, it was not only established the minimum number of vertex guards that is necessary to cover $P$ completely, but also it was determined in which vertices these guards must be placed. In other words, the MINIMUM VERTEX GUARD problem is solved for MIN-AREA grid $n$-ogons.

### Spiral grid $n$-ogons

Nilsson and Wood [100] have proven that *a collection of guards (mobile or stationary) cover a spiral polygon if, and only if, they see all edges of the reflex chain.* Their demonstration remains valid if guards are replaced by vertex guards, and since a spiral $n$-ogon ($n$*-vertex orthogonal polygon whose boundary can be divided into a reflex chain and a convex chain*) is a particular case of spiral polygons, it follows:

**Lemma 7.6** *A collection of vertex guards covers a spiral* $n$-ogon *if, and only if, they see all the edges of the reflex chain.*

Let $P$ be a spiral polygon having $n$ vertices, $k$ of which are reflex, and having its vertices labelled according to our previously described conventions for Spiral grid $n$-ogons (see section 7.2.1). Nilsson and Wood have also established that, for a guard to be able to see an edge of the reflex chain $e_i$, $i \in \{0, ..., k\}$, it has to be placed in a particular convex region, $CR_i$, defined in the following way:

1. if $i = 0$, they extend $e_0$ through $u_1$ until it intersects the convex chain. In this case, $CR_0$ is the region bounded by $\overline{c_{n-r}x_1}$, $x_1$ is the intersection point with the convex chain, and the subchain of the boundary of $P$ from $x_1$ to $c_{n-r}$ in counterclockwise order. See Figure 7.62 (a) for illustration.
2. if $i = k$, they extend $e_k$ through $u_k$ until it intersects the convex chain. In this case, $CR_k$ is the region bounded by $\overline{x_k u_k}$, $x_k$ is the intersection point with the convex chain, and the subchain of the boundary of $P$ from $u_k$ to $x_k$ in counterclockwise order. See Figure 7.62 (b) for illustration.
3. if $i \neq 0, k$, they extend $e_i$ through $u_i$ and $u_{i+1}$ until it intersects the convex chain. In this case, $CR_i$ is the region bounded $\overline{x'_i x_i}$, $x'_i$ and $x_i$ are the intersection points with the convex chain, and the subchain of the boundary of $P$ from $x_i$ to $x'_i$ in counterclockwise order. See Figure 7.62 (c) for illustration.

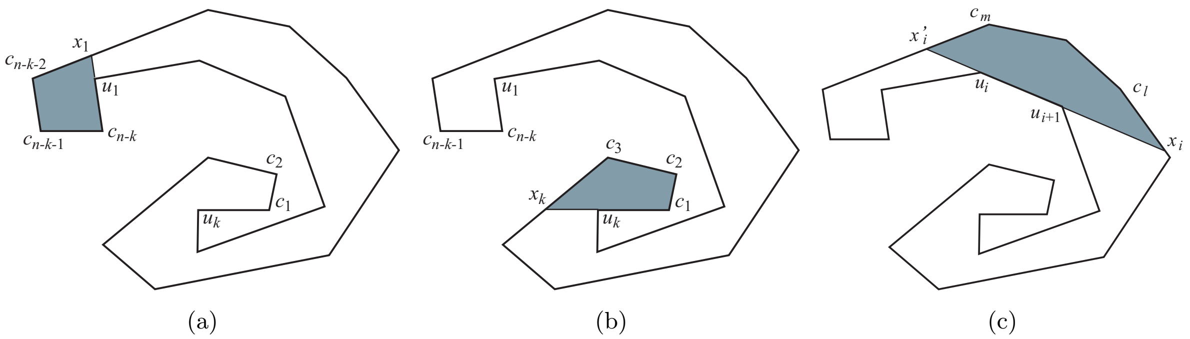



Figure 7.62: (a) $CR_0$; (b) $CR_k$ and (c) $CR_i$, with $i \neq 0, k$.

They also provided an algorithm to find the minimum number of stationary guards necessary to cover a spiral polygon. The main idea of their algorithm is: first place a guard $g_1$, in a specific position, that sees the first edge of the reflex chain and then keep on placing guards

$g_i$, in specific positions, whenever the edge $e_i$ of the reflex chain is not seen by the previously placed guard. This algorithm computes an optimum guard cover in a spiral polygon, however it does not give an explicit number of guards and it deals with guards and not vertex guards, which is a different problem. Basing on their algorithm, particularizing (for spiral $n$-ogon) and adapting (for vertex guards), we will prove that $\lfloor \frac{r}{2} \rfloor + 1$ vertex guards are necessary to cover any spiral $n$-ogon with $r$ reflex vertices.

Let $P$ be a spiral $n$-ogon with $r$ reflex vertices, the aim is to determine the minimum number of vertex guards that is needed to guard $P$. By lemma 7.6, it is only necessary to consider the visibility of the edges of the reflex chain. Moreover, being $e_i$ an edge of the reflex chain we already know that a guard, to be able to see $e_i$, it has to be placed in $CR_i$, as we are dealing with vertex guards, we can conclude, that, to be able see $e_i$, a vertex guard has to be placed on a vertex of $P$ that belongs to $CR_i$. In case of spiral $n$-ogons, these convex regions have a particular shape, they are rectangles (see [88]), and their forms are illustrated in Figure 7.63.

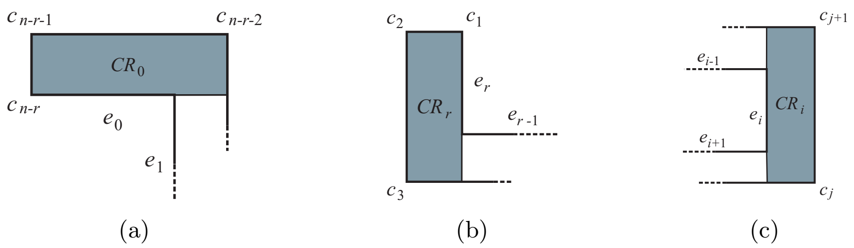



Figure 7.63: (a) $CR_0$; (b) $CR_r$ and (c) $CR_i$, $i \in \{1, ..., r-1\}$.

**Lemma 7.7** *Let $P$ be a spiral $n$-ogon with $r$ reflex vertices. A vertex guard that sees the edge $e_i$, with $0 < i < r$, can also see $e_{i-1}$ or $e_{i+1}$, but not both.*

*Proof:* Let $e_i \equiv \overline{u_i u_{i+1}}$ $(0 < i < r)$ be an edge of the reflex chain. For a vertex guard to be able to see $e_i$ it has to be placed on a vertex of $P$ that belongs to $CR_i$. As we saw before, being $P$ a spiral $n$-ogon, the only vertices of $P$ that belong to $CR_i$ are: $u_i, u_{i+1}, c_j$ or $c_{j+1}$ (see Figure 7.63 (c)). Thus, the guard has to be placed on one of these vertices. If one of the vertices $u_i$ or $c_{j+1}$ is chosen, then the vertex guard also sees $e_{i-1}$, but it does not see $e_{i+1}$. If one of the vertices $u_{i+1}$ or $c_j$ is selected, he also sees $e_{i+1}$, but it does not see $e_{i-1}$.

□

In the previous lemma it was proved that a vertex guard that sees an edge of the reflex chain, different from the first one and from the last one, only manages to see one of its adjacent edges. Let us see what happens with a vertex guard that sees the first or the last edge of the reflex chain:

(1) for a guard to be able to see $e_0 \equiv \overline{c_{n-r}u_1}$ it has to be placed on a vertex of $P$ that belongs to $CR_0$. As we saw before, being $P$ a spiral $n$-ogon, the only vertices of $P$ that belong to $CR_0$ are: $c_{n-r-2}, c_{n-r-1}, c_{n-r}$ and $u_1$. Thus, the guard has to be placed on one of these vertices. Of these positions it can be chosen one that also sees $e_1$, which is $c_{n-r-2}$ or $u_1$ (see Figure 7.64 (a)).

(2) for a guard to be able to see $e_r \equiv \overline{u_r c_1}$ it has to be placed on a vertex of $P$ that belongs to $CR_r$. As we saw before, being $P$ a spiral $n$-ogon, the only vertices of $P$ that belong to $CR_r$ are: $u_r, c_1, c_2$ and $c_3$. Thus, the guard has to be placed on one of these vertices. Of these positions it can be chosen one that also sees $e_{r-1}$, which is $c_3$ or $u_r$ (see Figure 7.64 (b)).

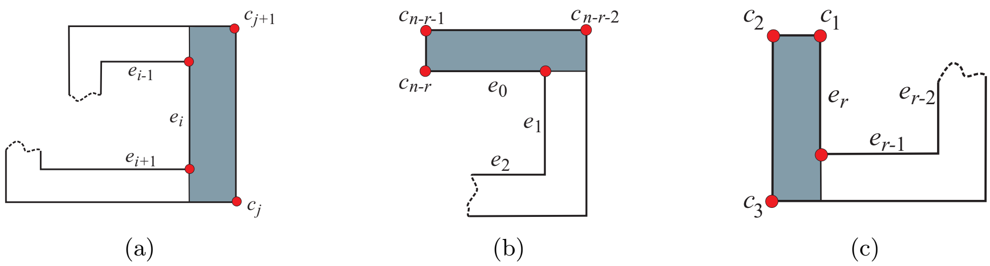



Figure 7.64: (a) $CR_i$, $i \in \{1, ..., r-1\}$; (b) $CR_0$ and (c) $CR_r$.

Therefore, from lemma 7.7, (1) and (2), it follows that a vertex guard sees at most two edges of the reflex chain.

