

A genetic algorithm for designing optimal patch configurations in GIS

by Christopher J Brookes, MSc.

A thesis submitted for the degree of Doctor of Philosophy

University College
University of London
London
1998



Abstract

Geographical Information Systems (GIS) are used for several types of spatial planning but so far they have not been used for optimal patch design. Optimal patch design is a generic spatial problem in which the objective is to design spatially explicit landuse maps when both the composition and configuration of patches are important criteria. There are many applications in conservation, forestry management, watershed management and the management of large military estates.

This thesis describes a new autonomous computer program, the genetic algorithm for optimal patch design (GAPD). GAPD combines four components: a genetic search algorithm, a parameterised region growing (PRG) program, raster GIS measurement functions and multi-criteria decision-making methods. The key component is the PRG which translates between the aspatial domain of the search algorithm and the spatial domain of the GIS. GAPD generates landuse maps that optimise the configuration and composition of patches to meet multiple objectives for a given set of input maps and criteria.

The theories of landscape ecology are used to establish a framework for formulating optimal patch design problems. The thesis describes the conceptual design of GAPD and its implementation and test, first as a prototype for solving single patch problems and then as a fully functional system for solving multi-objective multi-patch problems. The feasibility of GAPD was established by investigations of issues concerning the representation and measurement of configuration in raster data structures and by testing the efficiency and effectiveness of GAPD with simple problems. GAPD was further evaluated in five hypothetical problems designed to cover a range of different scenarios. The results are promising and show that GAPD has potential as a decision support tool.

The final section recommends a number of topics for further research covering technical developments of GAPD, the application of GAPD to real problems and investigations of general issues of optimal patch design.

Acknowledgements

I am very grateful to Peter Fisher, who set me on the research trail at Leicester University, and to my supervisor at UCL, Paul Densham, who prevented me from straying from the path. I also thank the Graduate School and the Geography Department of UCL for their financial support and the Geography Department of Portsmouth University for supporting my attendance at conferences.

This work has benefited from the comments of numerous people including several anonymous referees of journal submissions and people I have met at conferences. I particularly want to thank Peter Atkinson of Southampton University and Philip Lewis of UCL.

List of Contents

| | |
|--|----|
| 1. INTRODUCTION, BACKGROUND AND RESEARCH OUTLINE | 18 |
| 1.1 Introduction | 18 |
| 1.1.1 The Optimal Patch Design Problem | 18 |
| 1.1.2 Applications | 18 |
| 1.2 Background | 20 |
| 1.2.1 Context | 20 |
| 1.2.2 Previous research | 22 |
| 1.3 Research outline | 23 |
| 1.3.1 The research question | 23 |
| 1.3.2 Aims and Objectives | 24 |
| 1.4 Implementation and test | 24 |
| 1.4.1 Conceptual model | 24 |
| 1.4.2 Implementation | 25 |
| 1.4.3 Testing | 26 |
| 1.5 Structure of the thesis | 27 |
| 1.6 Summary | 28 |
| 2. DECISION-MAKING | 29 |
| 2.1 Introduction | 29 |
| 2.2 Overview of decision science | 29 |
| 2.3 Multi-Criteria Decision-Making | 32 |
| 2.3.1 Overview | 32 |
| 2.3.2 Compensatory methods | 33 |
| 2.3.3 Non-compensatory methods | 36 |
| 2.4 Search methods | 37 |
| 2.5 Decision-making and decision support in GIS: A review | 40 |
| 2.6 Summary | 43 |
| 3. OPTIMAL PATCH DESIGN AND CONSERVATION | 44 |
| 3.1 Introduction | 44 |
| 3.2 Ecological background | 44 |
| 3.2.1 The fragmentation problem | 44 |
| 3.2.2 Conservation strategies and goal setting | 45 |
| 3.3 Applications of decision-making techniques to conservation | 47 |
| 3.4 Reserve Design - designing optimal patches | 48 |
| 3.4.1 Composition and configuration | 48 |
| 3.4.2 Spatial factors | 48 |
| 3.4.2.1 Landscape ecology | 49 |
| 3.4.2.2 Conservation biology | 51 |
| 3.4.3 Summary of spatial influences | 55 |
| 3.5 Summary | 57 |
| 4. OPTIMAL PATCH DESIGN ON RASTER MAPS | 59 |
| 4.1 Introduction | 59 |
| 4.2 The patch design problem | 59 |