**Proposition 7.14** $\lfloor \frac{r}{2} \rfloor + 1$ *vertex guards are necessary to cover any spiral $n$-ogon with $r$ reflex vertices.*

*Proof:* Let $P$ be a spiral $n$-ogon with $r$ reflex vertices. By definition, $\partial P$ can be divided into a reflex chain and a convex chain. The reflex chain has $r+1$ edges: $e_0, e_1, e_2, ..., e_r$. Two cases can take place: $r$ is odd or $r$ is even.

1. If $r$ is odd, place the guards at the following vertices: $u_1, u_3, ..., u_{r-2}, u_r$, i.e., $u_{1+2k}$, with $k = 0, 1, ..., \frac{r-1}{2}$ (see Figure 7.65).

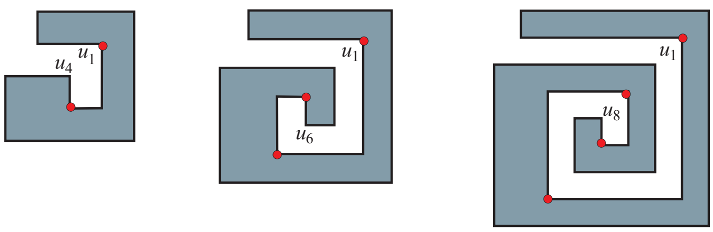



Figure 7.65: Spiral $n$-ogons with $r$ odd.

These guards see all the edges of the reflex chain. In fact, $u_{1+2k}$, $k \in \{0, 1, ..., \frac{r-1}{2}\}$, is the reflex vertex common to edges $e_{2k}$ and $e_{1+2k}$, thus $e_{2k}$ and $e_{1+2k}$ are seen by the vertex guard placed on $u_{1+2k}$. Consequently, $e_0$ and $e_1$ are seen by the vertex guard placed on $u_1$, $e_2$ and $e_3$ are seen by the vertex guard placed on $u_3$, ..., and $e_{r-1}$ and $e_r$ are seen by the vertex guard placed on $u_r$. Therefore, these guards cover $P$ since they see all the edges of the reflex chain, and by lemma 7.6 this is enough.
Thus, $\frac{r-1}{2} + 1 = \frac{r+1}{2}$ vertex guards cover $P$. To see that less than $\frac{r+1}{2}$ vertex guards do not cover $P$, assume the contradiction. Suppose that there is a set of vertex guards $S$, with $|S| \leq \frac{r+1}{2} - 1$, that cover $P$. We know that each vertex guard see at most 2 edges of the reflex chain, thus at most $2 \times |S| \leq r + 1 - 2 = r - 1$ edges are seen by these vertex guards. As the reflex chain has $r + 1$ edges, at least two edges of the reflex chain are not seen, as a consequence $P$ is not covered by the vertex guards in $S$.

2. If $r$ is even, place the guards at the following vertices: $u_1, u_3, ..., u_{r-1}, c_1$, i.e., $u_{1+2k}$, with $k = 0, 1, ..., \frac{r}{2} - 1$, and $c_1$ (see Figure 7.66).

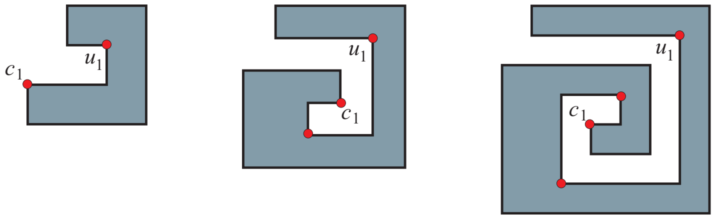



Figure 7.66: Spiral $n$-ogons with $r$ even.

These guards see all the edges of the reflex chain. In fact, as in the previous case, $u_{1+2k}$, $k \in \{0, 1, ..., \frac{r}{2} - 1\}$, is the reflex vertex common to edges $e_{2k}$ and $e_{1+2k}$, thus $e_{2k}$ and $e_{1+2k}$ are seen by the vertex guard placed on $u_{1+2k}$. Consequently, $e_0$ and $e_1$ are seen by the vertex guard placed on $u_1$, $e_2$ and $e_3$ are seen by the vertex guard placed on $u_3$, ..., and $e_{r-2}$ and $e_{r-1}$ are seen by the vertex guard placed on $u_{r-1}$. Consequently, the guards placed on $u_1, u_3, ..., u_{r-1}$ see the edges $e_0, e_1, ..., e_{r-1}$ of the reflex chain. Finally, $c_1$ is an endpoint of $e_r$, thus the guard placed on $c_1$ see $e_r$. Therefore, the guards placed on $u_1, u_3, ..., u_{r-1}, c_1$ cover $P$ since they see all the edges of the reflex chain, and by lemma 7.6 this is enough.
Thus, $\frac{r}{2} + 1$ vertex guards cover $P$. To see that less than $\frac{r}{2} + 1$ vertex guards does not cover $P$, assume the contradiction. Suppose that there is a set of vertex guards $S$, with $|S| \leq \frac{r}{2}$, that covers $P$. We know that each vertex guard sees at most 2 edges of the reflex chain, thus at most $2 \times |S| \leq 2 \times \frac{r}{2} = r$ edges are seen by these vertex guards. As the reflex chain has $r + 1$ edges, at least one edge of the reflex chain is not seen, as a consequence $P$ is not covered by the vertex guards in $S$.

Concluding, if $r$ is odd it will be necessary $\frac{r+1}{2}$ vertex guards to cover $P$; and if $r$ is even it will be necessary $\frac{r}{2}+1$ vertex guards to cover $P$. In any case, $\lfloor \frac{r}{2} \rfloor + 1$ vertex guards are necessary to cover $P$.

□

Since any SPIRAL grid $n$-ogon is a spiral $n$-*ogon*, proposition 7.14 not only gives the guarantee of that $\lfloor \frac{r}{2} \rfloor + 1$ vertex guards are necessary to cover a SPIRAL grid $n$-ogon, but also establishes a possible positioning for these guards, which is:

$$\begin{cases} u_{1+2k}, k = 0, 1, ..., \frac{r-1}{2} & \text{for } r \text{ odd} \\ \\ u_{1+2k}, k = 0, 1, ..., \frac{r}{2} - 1 \quad \text{and } c_1 & \text{for } r \text{ even} \end{cases}$$

As it is already known that $\lfloor \frac{n}{4} \rfloor = \lfloor \frac{r}{2} \rfloor + 1$ vertex guards, or fewer, are required to cover any $n$-ogon, we can conclude that SPIRAL grid $n$-ogons give us the worst scenario within the THIN grid $n$-ogons.

### 7.3.2 Maximum Hidden Vertex Set Problem on grid n-ogons

Remember that the MAXIMUM HIDDEN VERTEX SET problem, MHVS($P$) problem, asks for a hidden vertex set $H \subset V_P$ of maximum cardinality. In this subsection the MVHS($P$) problem, being $P$ a THIN grid $n$-ogon, will be studied.

#### 7.3.2.1 Thin grid $n$-ogons

Let $P$ be a THIN grid $n$-ogon and $S = u_1 u_2 \ldots u_m$ its skeleton, where $m = \frac{n}{2}$. Let us assume, without loss of generality, that the first edge of $S$, $\overline{u_1 u_2}$, is horizontal and that $u_2$ is to the right of $u_1$. Note that, the $\partial P$ consists of two joined polygonal chains, $C_1$ and $C_2$, "parallel" to $S$, where the first edge of $C_1$ is a bottom edge and the first edge of $C_2$ is a top edge. Notice that, $C_1$ and $C_2$ can be expressed as ordered sequences of vertices $C_1 = v_1^1 v_2^1 \ldots v_m^1$ and $C_2 = v_1^2 v_2^2 \ldots v_m^2$, where $v_i^1$ denotes the $i^{th}$ vertex of $C_1$ and $v_i^2$ denotes the $i^{th}$ vertex of $C_2$ (see Figure 7.67).

This way, $\partial P = C_1 \cup \overline{v_m^1 v_m^2} \cup C_2 \cup \overline{v_1^2 v_1^1}$. Observe, also, that, if $S$ is traversed, starting at vertex $u_1$, $C_1$ is always on the right of $S$ and $C_2$ on the left.

To each vertex of the skeleton we correspond two vertices of the polygon, one in $C_1$ and another one in $C_2$. That is, to $u_i \in S$ we correspond the vertices $v_i^1 \in C_1$ and $v_i^2 \in C_2$. And to each edge of the skeleton we correspond two parallel edges of the polygon, one in $C_1$ and another one in $C_2$. That is, to $\overline{u_i u_{i+1}} \in S$ we correspond the edges $\overline{v_i^1 v_{i+1}^1} \in C_1$ and $\overline{v_i^2 v_{i+1}^2} \in C_2$. Note that, by construction of the skeleton, we can easily see that any point of $\overline{v_i^1 v_{i+1}^1}$ sees any point of $\overline{v_i^2 v_{i+1}^2}$.

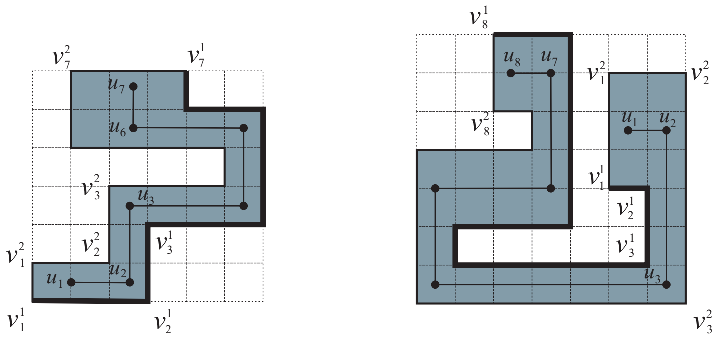



Figure 7.67: Two THIN grid $n$-ogons, its skeletons and the chains $C_1$ and $C_2$ ($C_1$ in bold).

Now, for each $u_{2k-1} \in S$ with $k = 1, \ldots, \lceil \frac{n}{4} \rceil$, we mark a hidden vertex in $P$, in the following way: for $k = 1$ we mark $v_1^1$; for $k \neq 1$ we mark $v_{2k-1}^1$ or $v_{2k-1}^2$, depending if $v_{2k-2}^1$ is reflex or convex, respectively (see Figure 7.68, for illustration).

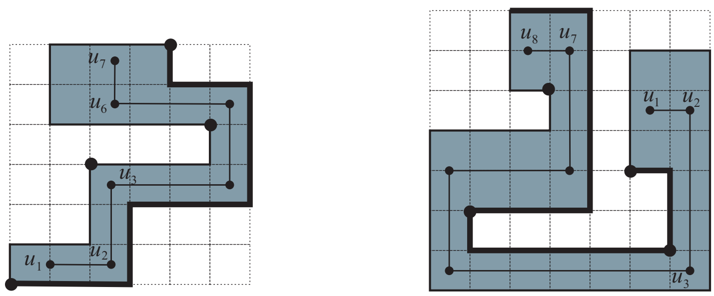



Figure 7.68: Two THIN grid $n$-ogons and marked hidden vertices ($C_1$ in bold).

Note that, the $\lceil \frac{n}{4} \rceil$ marked vertices form a hidden vertex set, since each time that a new vertex is marked as hidden it can be guaranteed that it does not see any of the vertices that previously had been marked as hidden. In fact, for $k = 1$ it is trivial. For $k \neq 1$, there are two cases, depending if $v_{2k-2}^1$ is reflex (*Case* 1) or convex (*Case* 2), as we can se in Figure 7.69.

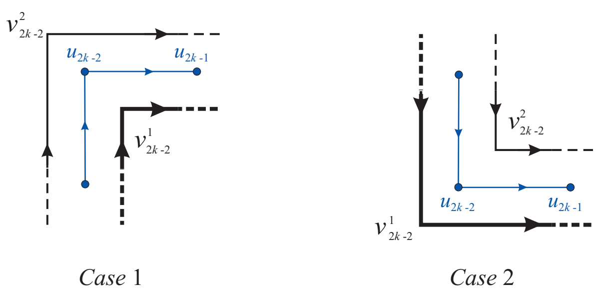



Figure 7.69: On the left $v_{2k-2}^1$ is reflex and on the right it is convex.

In Case 1 the vertex that is marked as hidden is the vertex $v_{2k-1}^1$ and in Case 2 it is the vertex $v_{2k-1}^2$. In both cases the marked vertex does not see any of the already marked as hidden, since, of the already "visited" vertices, this one only sees $v_{2k-2}^1$ and $v_{2k-2}^2$ (see Figure

7.70).

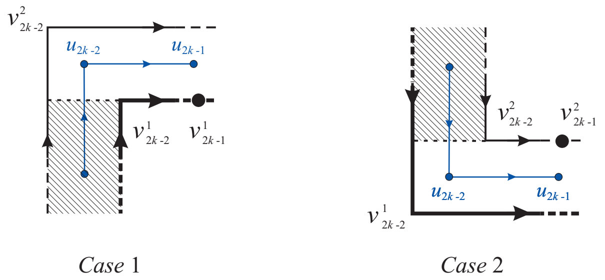



Figure 7.70: The shaded zones are not visible by the marked vertices.

Observe that, if the vertices $v^2_{2k-1}$ (in Case 1) and $v^1_{2k-1}$ (in Case 2) are marked as hidden, it can not be guaranteed that they do not see any of the vertices already marked as hidden, since they see more backwards (see Figure 7.71). Therefore, lemma 7.8 follows.

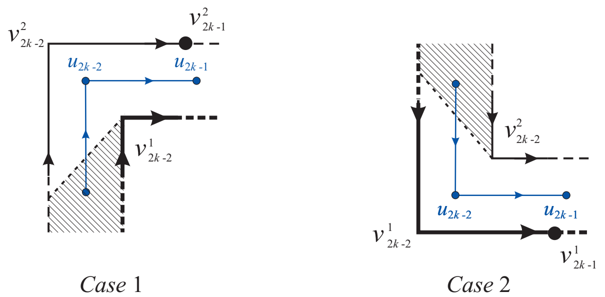



Figure 7.71: The shaded zones are not visible by the marked vertices.

**Lemma 7.8** *For any* THIN *grid $n$-ogon there is a hidden vertex set $H$ and $|H| = \lceil \frac{n}{4} \rceil$.*

Now, it will be proven that the maximum cardinality of an hidden vertex set in a THIN grid $n$-ogon is $\lceil \frac{n}{4} \rceil$. To prove this result lemma 7.9 is introduced.

**Lemma 7.9** *Let $P$ be a* THIN *grid $n$-ogon and $S$ its skeleton. To two consecutive vertices in $S$ it corresponds, at most, one hidden vertex in $P$.*

*Proof:* Let $u_i$ and $u_{i+1}$ be two consecutive vertices in $S$. The corresponding vertices in $P$ are $v_i^1$, $v_i^2$, $v_{i+1}^1$ and $v_{i+1}^2$, respectively. By the correspondence previously established, any point of the edge $\overline{v_i^1 v_{i+1}^1}$ sees any point of the edge $\overline{v_i^2 v_{i+1}^2}$, in particular the vertices of the edges. Therefore, $v_i^1$ sees $v_i^2$ and $v_{i+1}^2$; and $v_{i+1}^1$ sees $v_i^2$ and $v_{i+1}^2$. And it is obvious, that $v_i^1$ sees $v_{i+1}^1$ and that $v_i^2$ sees $v_{i+1}^2$.

□

**Theorem 7.1** *Let $P$ be a* THIN *grid $n$-ogon. The maximum cardinality of an hidden vertex set in $P$ is $\lceil \frac{n}{4} \rceil$.*

*Proof:* By lemma 7.8 there is a hidden vertex set in $P$ with cardinality $\lceil \frac{n}{4} \rceil$. Suppose, now, that there is a hidden vertex set $H$, with $|H| \geq \lceil \frac{n}{4} \rceil + 1$. Since the skeleton of $P$ has $\lceil \frac{n}{4} \rceil$ vertices with index odd, this implies that a hidden vertex has to be placed on a vertex of $P$ that corresponds to a vertex of the skeleton with even index. In other words, it means that two hidden vertices will have to be placed on two vertices of $P$ that correspond to two consecutive vertices of the skeleton, in contradiction with lemma 7.9. □

## 7.4 Concluding Remarks

In this chapter, it was studied a particular type of orthogonal polygons, the grid $n$-ogons, and presented some results related to them, including some guarding and hiding problems on different subclasses of this type of polygons.

As to the guarding problems, related to the grid $n$-ogons, the one which motivates us most is the MVGS($P$) problem. It was shown that to cover any FAT grid $n$-ogon it is always sufficient two vertex guards. Furthermore, it was established where these guards could be placed. Concerning the THIN grid $n$-ogons it was proven that to cover a MIN-AREA grid $n$-ogon and a SPIRAL grid $n$-ogon it is always sufficient and necessary $\lceil \frac{n}{6} \rceil$ and $\lfloor \frac{n}{4} \rfloor$ vertex guards, respectively. Moreover, it was shown where these guards must be placed.

Regarding the hiding problem, the one that motivates us most is the MHVS($P$) problem. It was proved that the maximum cardinality of a hidden vertex set in a THIN grid $n$-ogon is $\lceil \frac{n}{4} \rceil$. Moreover, a possible positioning for these hidden vertices was established.

It was, also, established a possible classification for THIN grid $n$-ogons, as a step to launch an expression that relates $n$ to $|\text{THIN}(n)|$. This classification/taxonomy was done by resorting to the skeleton structure of THIN grid $n$-ogons and their corresponding binary representation.

However, there are still some open problems related to THIN grid $n$-ogons, namely:

(1) How many elements THIN grid $n$-ogons does each class have? (open problem 7.1)

(2) Will it be possible to find an algorithm that generates all THIN grid $n$-ogons of the same class? (open problem 7.2)

(3) Does it exist any expression that relates $n$ to $|\text{THIN}(n)|$? (open problem 9.2)

Another open problem is:

**Open Problem 7.4** *Given a* THIN *grid $n$-ogon what is the minimum number of vertex guards needed to cover it?*

# Chapter 8

# Spiral and Histogram Polygons

In this chapter are studied the Maximum Hidden Set problem, MHS($P$), and the Maximum Hidden Vertex Set problem, MHVS($P$), on two classes of polygons, the spiral and histogram polygons. The chapter is divided in two sections. In section 8.1 the MHS($P$) and MHVS($P$) problems, being $P$ a spiral polygon (subsection 8.1.1) and a histogram polygon (subsection 8.1.2), are studied. Particularly, concerning the MHVS($P$) problem, in subsection 8.1.1 tight bounds for the maximum number of hidden vertices in a spiral polygon are determined and a linear algorithm that places a hidden vertex guard set $H$ on a spiral polygon $P$ is developed (which we strongly believe that is the solution for the MHVS($P$) problem). In section 8.2 some conclusions are presented.