| | | |
|---------|---|-----|
| 4.3 | Cell suitability assessment | 62 |
| 4.4 | Heuristics | 63 |
| 4.4.1 | The Iterative Relaxation heuristic | 64 |
| 4.4.2 | Region-growing heuristics | 65 |
| 4.5 | Search | 66 |
| 4.5.1 | Measuring spatial properties | 67 |
| 4.5.2 | Evaluating patch utility | 69 |
| 4.6 | Overview of the GAPD conceptual design | 69 |
| 5. | GENETIC ALGORITHMS | 72 |
| 5.1 | Introduction | 72 |
| 5.2 | History and background | 72 |
| 5.3 | Genetic Algorithm Fundamentals | 74 |
| 5.4 | The canonical genetic algorithm | 77 |
| 5.5 | Genetic Algorithms for decision-making | 77 |
| 5.5.1 | How genetic algorithms are applied in decision-making | 77 |
| 5.5.2 | When genetic algorithms are applied in decision-making | 79 |
| 5.6 | Review of some genetic algorithm applications | 80 |
| 5.7 | Summary of Operational issues | 82 |
| 5.7.1 | Coding | 82 |
| 5.7.2 | Fitness function | 82 |
| 5.7.3 | Genetic Operators | 83 |
| 5.7.4 | Initial population generation | 83 |
| 5.7.5 | Number of generations | 84 |
| 5.7.6 | Selection methods | 84 |
| 5.7.7 | Hybrid algorithms | 85 |
| 5.8 | Application of genetic algorithms to optimal patch design | 86 |
| 6. | DESIGN AND IMPLEMENTATION | 88 |
| 6.1 | Introduction | 88 |
| 6.2 | GAPD structure | 89 |
| 6.3 | The prototype GAPD | 92 |
| 6.3.1 | Introduction | 92 |
| 6.3.2 | The PRG component | 93 |
| 6.3.2.1 | Simple region-growing - SRG | 94 |
| 6.3.2.2 | Parameterised shape-growing - PSG | 95 |
| 6.3.2.3 | PRG - the fusion of SRG and PSG | 103 |
| 6.3.2.4 | Summary | 106 |
| 6.3.3 | The prototype genetic search driver | 106 |
| 6.3.3.1 | Physical components | 106 |
| 6.3.3.2 | Operational context | 111 |
| 6.4 | The transition from the prototype to GAPD | 112 |
| 6.4.1 | Introduction | 112 |
| 6.4.2 | Possible alternative drivers | 112 |
| 6.4.2.1 | Variable length string | 112 |
| 6.4.2.2 | Hierarchic structure | 113 |
| 6.4.2.3 | Other possibilities | 114 |

| | |
|---|-----|
| 6.4.2.4 Why this line was abandoned | 114 |
| 6.5 The full function GAPD | 115 |
| 6.5.1 Modified PRG | 115 |
| 6.5.2 The GAPD driver | 118 |
| 6.5.3 Pragmatic variations in the implementation of GAPD | 119 |
| 6.6 The measurement and evaluation functions | 119 |
| 6.6.1 Basic raster functions | 120 |
| 6.6.2 Attribute measurements | 121 |
| 6.6.3 Evaluation functions | 122 |
| 6.7 Further developments | 124 |
| 6.8 Summary | 125 |
| | |
| 7. FEASIBILITY TESTS | 126 |
| 7.1 Introduction | 126 |
| 7.2 The issue of shape in rasters | 126 |
| 7.2.1 Introduction | 126 |
| 7.2.2 The raster data structure | 127 |
| 7.2.3 Measuring shape on a raster | 129 |
| 7.2.4 Experiment 1 : Effect of measurement technique | 130 |
| 7.2.5 Experiment 2 : Varying the shape control parameters | 133 |
| 7.2.6 Experiment 3 : Effect of size | 134 |
| 7.2.7 Experiment 4 : Effect of shape control parameters | 136 |
| 7.2.8 Conclusions | 139 |
| 7.3 Effectiveness | 141 |
| 7.4 Efficiency | 143 |
| 7.4.1 Introduction | 143 |
| 7.4.2 Test maps | 144 |
| 7.4.3 Method | 151 |
| 7.4.4 Results | 152 |
| 7.4.5 Discussion | 157 |
| | |
| 8. EVALUATION TESTS | 158 |
| 8.1 Objectives and methodology | 158 |
| 8.1.1 Objectives | 158 |
| 8.1.2 Test design | 158 |
| 8.1.3 Test methods | 161 |
| 8.1.4 Test verification | 162 |
| 8.2 Evaluation 1 : Designing a single compact patch | 163 |
| 8.2.1 Purpose | 163 |
| 8.2.2 Problem description | 164 |
| 8.2.2.1 Objective function | 164 |
| 8.2.2.2 Data | 164 |
| 8.2.3 Method | 165 |
| 8.2.4 Results | 168 |
| 8.2.5 Discussion | 175 |
| 8.3 Evaluation 2 : Single Objective Multi-Patch | 177 |
| 8.3.1 Purpose | 177 |