Let mention that some of the results appearing in this chapter have been published in [20]

## 8.1 Maximum Hidden Vertex Set and Maximum Hidden Set Problems

### 8.1.1 Spiral Polygons

Remember that the reflex vertices of spiral polygon $P$ form a single chain of consecutive vertices (see Definition 1.4 in Chapter 1 and Figure 8.1). In this subsection the problems of hiding points and vertices on this class of polygons will be studied.

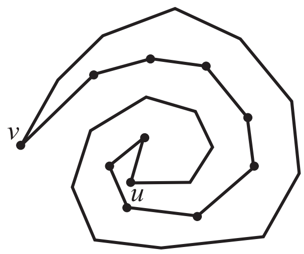



Figure 8.1: An example of a spiral polygon with its reflex chain.

First of all, combinatorial aspects of hiding vertices on a spiral polygon $P$ are solved. The following theorem relates the number of reflex $r$ of $P$ and the maximum cardinality of a set of hidden vertices $h$ on $P$.

**Theorem 8.1** *If $P$ is a spiral polygon with $r$ reflex vertices, then the maximum number of hidden vertices, $h$, verifies $\lceil \frac{r}{2} \rceil + 1 \leq h \leq r + 1$.*

*Proof:* The lower bound $\lceil \frac{r}{2} \rceil + 1 \leq h$ is obtained verifying that, if only the vertices of the reflex chain are considered, then they can be always marked hidden alternately (as shown in Figure 8.2 (a)). That is, every other vertex of the reflex chain is marked as hidden. Therefore, in any spiral polygon with $r$ reflex vertices at least $\lceil \frac{r}{2} \rceil + 1$ vertices can be hidden.

On the other hand, as shown by Shermer in [117], any polygon $P$ with $r$ reflex vertices can be decomposed into $r + 1$ convex pieces and consequently admits at most $r + 1$ hidden points. In the particular case of spiral polygons, this bound is achieved on vertices, as shown in the polygon of Figure 8.2 (b).

(a) (b)

Figure 8.2: Bounds for $h$. Black dots represent hidden vertices.

□

Now, an algorithm that places a hidden vertex set $H$ on a spiral polygon $P$ will follow.

### Algorithm

By definition, the boundary of a spiral polygon can be divided into a reflex chain and a convex chain. Denote by $C$ and $R$ the convex chain and the reflex chain, respectively. And denote by $u$ and $v$ the first and the last vertex of $C$, respectively. The proposed algorithm runs simultaneously through both chains, from $v$ to $u$, adding in every step a vertex to the set of hidden vertices $H$. The fundamental idea is to advance from $v$ to $u$ by both chains marking as hidden in each step the vertex that illuminates less the convex chain in the advance direction.

Let $P$ be a spiral polygon with $n$ vertices, denoting its reflex vertices by $\{u_1, u_2, \ldots, u_r\}$ and the vertices of its convex chain $C$ by $\{u = c_1, c_2, \ldots, v = c_{n-r}\}$, the description of the algorithm will follow.

**Algorithm 8.1** Algorithm to place hidden vertices

**Input:** A spiral polygon $P$ with $n$ vertices.

**Output:** $H \subset V_P$, a set of hidden vertices.

1. Mark as hidden the last vertex of the convex chain, $v \in H$;
2. Advance, simultaneously, through the chains $C$ and $R$ from $v$ to $u$. Let $u_k$ and $c_j$ be the first vertices of the chains $R$ and $C$, respectively, not visible from the last hidden vertex added to $H$;
3. If $c_j$ sees the next vertex of the concave chain $u_{k+1}$, then mark as hidden the vertex $u_k$; otherwise mark as hidden the vertex $c_j$ (see Figure 8.3);
4. Repeat the above process from step 2 until the vertex $u$ is reached.

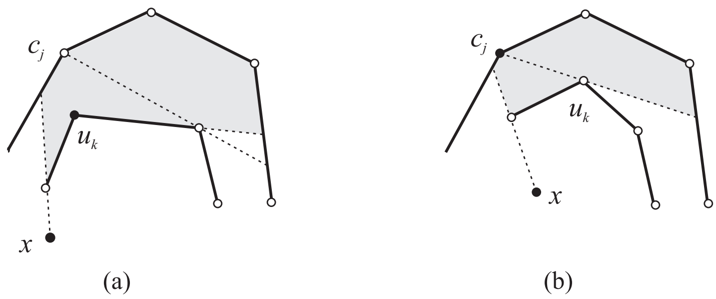



Figure 8.3: Placement of hidden vertices (the black dots represent vertices marked as hidden).

Note that the described algorithm induces a partition on the spiral polygon (see Figure 8.4). In fact, for each reflex vertex $u_k$ marked as hidden in step 3, it is considered the segment between $u_{k+1}$ and $C$ determined by the ray $\overrightarrow{u_k u_{k+1}}$, and, for each convex vertex $c_j$ marked as hidden in step 3, it is considered the segment between $u_k$ and the convex chain $C$ determined by the ray $\overrightarrow{c_j u_k}$. These segments allow to decompose the polygon in pieces of two types, $A$ (see Figure 8.3 (a)) and $B$ (see Figure 8.3 (b)). The pieces of type $A$ have two edges of the chain $R$ and the common vertex of these edges is marked as hidden. In the pieces of type $B$ there is only one edge of $R$ and the hidden vertex is on the convex chain.

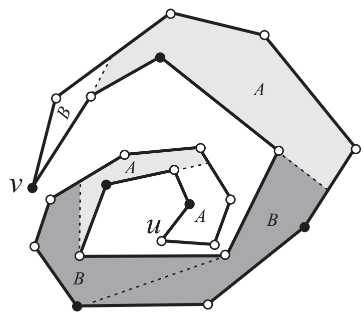



Figure 8.4: Decomposition into pieces $A$ and $B$.

Observe also that the algorithm constructs a hidden vertex set $H$ with an element in each of the pieces of the previously described partition (see Figure 8.4). We strongly believe that $H$ is an hidden vertex set of maximum cardinality.

**Conjecture 8.1** *Given a spiral polygon $P$, the previous algorithm obtains a set of hidden vertices $H$ of maximum cardinality.*

To prove this conjecture, we intend to use the partition of $P$ in pieces of type $A$ and $B$ to show that any other hidden vertex set $H^*$ verifies $|H^*| \leq |H|$ ($H$ is the algorithm output). If this conjecture is true, then we have an algorithm that solves the MHVS($P$) problem in linear time, being $P$ a spiral polygon. In fact, the described algorithm is linear: the visibility from the hidden vertices in step 2 is performed in $\mathcal{O}(n)$ because the visibility of each vertex is detected in constant time and the algorithm advances, without setback, by the reflex and convex chains. For the same reason, step 3 is also performed in linear time.

**Observation:** If the hidden points are not necessarily placed on vertices, then it is always reached the maximum value allowed for the number of hidden points. In a spiral polygon $P$ with $r$ reflex vertices, $r + 1$ points can always be hidden, since it is enough to place a hidden point in every side of the reflex chain (see Figure 8.5).

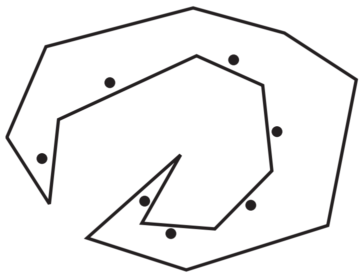


Figure 8.5: Hidden points on a spiral polygon.

### 8.1.2 Histogram Polygons

In this work only vertical histograms are considered. These type of polygons are sometimes used as pieces in the decomposition of orthogonal polygons. A *vertical histogram* $P$ is an orthogonal polygon with an horizontal edge, called the *base* of $P$, such that every point of $P$ is visible from a point of its base (see Figure 8.6). An horizontal edge whose vertices are reflex is designated by *fund edge*. The number of these edges determines the solution to the problem MHVS($P$), where $P$ is a histogram without collinear horizontal edges, as it is shown in the following theorem.

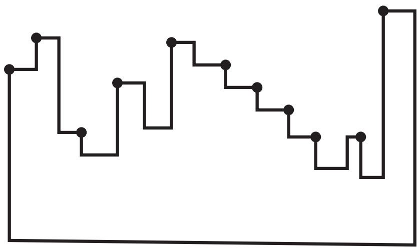



Figure 8.6: Histogram polygon.

**Theorem 8.2** *If $P$ is a histogram polygon without collinear horizontal edges, $r$ reflex vertices and $p$ fund edges; then the maximum number $h$ of hidden vertices in $P$ is $h = r - (p - 1)$.*

*Proof:* The demonstration will be done by induction on $p$.

- **Base Case** $p = 0$: In this case, the histogram is a (vertical) *pyramid*, that is, a vertical histogram that is monotone with respect to the $y$-axis (see Figure 8.7). In this case $h = r + 1$ (convex) vertices can be hidden, one in each horizontal edge different from the base.

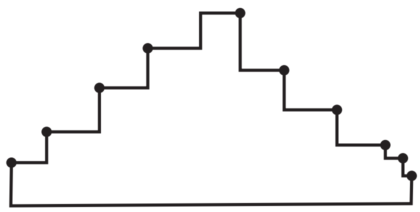


Figure 8.7: Pyramid polygon.

- **Inductive Step**: Assuming that the result is true for histograms with $p$ fund edges, let us demonstrate that it is also true for polygons with $p + 1$ fund edges. Draw an horizontal segment at the fund edge the closest to the base. In this way $P$ is decomposed into two histograms $P_1$ and $P_2$, and a rectangle $R$ (see Figure 8.8). Denote by $h_i$, $r_i$ and $p_i$ ($i = 1, 2$) the number of hidden vertices in $P_i$, the number of reflex vertices of $P_i$ and the number of edge funds of $P_i$, respectively. Note that, $r_1 + r_2 = r - 2$ and $p_1 + p_2 = p - 1$.
  By induction hypothesis

$$h_1 = r_1 - (p_1 - 1) \text{ and } h_2 = r_2 - (p_2 - 1)$$

thus,

$$h_1 + h_2 = (r_1 + r_2) + (p_1 + p_2) + 2 = (r - 2) - (p - 1) + 2 = r - (p - 1).$$

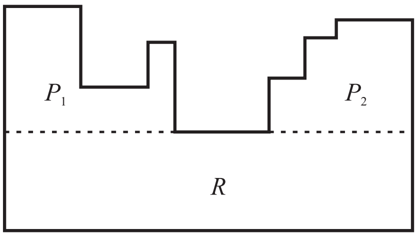



Figure 8.8: Histogram decomposition.

Besides, on $R$ cannot be placed any hidden vertex, since they are all visible from some of the histograms, then

$$h = r - (p - 1).$$

Moreover, an hidden vertex set $H$ of cardinal $h$ is obtained by placing a hidden point in the convex vertex of every horizontal edge, which is neither the base nor a fund. In the horizontal edges with two convex vertices only one point is placed in one of them. In the histogram illustrated in Figure 8.6 the hidden vertices are represented by black dots

□

**Observation:** If we hide points in histograms without horizontal collinear edges, it also becomes possible, as it happens on spiral polygons, to reach the possible maximum. By hiding one point in every horizontal edge we can hide $h = r + 1$ points being $r$ the number of reflex vertices (see Figure 8.9).

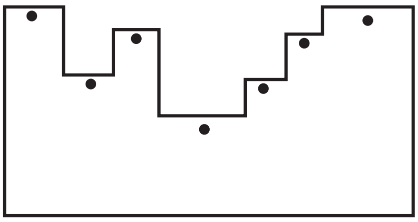


Figure 8.9: Hidden points on histograms polygons.

## 8.2 Concluding Remarks

In this chapter the MHVS($P$) and MHS($P$) problems, where $P$ is a spiral or a histogram polygon, were studied. In subsection 8.1.1 these problems were discussed for spiral polygons, and a linear algorithm that places a hidden vertex guard set $H$ on a spiral polygon $P$ (which we believe that is the solution for the MHVS($P$) problem) was described. Besides, it was determined tight bounds for the maximum number $h$ of hidden vertices in a spiral polygon with $r$ reflex vertices. In subsection 8.1.2 it was analyzed the histogram polygons without collinear horizontal edges, for which it was obtained the maximum cardinal $h$ of hidden vertices that verifies $h = r - (p - 1)$, where $r$ is the number of reflex vertices and $p$ is the number of fund edges of the histogram polygon.

Note that, if a histogram polygon has collinear edges we know that $h \leq r - (p - 1)$. However, the following problem remains open:

**Open Problem 8.1** *Given a histogram polygon $P$ with collinear edges, what is the maximum number of points and vertices that can be hidden on $P$?*

# Chapter 9

# Conclusions

This thesis studied several visibility problems, particularly guarding and hiding problems on polygons. The addressed guarding problems were: the Minimum Vertex Guard Set, Minimum Vertex Floodlight Set and Minimum Vertex $k$-Modem Set problems, which were denoted by MVGS($P$), MVFS($P$) and MVkMS($P$), respectively, where $P$ is a polygon. In relation to hiding problems, the following problems were considered: the Maximum Hidden Set and the Minimum Vertex Guard Set problems, denoted by MHVS($P$) and MHVS($P$), respectively, where $P$ is a polygon.

The above problems are $\mathcal{NP}$-hard (for example, the MVGS($P$), MHVS($P$) and MHS($P$) problems) or it is strongly believed that they are $\mathcal{NP}$-hard (for example, the MVFS($P$) and MVkMS($P$) problems). This means that finding exact and efficient methods to solve them is very unlikely. Thus, they were studied according to two lines of investigation: (1) the development of algorithms that establish approximate solutions and (2) the determination of optimal solutions on special classes of polygons. These two research lines are discussed in Part I and Part II.

**Part I**

The first part of this thesis proposed approximation algorithms to tackle the MHVS($P$), MVGS($P$), MVFS($P$) and MVkMS($P$) problems. Since metaheuristics and hybrid metaheuristics methods have been little explored in solving visibility problems, the focus was given to them. In this way, one of the main objectives of this work was to study how these methods behave when applied to this kind of problems. Although there are several different metaheuristic methods, the Simulated Annealing (SA) and the Genetic Algorithms (GAs) metaheuristics and hybridizations of these both were chosen because SA and GAs are well-known trajectory and population-based methods, respectively. Furthermore, they are widely used in solving combinatorial optimization problems.

All the proposed approximation algorithms were implemented in C/C++ and these implementations use the Computational Geometry Algorithms Library CGAL. In the various chapters SA and GAs methods were studied and compared with different parameter values, and several conclusions were presented. Therefore, the conclusions presented here are related to the selected method/strategy for each metaheuristic in each chapter. Some general conclusions, drawn from the performed computational experiences and the statistical studies, are presented in the following.

Concerning the solutions obtained by each method it can be concluded that:

($i$) *Metaheuristics (SA and GAs) and Greedy Algorithms.*
The SA metaheuristic showed to be better than or equal to the greedy algorithms; unlike the GA strategy, which does not improve the solutions given by the greedy algorithms. Particularly, for the MHVS($P$) problem on orthogonal polygons, the SA strategy is significantly better than the two implemented greedy strategies, but it is equivalent to the second one when applied to arbitrary polygons (see Chapter 3, subsection 3.2.1). In relation to the MVGS($P$) problem the two metaheuristic strategies do not improve the solutions obtained by the greedy strategies.

($ii$) *SA and GAs metaheuristics.*
The SA metaheuristic showed to obtain better or equal solutions than the GAs metaheuristic.
Particularly, for the MHVS($P$) and MVFS($P$) problems the SA strategy obtains significantly better solutions than the GA strategy, in contrast to the MVGS($P$) problem where the solutions obtained by the two strategies can be considered "equal", both for orthogonal and arbitrary polygons.

($iii$) *Non-hybrid methods and Hybrid metaheuristics.*
Recall that two hybrid metaheuristics were developed: the first one uses a SA strategy as a genetic operator of a GA method and the second one uses a SA strategy to generate the initial population of a GA. The first hybrid metaheuristic is always better than any other non-hybrid method and it is better than or equal to the second hybrid metaheuristic.
Regarding the two hybrid metaheuristics, for the MVGS($P$) problem the solutions obtained are not significantly different, both for orthogonal and arbitrary polygons. However, for the MVFS($P$) problem the first hybrid metaheuristic is always better than the second one.

The following four tables summarize the results obtained in the first part of this work. The tables show the studied problems and the references in this dissertation. For each problem, the tables present the strategy that performs best, the obtained solution and the corresponding approximation factor and, finally, the well-known combinatorial bounds if they exist.

| Problem | Arbitrary Polygons | | | | |
|---|---|---|---|---|---|
| | Strategy | Approximate Solution | Approximate Ratio | Combinatorial Bound | Reference |
| MHVS($P$) | SA | $\lceil \frac{n}{3.74} \rceil$ | 1.62 | $\lceil \frac{n}{2} \rceil$ | Section 3.4.1 |
| MVGS($P$) | Hybrid | $\lceil \frac{n}{6.64} \rceil$ | 1.66 | $\lfloor \frac{n}{3} \rfloor$ | Section 4.4.1 |
| MVkMS($P, k$), $k = 2$ | Hybrid | $\lceil \frac{n}{26.10} \rceil$ | - | Unknown | Section 6.4.2 |
| MVkMS($P, k$), $k = 4$ | Hybrid | $\lceil \frac{n}{52.35} \rceil$ | - | Unknown | Section 6.4.2 |

Table 9.1: Studied problems on arbitrary polygons.

Note that, the previous table does not refer to the MVFS($P$) problem because it only applies to orthogonal polygons.

| Problem | Orthogonal Polygons | | | | |
|---|---|---|---|---|---|
| | Strategy | Approximate Solution | Approximate Ratio | Combinatorial Bound | Reference |
| MHVS($P$) | SA | $\lceil \frac{n}{3.80} \rceil$ | 1.54 | $\frac{n-2}{2}$ | Section 3.4.2 |
| MVGS($P$) | Hybrid | $\lceil \frac{n}{7.29} \rceil$ | 1.80 | $\lfloor \frac{n}{4} \rfloor$ | Section 4.4.2 |
| MVFS($P$) | Hybrid | $\lceil \frac{n}{4.29} \rceil$ | 2 | $\lfloor \frac{3n-4}{8} \rfloor$ | Section 5.4.1 |
| MVkMS($P, k$), $k = 2$ | Hybrid | $\lceil \frac{n}{27.39} \rceil$ | - | Unknown | Section 6.4.2 |
| MVkMS($P, k$), $k = 4$ | Hybrid | $\lceil \frac{n}{57.47} \rceil$ | - | Unknown | Section 6.4.2 |

Table 9.2: Studied problems on orthogonal polygons.