| | |
|---|-----|
| 8.3.2 Problem description | 177 |
| 8.3.2.1 Objective function | 177 |
| 8.3.2.2 Data | 178 |
| 8.3.3 Method | 180 |
| 8.3.4 Results | 182 |
| 8.3.5 Discussion | 187 |
| 8.4 Evaluation 3 : Multi-objective single-patch problem | 188 |
| 8.4.1 Purpose | 188 |
| 8.4.2 Problem description | 189 |
| 8.4.2.1 Objective function | 189 |
| 8.4.2.2 Data | 192 |
| 8.4.3 Method | 192 |
| 8.4.4 Results | 194 |
| 8.4.5 Discussion | 200 |
| 8.5 Evaluation 4 : Multi-patch problem | 201 |
| 8.5.1 Purpose | 201 |
| 8.5.2 Problem description | 202 |
| 8.5.2.1 Objective function | 202 |
| 8.5.2.2 Data | 204 |
| 8.5.3 Method | 205 |
| 8.5.4 Results | 207 |
| 8.5.5 Discussion | 214 |
| 8.6 Evaluation 5 : Multi-objective multi-patch problem | 216 |
| 8.6.1 Purpose | 216 |
| 8.6.2 Problem description | 217 |
| 8.6.2.1 Objective function | 217 |
| 8.6.2.2 Data | 218 |
| 8.6.3 Method | 220 |
| 8.6.4 Results | 223 |
| 8.6.5 Discussion | 235 |
| 8.7 Summary | 237 |
| 8.7.1 Review | 237 |
| 8.7.2 Generality of GAPD | 237 |
| 8.7.3 Performance | 238 |
| 8.7.4 Implementation | 240 |
| 8.7.5 Appropriate use of GAPD | 240 |
| 8.7.6 Conclusion | 241 |
| | |
| 9. REVIEW AND RECOMMENDATIONS FOR FURTHER RESEARCH | 242 |
| 9.1 Summary | 242 |
| 9.2 Recommendations for further research | 244 |
| 9.3 Integration with GIS | 246 |
| 9.4 Other aspects not previously covered | 247 |
| 9.5 Contribution | 250 |
| | |
| APPENDICES | 252 |
| LIST OF REFERENCES | 271 |

List of tables

| | |
|---|-----|
| 3.1 Landscape elements: their functions and spatial characteristics | 56 |
| 4.1 Levels of spatial arrangement | 68 |
| 6.1 Comparison of the prototype GAPD and the full function GAPD | 93 |
| 6.2 Shape definition codes for figure 6.7 | 102 |
| 6.3 Genetic code structure and sample parameter domains (the values for size, row and column depend on the problem and the dimensions of the raster) | 108 |
| 7.1 GAPD solutions to optimise core area, edge area and both core and edge area, on a uniform raster | 142 |
| 7.2 Estimated patch sizes for maps in series 1 and 2 | 145 |
| 7.3 Comparison of GAPD and exhaustive search in the GAPD efficiency tests . . | 153 |
| 7.4 Comparison of GAPD and exhaustive search for map 1 in series 2 | 156 |
| 8.1 Summary of the five evaluation tests | 160 |
| 8.1a Genetic code for evaluation test 1 | 166 |
| 8.2 Genetic algorithm control parameters in evaluation test 1 | 166 |
| 8.3 IR iterations to find a patch of the right size in evaluation test 1 | 168 |
| 8.4 Solutions for the three methods in evaluation test 1 | 169 |
| 8.5 GAPD progress in evaluation test 1 showing initial population values, final population values and the generation when maximum utility became better than the PRG solution | 175 |
| 8.5a Genetic code for evaluation test 2 | 180 |
| 8.6 IR iterations for the 100 and 200 allocations in evaluation test 2 | 183 |
| 8.7 The six solutions in evaluation test 2 | 183 |
| 8.7a Genetic code for evaluation test 3 | 190 |
| 8.8 PRG parameters (PDCs) for the GAPD solutions in evaluation test 3 | 195 |