Since the MVkMS($P$) was also studied on monotone arbitrary polygons and on grid monotone orthogonal polygons, Tables 9.3 and 9.4 present the related results.

| Problem | Monotone Arbitrary Polygons | | | | |
|---|---|---|---|---|---|
| | Strategy | Approximate Solution | Approximate Ratio | Combinatorial Bound | Reference |
| MVkMS($P, k$), $k = 2$ | Hybrid | $\lceil \frac{n}{15.10} \rceil$ | - | $\lceil \frac{n}{6} \rceil$ | Section 6.4.1 |
| MVkMS($P, k$), $k = 4$ | Hybrid | $\lceil \frac{n}{26.80} \rceil$ | - | $\lceil \frac{n}{8} \rceil \leq G_{km}(n)$[1] $\leq \lceil \frac{n}{10} \rceil$ | Section 6.4.1 |

Table 9.3: Studied problems on monotone arbitrary polygons.

[1] $G_{km}(n)$ is an upper bound for the minimum cardinality of vertex guard set for $P$.

| Problem | Monotone grid $n$-ogons Polygons | | | | |
|---|---|---|---|---|---|
| | Strategy | Approximate Solution | Approximate Ratio | Combinatorial Bound | Reference |
| MVkMS$(P,k)$, $k=2$ | Hybrid | $\lceil \frac{n}{18.31} \rceil$ | - | Unknown | Section 6.4.2 |
| MVkMS$(P,k)$, $k=4$ | Hybrid | $\lceil \frac{n}{35.84} \rceil$ | - | Unknown | Section 6.4.2 |

Table 9.4: Studied problems on monotone grid $n$-ogons polygons.

In general, in association with the running time of the algorithms, the better the obtained solution is, the longer the used strategy takes. This was an expected scenario and it occurred with the hybrid metaheuristics. As future research, it is intended to improve the runtime of these strategies.

Remember that, being $P$ a polygon, a linear algorithm to determine the visibility polygon of $x \in P$, $Vis(P,x)$, is well-known [85]. However, an algorithm to determine the region covered by a $k$-modem located at a point $x \in P$, $Vis_k(x,P)$, is unknown up date. Thus, an algorithm that runs in $\mathcal{O}(n^2)$ time was developed and implemented to determine $Vis_k(x,P)$ for all the possible values of $k$ $(0 \leq k \leq n)$ (see Chapter 6, section 6.2), since it was necessary to solve the MVkMS$(P,k)$ problem. The developing of an algorithm to lower this computational complexity for a fixed value of $k$ is intended as future research. It is also planned to develop a method that allows to determine the approximation ratio of the algorithm implemented to tackle the MVkMS$(P)$ problem.

In conclusion, it is clear that the metaheuristics, in particular the hybrid metaheuristics, proved to be a good approach to solve the studied problems. It was given experimental evidence that they perform well in practice, on a large set of input data. All the solutions were very satisfactory in the sense that they were always close to optimal (within an approximation factor of 2 for all randomly generated instances). As a result, there are several directions for further research. It would be interesting to reduce the runtime of the already implemented strategies. It would be also interesting to develop and implement other metaheuristics (e.g., the Ant Colony System) and to explore other combinations of metaheuristics in order to improve not only the obtained solutions of the studied problems, but also the algorithms' runtime, as well as solving other $\mathcal{NP}$-hard visibility problems.

Note that there are more alternatives to explore with respect to the parameters of the SA and GAs metaheuristics, but these are almost infinite. Along this thesis, it was attempted to find references for these parameters. Nevertheless, a more exhaustive study in future investigations might improve the obtained results and the algorithms' runtime.

**Part II**

The second part of this thesis studied the MVGS$(P)$ MHVS$(P)$, MHS$(P)$ problems applied to special classes of polygons. A particular attention was given to orthogonal polygons: it was introduced a subclass of orthogonal polygons, the grid $n$-ogons, and it was presented structural properties in order to simplify the study of the problems. The hiding problems were also applied to histograms and spiral polygons.

The following three tables summarize the results obtained in the second part of the thesis. The tables show the studied problems, the obtained solutions and the references in this dissertation.

| Problem | Grid $n$-ogons Polygons | | | | | |
|---|---|---|---|---|---|---|
| | FAT grid $n$-ogons | | THIN grid $n$-ogons | | | |
| | | | MIN-AREA | | SPIRAL | |
| | Solution | Reference | Solution | Reference | Solution | Reference |
| MVGS($P$) | 2 | Section 7.3.1.1 | $\lceil \frac{n}{6} \rceil$ | Section 7.3.1.2 | $\lceil \frac{n}{4} \rceil$[1] | Section 7.3.1.2 |

Table 9.5: Guarding problem on grid $n$-ogons polygons.

[1]This solution is also valid for orthogonal spiral polygons.

| Problem | Grid $n$-ogons Polygons | |
|---|---|---|
| | THIN grid $n$-ogons | |
| | Solution | Reference |
| MVGS($P$) | $\lceil \frac{n}{4} \rceil$ | Section 7.3.2 |

Table 9.6: Hiding problem on grid $n$-ogons polygons.

Concerning the grid $n$-ogons, there are some rather difficult and challenging problems that remain open, and it would be interesting to study them as future research, namely:

**Open Problem 9.1** *Is there an expression to relate $n$ to $|\text{THIN}(n)|$?*

**Open Problem 9.2** *What is the area value of "the"* THIN *grid $n$-ogon with maximum area?*

**Open Problem 9.3** *Given a* THIN *grid $n$-ogon what is the minimum number of vertex guards needed to cover it?*

Since combinatorial bounds for the MHVS($P$) and MHS($P$) problems, being $P$ a spiral polygon or a histogram polygon, were also determined, the next table presents the obtained results.

| Problem | Spiral polygons | | Histogram polygons | |
|---|---|---|---|---|
| | Combinatorial Bound | Reference | Solution | Reference |
| MHVS($P$) | $\lceil \frac{r}{2} \rceil + 1 \leq h \leq r + 1$ | Section 8.1.1 | $h = r - (p - 1)$ | Section 8.1.2 |
| MHS($P$) | $h = r + 1$ | Section 8.1.1 | $h = r + 1$ | Section 8.1.2 |

Table 9.7: Hiding problems on spiral and histogram polygons.

**Note:** In the above table $r$ denotes the number of reflex vertex of $P$, $h$ the maximum cardinality of a set of hidden vertices for $P$ and $p$ the number fund edges of $P$.

Remember that for spiral polygons, besides the established combinatorial bounds for the maximum cardinality of a hidden vertex set $h$, it was also developed a linear algorithm to obtain a hidden vertex set, which is believed to have cardinality $h$ (see Conjecture 8.1 in Chapter 8). In other words, we think that, given a spiral polygon $P$, the developed algorithm in Chapter 8 (subsection 8.1.1) obtains a set of hidden vertices $H$ of maximum cardinality. As a future work it is intended to prove that this is true.

Concerning histogram polygons, if the polygons have not collinear horizontal edges then there is an established upper bound for $h$: $r-(p-1)$. However, the following problem remains open.

**Open Problem 9.4** *What is the maximum number of points that can be hidden on a given histogram polygon with collinear horizontal edges?*

Finally, concerning the first line of investigation the presented results allow to conclude that hybrid metaheuristics are a "good" approach to solve visibility problems. On the other hand, the second line of investigation is left with several open problems. In conclusion, the development of these two lines of research should be continued, determining "good" approximate solutions and identifying special classes of polygons for which specific algorithms can be developed.

# Bibliography


[1] Gdtoolkit. University of Rome. http://www.dia.uniroma3.it/gdt/. 3

[2] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 3, 19

[3] E. Aarts, J. Korst, and W. Michiels. Simulated annealing. In *Handbook of Approximation Algorithms and Metaheuristics (Chapman & Hall/Crc Computer & Information Science Series)*. Chapman & Hall/CRC, 2007. 27

[4] M. Abellanas, E. Alba, S. Canales, and G. Hernández. Solving the illumination problem with heuristics. In Todor Boyanov, Stefka Dimova, Krassimir Georgiev, and Geno Nikolov, editors, *Numerical Methods and Applications*, volume 4310 of *Lecture Notes in Computer Science*, pages 205–213. Springer, 2006. 5

[5] M. Abellanas, E. Alba, S. Canales, and G. Hernández. Resolución de un problema de iluminación con simulated annealing (in spanish). In *Actas de MAEB'07*, pages 771–778, Tenerife, España, 2007. 5

[6] J. Abello, V. Estivill-Castro, T. C. Shermer, and J. Urrutia. Illumination with orthogonal floodlights. In *ISAAC '95: Proceedings of the 6th International Symposium on Algorithms and Computation*, pages 362–371, London, UK, 1995. Springer-Verlag. 10

[7] S. V. Adinolfi. *Optimización geométrica y applicaciones en visisbilidad.* PhD thesis, Universitat Politècnica de Catalunya, 1997. 1, 2

[8] A. Aggarwal. *The art gallery problem: Its variations, applications, and algorithmic aspects.* PhD thesis, Johns Hopkins University, 1984. 9, 65, 67, 77

[9] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, J. Urrutia, and B Vogtenhuber. Modem illumination of monotone polygons. In *Proc.* $25^{th}$ *European Workshop on Computational Geometry EuroCG '09*, pages 167–170, Brussels, Belgium, 2009. viii, 10, 13, 135, 136, 137, 138

[10] E. Alba and G. Luque. *Parallel Metaheuristics: A New Class of Algorithms*, chapter 2. Measuring the Performance of Parallel Metaheuristics, pages 43–62. Wiley Series on Parallel and Distributed Computing. Wiley, 2005. 20

[11] Y. Amit, J.S.B. Mitchell, and E. Packer. Locating guards for visibility coverage of polygons. In *Proceedings of the Workshop on Algorithm Engineering and Experiments*, pages 1–15, 2007. 4, 49, 77, 78

[12] M. Arkin, E, J.S.B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 20–27, New York, NY, USA, 2008. ACM. 3

[13] S. Arora and C. Lund. Hardness of approximations. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS, 1996. 50, 78, 121

[14] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Trans. Comput.*, 30(12):910–914, 1981. 10

[15] David Avis and Godfried T. Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, 1981. 8

[16] A. L. Bajuelos, S. Canales, G. Hernández, and A. M. Martins. Some problems related to grid n-ogons. In *XII Spanish Workshop on Computational Geometry (EGC'07)*, pages 265–272, June 2007. 16, 159

[17] A. L. Bajuelos, S. Canales, G. Hernández, and A. M. Martins. Estimating the maximum hidden vertex set in polygons. In *ICCSA '08: Proceedings of the 2008 International Conference on Computational Sciences and Its Applications*, pages 421–432, Washington, DC, USA, 2008. IEEE Computer Society. 16, 39

[18] A.L. Bajuelos, S. Canales, G. Hernández, and A. M. Martins. Solving some combinatorial problems in grid n-ogons. In *ACS'07: Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science*, pages 151–156, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS). 16, 159

[19] A.L. Bajuelos, S. Canales, G. Hernández, and A.M. Martins. Solving some combinatorial problems in grid n-ogons. *International Journal of Mathematics and Computers in Simulation, NAUN*, 1(2):177–183, 2007. 16, 159

[20] A.L. Bajuelos, S. Canales, G. Hernández, and A.M. Martins. Escondiendo puntos en espirales e histogramas (in spanish). In *Proc. of VI Jornadas de Matemática Discretas*, pages 85–93, 2008. 16, 205

[21] A.L. Bajuelos, S. Canales, G. Hernández, and A.M. Martins. Minimum vertex guard problem for orthogonal polygons: a genetic approach. In *Proc. 10$^{th}$ WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems (MAMECTIS'08)*, pages 78–84, 2008. 16, 66

[22] A.L. Bajuelos, S. Canales, G. Hernández, and A.M. Martins. Optimizing the minimum vertex guard set on simple polygons via a genetic algorithm. *WSEAS Transactions in Information Science and Applications*, 5(11):1584–1596, 2008. 16, 66

[23] A.L. Bajuelos, S. Canales, G. Hernández, and A.M. Martins. Aproximando la iluminación por módems (in spanish). In *XIII Spanish Workshop on Computational Geometry*, pages 67–74, June 2009. 16

[24] A.L. Bajuelos, A.P. Tomás, and Marques F. Partitioning orthogonal polygons by extension of all edges incident to reflex vertices: Lower and upper bounds on the number of pieces. In *ICCSA (3)*, pages 127–136, 2004. 15, 159, 162, 163, 164, 165, 166, 189

[25] C. Blum and R. Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003. 22, 23, 24, 26, 27, 28, 31, 33, 34, 35, 36

[26] P. Bose, L. Guibas, A. Lubiw, M Overmars, D. Souvaine, and J. Urrutia. The floodlight problem. *J. Assoc. Comput. Mach*, 9:399–404, 1993. 10

[27] A. Bottino and A. Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recogn.*, 41(11):3343–3355, 2008. 4

[28] F. Busetti. Simulated annealing overview, 2003. 29, 72

[29] Blum C. and Roli A. Hybrid metaheuristics: An introduction. In *Hybrid Metaheuristics*, pages 1–30. 2008. 22, 23, 24, 27, 35, 36

[30] S. Canales. *Métodos Heurísticos en Problemas Geométricos. Visibilidad, iluminación y vigilancia*. PhD thesis, Universidad Politécnica de Madrid, Madrid, Spain, 2004. 5, 26, 27, 28, 30, 31, 34, 35

[31] J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and B. Palop. Optimal location of transportation devices. *Comput. Geom.*, 41(3):219–229, 2008. 3

[32] M. Cary, A. Rudra, A. Sabharwal, and E. Vee. Floodlight illumination of infinite wedges. *Computational Geometry*, In Press, Corrected Proof:–, 2009. 10

[33] T. Christ, M. Hoffmann, Y. Okamoto, and T. Uno. Improved bounds for wireless localization. In *SWAT*, pages 77–89, 2008. 135

[34] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975. 3, 8, 67

[35] C. Clark, Stephen M. R., and J.-C. Latombe. Motion planning for multiple mobile robot systems using dynamic networks. In *IEEE Int. Conference on Robotics and Automation*, pages 4222–4227, 2003. 3

[36] M.C. Couto, C.C. Souza, and P.J. Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *SIBGRAPI '07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 87–94, Washington, DC, USA, 2007. IEEE Computer Society. 4, 15, 159, 187

[37] M.C. Couto, C.C. Souza, and P.J. Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *WEA*, pages 101–113, 2008. 15, 159, 187

[38] Anghinolfi D. and Paolucci M. *Simulated Annealing*, chapter 1. Simulated Annealing as an Intensification Component in Hybrid Population-Based Metaheuristics, pages 1–26. IN-TECH, 2008. 38

[39] M. de Berg, O. Cheong, M. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Heidelberg, 3rd edition, 2008. 1, 2

[40] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ. http://www.cs.brown.edu/people/rt/gdbook.html. 3

[41] J. Dietel, H. Hecker, and A. Spillner. A note on optimal floodlight illumination of stages. *Information Processing Letters*, 105(4):121 – 123, 2008. 10

[42] D. Dobkin and S. Teller. Computer graphics. In *Handbook of discrete and computational geometry*, pages 1090–1116. CRC Press, Inc., Boca Raton, Florida, USA, 2004. 3

[43] D.P. Dobkin. Computational geometry and computer graphics. *Proc. IEEE*, 80:141–1, 1992. 3

[44] M. Dorigo and T. Sttzle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004. 23

[45] H. Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987. 1

[46] H. Edelsbrunner. Biological applications of computational topology. In J.E Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 1395–1412. CRC Press, Inc., Boca Raton, Florida, USA, 2004. 3

[47] H. Edelsbrunner, J. O'Rourke, and E. Welzl. Stationing guards in rectilinear art galleries. *Comput. Vision Graph. Image Process*, 270:167–176, 1984. 9

[48] A. Efrat, S. Har-Peled, and J. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *Proceedings of the 3rd International Conference on Broadband Communications, Networks and Systems (Broadnets'05)*, pages 714–723, Boston, Massachusetts, 2005. 4

[49] S. Eidenbenz. How many people can hide in a terrain? In *Lecture Notes in Computer Science 1741 (ISAAC'99*, pages 184–194, 1999. 9, 11

[50] S. Eidenbenz. *(In-)Approximability of Visibility Problems on Polygons and Terrains.* PhD thesis, Institute for Theoretical Computer Science, ETH, Zurich, 2000. 4, 39

[51] S. Eidenbenz. Finding minimum hidden guard sets in polygons: tight approximability results. *Comput. Geom. Theory Appl.*, 34(2):49–57, 2006. 9, 11

[52] S. Eidenbenz and C. Stamm. Maximum clique and minimum clique partition in visibility graphs. In *TCS '00: Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, pages 200–212, London, UK, 2000. Springer-Verlag. 50

[53] D. Eppstein, M.T. Goodrich, and N. Sitchinava. Guard placement for efficient point-in-polygon proofs. In *Symposium on Computational Geometry*, pages 27–36, 2007. 135

[54] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, September 2006. 4

[55] V. Estivill-Castro, J. O'Rourke, J. Urrutia, and D. Xu. Illumination of polygons with vertex lights. *Inf. Process. Lett.*, 56(1):9–13, 1995. 10

[56] R. Fabila-Monroy, A.R Vargas, and J. Urrutia. On modem illumination problems. XIII Spanish Workshop on Computational Geometry, June 2009. 10, 136

[57] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of cgal, a computational geometry algorithms library. *Software – Practice and Experience*, 30(11):1167–1202, 2000. Special Issue on Discrete Algorithm Engineering. 2, 3

[58] H.-Y.F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition. *IEEE Transactions on Computers*, 24(6):636–650, 1975. 174

[59] S. Fisk. A short proof of chvatal's watchman theorem. *Journal of Combinatorial Theory Series B*, 24:374+, 1978. 67

[60] E. Fogel, R. Wein, B. Zukerman, and D. Halperin. 2d regularized boolean set-operations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.2.1 edition, 2006. 19

[61] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979. 21

[62] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741, Nov. 1984. 28

[63] S. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, USA, 2007. 4

[64] S.K. Ghosh. Approximation algorithms for art gallery problems. In *Proceedings of the Canadian Information Processing Society Congress*, pages 429–434, 1987. 4

[65] Fred W. Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 114 of *International Series in Operations Research & Management Science*. Springer, January 2003. 24

[66] J.E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, USA, 2004. 1

[67] D. Henderson, S.H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In J.E Goodman and J. O'Rourke, editors, *Handbook of Metaheuristics*, pages 287–319. Springer, 2003. 26

[68] J. Hershberger. Finding the visibility graph of a polygon in time proportional to its size. *Algorithmica*, 4:141–155, 1989. 42

[69] S. Hert, M. Hoffmann, L. Kettner, and S. Schnherr. Geometric object generators. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.2.1 edition, 2006. 19

[70] J.H. Holland. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992. 1st edition: 1975, The University of Michigan Press, Ann Arbor. 29

[71] R. Honsberger. *Mathematical Gems II*. Mathematical Association of America, 1976. 3

[72] J. Huang. Visibility problems occurring in radiation treatment planning. Master's thesis, Ottawa-Carleton Institute for Computer Science, School of Computer Science, Carleton University, Ottawa, Ontario, Canada, 2001. 3