| | |
|---|-----|
| 8.9 Patch attribute and utility values in evaluation test 3 | 196 |
| 8.10 Change in patch characteristics with changing objective weights in evaluation test 3 | 197 |
| 8.10a Genetic code for evaluation test 4 | 206 |
| 8.11 Preliminary trial of GAPD with modified PRG for evaluation test 4 | 209 |
| 8.12 Comparison of different versions of GAPD in evaluation test 4 | 210 |
| 8.13 Comparison of results for different allocation levels in evaluation test 4 ... | 211 |
| 8.14 Comparison of different versions of GAPD for the 600 allocation level in evaluation test 4 | 212 |
| 8.15 Conflict between aspatial objectives at different allocation levels in evaluation test 5 | 220 |
| 8.16 Genetic code for the multi-objective GAPD in evaluation test 5 | 221 |
| 8.17 Patch attributes for all solutions in evaluation test 5 | 224 |
| 8.18 Agriculture patches in evaluation test 5 | 225 |
| 8.19 Carpet industry patches in evaluation test 5 | 226 |
| 8.20 IR iterations to find the agriculture patch (IR-AG) in evaluation test 5 | 227 |
| 8.21 IR iterations to find the carpet industry patch (IR-CARP) in evaluation test 5 | 227 |
| 8.22 PRG parameters (PDCs) for the single objective GAPD solutions in evaluation test 5 | 230 |
| 8.23 Utility of the multi-objective GAPD solutions in evaluation test 5 | 233 |
| 8.24 PRG parameters (PDCs) for the multi-objective GAPD solutions in evaluation test 5 | 234 |
| 8.25 Solution time for complete enumeration, in everyday time units, on a computer with terraflop performance | 239 |

List of Figures

| | |
|---|-----|
| 2.1 A non-monotonic stepwise linear utility function | 34 |
| 4.1 GAPD conceptual design | 70 |
| 6.1 Logical structure of GAPD | 90 |
| 6.2 Physical structure of GAPD | 91 |
| 6.3 GAPD flow diagram | 91 |
| 6.4 SRG parameter structure (PDC) | 95 |
| 6.5 PSG parameter structure (PDC) | 97 |
| 6.6 Sinusoidal shape with 4 axes and a ratio of major to minor axes of 2:1 | 98 |
| 6.6a Diagram showing the meanings of the measurements d and θ in equations 6.1 and 6.2 and how they are calculated | 100 |
| 6.7 Shapes grown by PSG with dual shape definition codes (the SDCs are listed in table 6.2) | 103 |
| 6.8 PRG parameter structure (PDC) | 104 |
| 6.9 Genetic code for the GAPD prototype | 107 |
| 6.10 Genetic operators (a): crossover, mutation and creep | 109 |
| 6.11 Genetic operators (b): average and sum | 110 |
| 6.12 Genetic code for the full function GAPD | 116 |
| 6.13 Core area and edge area for various shapes | 122 |
| 7.1 Variation of core area score with size | 131 |
| 7.2 Variation of PI score with size | 132 |
| 7.3 Variation of core area with number of axes for 3 numerator values | 134 |
| 7.4 Variation of core area with size for a constant proportional edge effect | 135 |
| 7.5 Variation of core area with size for number of axes = 8 | 137 |
| 7.6 Variation of core area with size for number of axes = 10 | 138 |

| | |
|---|-----|
| 7.7 Variation of core area with size for number of axes = 12 | 138 |
| 7.8 Shapes found by GAPD optimising core (top row), edge (bottom row) and both core and edge (middle row) | 143 |
| 7.9 Suitability map 2.1 | 146 |
| 7.10 Suitability map 2.2 | 146 |
| 7.11 Suitability map 2.3 | 147 |
| 7.12 Suitability map 2.4 | 147 |
| 7.13 Suitability map 2.5 | 148 |
| 7.14 Suitability map 2.6 | 148 |
| 7.15 Suitability map 2.7 | 149 |
| 7.16 Suitability map 2.8 | 149 |
| 7.17 Suitability map 2.9 | 150 |
| 7.18 Suitability map 2.10 | 150 |
| 7.19 GAPD and exhaustive search utility scores in map series 1 | 154 |
| 7.20 GAPD and exhaustive search utility scores in map series 2 | 154 |
| 8.1 Suitability map for evaluation test 1 | 165 |
| 8.2 Patch found by IR for the 150 allocation in evaluation test 1 | 170 |
| 8.3 Patch found by IR for the 200 allocation in evaluation test 1 | 170 |
| 8.4 Patch found by IR for the 300 allocation in evaluation test 1 | 171 |
| 8.5 Four patches found using PRG for the 150 allocation in evaluation test 1 ... | 171 |
| 8.6 Four patches found using PRG for the 200 allocation in evaluation test 1 ... | 172 |
| 8.7 Four patches found using PRG for the 300 allocation in evaluation test 1 ... | 172 |
| 8.8 Patch found by GAPD for the 150 allocation level in evaluation test 1 | 173 |
| 8.9 Patch found by GAPD for the 200 allocation level in evaluation test 1 | 173 |

| | |
|---|-----|
| 8.10 Patch found by GAPD for the 300 allocation level in evaluation test 1 | 174 |
| 8.11 Suitability map for evaluation tests 2 and 3 | 179 |
| 8.12 Distance layer for evaluation tests 2 and 3 | 179 |
| 8.13 IR-a1: patches found by IR for the 100 allocation in evaluation test 2 | 184 |
| 8.14 IR-a2: patches found by IR for the 200 allocation in evaluation test 2 | 184 |
| 8.15 G-a1: patches found by GAPD for the 100 allocation in evaluation test 2, with no distance constraint | 185 |
| 8.16 G-b1: patches found by GAPD for the 100 allocation in evaluation test 2, with a distance constraint | 185 |
| 8.17 G-a2: patches found by GAPD for the 200 allocation in evaluation test 2, with no distance constraint | 186 |
| 8.18 G-b2: patches found by GAPD for the 200 allocation in evaluation test 2, with a distance constraint | 186 |
| 8.19 Optimal patch for the core objective (weight 10-0) in evaluation test 3 | 198 |
| 8.20 Optimal patch for two equally weighted objectives (weight 5-5) in evaluation test 3 | 198 |
| 8.21 Optimal patch for the isolation objective (weight 0-10) in evaluation test 3 . | 199 |
| 8.22 Data layer showing landuse categories in evaluation test 4 | 213 |
| 8.23 Six patches found by iterative application of single patch GAPD in evaluation test 4 | 213 |
| 8.24 Six patches found by fixed multi-patch GAPD in evaluation test 4 | 214 |
| 8.25 Eight patches found by variable multi-patch GAPD in evaluation test 4 . . . | 214 |
| 8.26 Suitability map for agriculture in evaluation test 5 | 219 |
| 8.27 Suitability map for the carpet industry in evaluation test 5 | 219 |
| 8.28 Location of the agriculture patch found by IR in evaluation test 5 | 228 |
| 8.29 Location of the carpet industry patch found by IR in evaluation test 5 | 229 |

| | |
|--|------------|
| 8.30 Location of the agriculture patch found by single objective GAPD in evaluation test 5 | 230 |
| 8.31 Location of the carpet industry patch found by single objective GAPD in evaluation test 5 | 231 |
| 8.32 Location of the agriculture and carpet industry patches found by the hierarchical GAPD method in evaluation test 5 | 232 |
| 8.33 Location of the agriculture and carpet industry patches found by multi-objective GAPD in evaluation test 5 | 233 |

List of Appendices

| | |
|---|-----|
| APPENDIX I. Technical programming notes | 252 |
| APPENDIX II. Pseudo code | 255 |
| APPENDIX III. Operational modes of GAPD | 263 |
| APPENDIX IV. Estimating the search space size | 268 |

Additional material

Two offprints of journal papers by the author are included with the thesis. The first (Brookes 1997a) describes some of the early development work on PRG and the second (Brookes 1997b) describes some experiments with the GAPD prototype. Where it is necessary to maintain the flow material from these papers also appears in the main body of the text. Other parts are referred to in the text and the papers are included with the thesis for the convenience of the reader.