[73] F. Hurtado. *Problemas Geométricos de Visibilidad*. PhD thesis, Universitat Politcnica de Catalunya, 1993. 11

[74] F. Hurtado, O. Serra, and J. Urrutia. Hiding points in arrangements of segments. *Discrete Math.*, 162(1-3):187–197, 1996. 11

[75] L. Ingber. Very fast simulated re-annealing. *Mathematical Computer Modeling*, 12(8):967–973, 1989. 28

[76] Sastry K., Goldberg D., and Kendall G. *Search methodologies : introductory tutorials in optimization and decision support techniques*, chapter 4. Genetic Algorithms, pages 97–125. Wiley Series on Parallel and Distributed Computing. Springer, New York, 2005. 31, 32, 33

[77] J. Kahn, M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal of Algebraic and Discrete Methods*, 4(2):194–206, 1983. 9, 67

[78] A. K. Kamrani and S. M. Salhieh. *Product Design for Modularity*, chapter 3. Design for Modularity, pages 85–122. Springer, 2000. 34

[79] D. Kima, C.-H. Chob, Y. Choa, J. Ryua, J. Bhakc, and D.-S. Kim. Pocket extraction on proteins via the voronoi diagram of spheres. *Journal of Molecular Graphics and Modelling*, 26(7):1104–1112, 2008. 3

[80] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983. 24

[81] M. van Kreveld and R.I. Silveira. Embedding rivers in polyhedral terrains. In *Proc. 25th ACM Symposium on Computational Geometry (SoCG)*, pages 771–778, 2009. 3

[82] A. Kumar. A novel genetic algorithm approach to solve map colour problem. In *ICETET '08: Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 288–291, Washington, DC, USA, 2008. IEEE Computer Society. 29

[83] A. Laurentini. Guarding the walls of an art gallery. *The Visual Computer*, 15(6):265–278, 1999. 9

[84] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, March 1986. 12, 67

[85] D.T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983. 42, 68, 115, 216

[86] J. van Leeuwen and A.A. Schoone. Untangling a travelling salesman tour in the plane. In J. R. Mhlbacher, editor, *Proc. 7th Internat. Workshop Graph-Theoret. Concepts Comput. Sci.*, pages 87–98, 1982. 19

[87] M. Lozano and C. García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, In Press, Corrected Proof, 2009. 23, 36

[88] G. MacDonald. Isomorphism and layout of spiral polygons. Master's thesis, Simon Fraser University, Burnaby, British Columbia, Canada, 1993. 197

[89] Samir W. Mahfoud and David E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. *Parallel Comput.*, 21(1):1–28, 1995. 37

[90] J. Maroco. *Análise Estatística - Com utilização do SPSS (in Portuguese)*. Edições Sílabo, third edition, 2007. 20

[91] A.M. Martins and A.L. Bajuelos. Some properties of fat and thin grid n-ogons. In *Proc. of International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2005)*, pages 361–365. Wiley-VCH Verlag, 2005. 16, 159

[92] A.M. Martins and A.L. Bajuelos. Characterizing and covering some subclasses of orthogonal polygons. In *Computational Science ICCS 2006: 6th International Conference*, pages 255–262. Lecture Notes in Computer Science (LNCS) 3992, Springer-Verlag, 2006. 16, 159

[93] A.M. Martins and A.L. Bajuelos. Guarding two subclasses of orthogonal polygons. In *Proc. International Conference of Computational Methods in Sciences and Engineering (ICMSE 2006)*, pages 372–375. Lecture Series on Computer and Computational Sciences, VSP/Brill, 2006. 16, 159

[94] A.M. Martins and A.L. Bajuelos. Vertex guards in a subclass of orthogonal polygons. *International Journal of Computer Science and Network Security (IJCSNS)*, 6(9):102–108, 2006. 16, 159

[95] N. Megiddo. Linear-time algorithms for linear programming in r3 and related problems. In *SFCS '82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 329–338, Washington, DC, USA, 1982. IEEE Computer Society. 2

[96] K. Mehlhorn, S. Näher, and C. Uhrig. The leda platform of combinatorial and geometric computing. In *ICALP '97: Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 7–16, London, UK, 1997. Springer-Verlag. 3

[97] K. Mehlhorn, S. Näher, and C. Uhrig. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. 3

[98] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. 24

[99] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. J. Wiley, New York, 1988. 164

[100] B.J. Nilsson and D. Wood. Optimum watchmen routes in spiral polygons. In *Proceedings of the Second Canadian Conference in Computational Geometry*, pages 127–136, 1990. 5, 174, 195

[101] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987. 4, 5, 7, 9, 10, 137, 138, 159

[102] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 1998. 1, 67

[103] J. O'Rourke and G. T. Toussaint. Pattern recognition. In J.E Goodman and J. O'Rourke, editors, *Handbook of discrete and computational geometry*, pages 1135–1162. CRC Press, Inc., Boca Raton, Florida, USA, 2004. 3

[104] Joseph O'Rourke. Galleries need fewer mobile guards: a variation on Chvátal's theorem. *Geom. Dedicata*, 14:273–283, 1983. 10

[105] Ibrahim H. Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63, 1996. 38

[106] J. Pach, editor. *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*. Springer Verlag, 1993. 1

[107] E. Packer. Computing multiple watchman routes. In *WEA*, pages 114–128, 2008. 4

[108] T. Pavlidis and H. Y. Feng. Shape discrimination, syntatic pattern recognition., 1997. 174

[109] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, August 1985. 1, 2

[110] Computational Science Education Project. *Mathematical Optimization*. 1995. 26, 27, 29

[111] Van den Berg J.P R. Wein, R. and D. Halperin. Planning near-optimal corridors amidst obstacles. In *Proc. 7th International Workshop on the Algorithmic Foundations of Robotics - WAFR 2006*, 2006. 3

[112] C.R. Reeves. Genetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 55–82. Kluwer Academic Publishers, 2003. v, 30, 31, 32, 33, 34

[113] J.-R. Sack and J. Urrutia. *Handbook of computational geometry*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2000. 1

[114] S. Schirra. Designing a computational geometry algorithms library. Research Report MPI-I-97-1-014, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, July 1997. 2

[115] D. Schuchardt and H. Hecker. Two np-hard art-gallery problems for ortho-polygons. *Math. Logiv Quart*, 41:261–267, 1995. 12, 65, 67, 77

[116] M. Sharir. Algorithmic motion planning in robotics. *Computer*, 22(3):9–20, 1989. 3

[117] T. Shermer. Hiding people in polygons. *Computing*, 42(2-3):109–131, 1989. 9, 11, 13, 39, 40, 206

[118] T.C. Shermer. Several short results in the combinatorics of visibility. Technical Report 91–2, May 1991. 10

[119] T.C. Shermer. Recent results in art galleries [geometry]. *Proceedings of the IEEE*, 80(9):1384–1399, Sep 1992. 4, 5, 9, 42

[120] J. Snoeyink and Z. Chong. Generating random monotone polygons. Technical report. 149

[121] W. Steiger and I. Streinu. Illumination by floodlights. *Computational Geometry*, 10(1):57 – 70, 1998. 10

[122] Harold Szu and Ralph Hartley. Fast simulated annealing. *Physics Letters A*, 122(3-4):157 – 162, 1987. 28

[123] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, 2002. 35, 36

[124] A.P. Tomás and A.L. Bajuelos. Quadratic-time linear-space algorithms for generating orthogonal polygons with a given number of vertices. In *ICCSA (3)*, pages 117–126, 2004. viii, ix, 15, 159, 160, 161, 162

[125] A.P. Tomás, A.L. Bajuelos, and F. Marques. Approximation algorithms to minimum vertex cover problems on polygons and terrains. In *International Conference on Computational Science*, pages 869–878. Springer-Verlag, 2003. 4, 151

[126] A.P. Tomás, A.L. Bajuelos, and F. Marques. On visibility problems in the plane-solving minimum vertex guard problems by successive approximations. In *on-line Proceedings of Artificial Intelligence and Mathematics*, 2006. 4, 15, 159, 187

[127] G. Toussaint. What is computational geometry? *Proceedings of the IEEE*, 80(9):1347–1363, Sep 1992. 1

[128] G. T. Toussaint. *Computational Geometry.* NorthHolland, Amsterdam, Netherlands, 1985. 1

[129] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of computational geometry*, pages 973–1027. Elsevier, 2000. 4, 5, 7, 8, 9, 10, 12, 13, 68, 112, 114

[130] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985. 24

[131] Y. Wang, C. Hu, and Y. Tseng. Efficient placement and dispatch of sensors in a wireless sensor network. *IEEE Transactions on Mobile Computing*, 7(2):262–274, 2008. 135

[132] D. W. Waynem. *Applied Non-Parametric Statistics.* PWS-KENT Publishing Company, Boston, second edition, 1990. The Duxbury Advanced Series in Statistics and Decision Sciences. 20

[133] T. Weise. *Global Optimization Algorithms - Theory and Application.* Thomas Weise, 2007-05-01 edition, 2007. The book is online available at http://www.it-weise.de/documents/index.html#W2007GOGP. 31, 32

[134] C. Worman and M. J. Keil. Polygon decomposition and the orthogonal art gallery problem. *Int. J. Comput. Geometry Appl.*, 17(2):105–138, 2007. 5

[135] X Yao. A new simulated annealing algorithm. *International Journal of Computer Mathematics*, 56:161–168, 1995. 28

[136] C. Zonnenberg. Conformal geometric algebra package. Master's thesis, Department of Information and Computing Sciences, Faculty of Sciences, Utrecht University, Netherlands, 2007. 3