Definitions

1. Acronyms

GAPD. Genetic algorithm for optimal patch design

GIS. Geographical Information System

MCDM. Multi-criteria decision-making

MPDC. Multiple patch definition code

PDC. Patch definition code

PRG. Parameterised region-growing

PSG. Parameterised shape-growing

SRG. Simple region-growing

SDC. Shape definition code

2. IDRISI functions

AREA. Calculates the total area of each cell type. Note that if two patches have the same type the AREA command will return the total area of both patches, not the area of each patch.

EXTRACT. Extracts values from one image using features defined on an overlay image.

GROUP. Groups contiguous cells with the same value into patches and assigns a patch identifier to each cell in the group. GROUP and AREA can be used together to find the area of each patch.

HISTO. Displays statistical information about an image, such as the number of cells of each type.

OVERLAY. Includes several options for overlay processing including mathematical operations. If an image is multiplied by a binary image zero valued cells in the binary image produce zero valued cells in the output image, therefore overlay can be used as a masking operation.

RANK. Assigns a unique value (rank) to each cell by sorting cells in ascending or descending order.

RECLASS. Classifies cells by their value and assigns a new value to each cell.

STRETCH. Transforms an image by reassigning the maximum and minimum values and then reclassifying each intermediate cell using an interpolation procedure. The interpolation can be linear or statistical. Stretch is typically used to improve the visual appearances of images.

3. IDRISI file structures

Document file. IDRISI images are held in two files. The document file is a text file containing a description of the image. It includes information such as the dimensions of the image and the range of cell values.

Image file. The image file holds the actual cell values. Image files can only be processed correctly if they are accompanied by a document file.

4. Other terms

Boundary effects. Effects caused by the finite limits of rasters. A patch cannot grow outside the raster so patch shape is constrained near the raster boundary.

Composition. The amount of different patches or elements within patches.

Configuration. The spatial distribution and shape of patches.

Core area. The inner area of a patch not affected by edge effects. Where edge effects are strong the core area is much smaller than total patch area.

Edge effects. Ecological, and other, effects occurring at the edge of patches. Edge effects create a transition zone between the interior and exterior of patches.

Effectiveness (of a search procedure). A measure of the quality of the solutions. A search is effective if it finds good solutions.

Efficiency (of a search procedure). A measure of the quality of the process. A search is efficient if it finds good solutions with limited resources. More rigorously, efficiency is the ratio of the alternatives examined to the total number of alternatives in the search space.

Matrix. The most extensive and connected landscape element.

Patch. A distinct relatively homogeneous area within a landscape.

1. INTRODUCTION, BACKGROUND AND RESEARCH OUTLINE

1.1 Introduction

Optimal patch design is a generic spatial planning problem because pattern and process are intricately linked in many spatial systems. The relevance of configuration as a criterion has been recognised within different planning contexts but the treatment of spatial factors has been *ad hoc*. This research addresses the issues in optimal patch design, formulates a conceptual model for a generic decision support tool, identifies suitable components and implements the model as a working system.

A major part of this thesis is the development and test of a computer system for designing optimal patch configurations using raster GIS. Some work has been carried out recently on similar continuous space problems with vector systems (Krzanowski 1997). The Genetic Algorithm for optimal Patch Design, GAPD, generates raster maps showing the explicit location and configuration of patches. The composition and configuration of patches are optimised to meet multiple objectives subject to multiple criteria. GAPD links a genetic algorithm with raster GIS and multi-criteria decision-making routines. A novel technique, parameterised region-growing, PRG (Brookes, 1997a), is used as a bridge between the aspatial domain of the genetic algorithm and the spatial domain of the GIS.

1.1.1 *The optimal patch design problem*

In this thesis the term **optimal patch design** refers to spatial planning problems in which the objective is to design spatially explicit plans that optimise both the composition of patches and their spatial configuration. The research originates from a recognition that decision making methods in raster GIS are inadequate for many site location and landuse planning problems. The fundamental problem is that traditional multi-criteria evaluation methods that have been applied in GIS evaluate cells individually without reference to other cells in the solution. The key issues are contiguity constraints on cells, to define

patches, and spatial relationships among patches. Patch is a term used in landscape ecology to denote physical entities. Patches can also be human constructs such as administration districts, planning zones or catchment areas. In the development and test of the methodology this research concentrates on problems in which spatial extent, configuration and distribution are significant criteria, because these are the areas that have not been addressed previously. However, the research has wider significance because such problems can be seen as existing somewhere in the middle range of a spectrum of problems. At one extreme, the objective is to partition a whole space and at the other to locate multiple point sites (ie. isolated cell). In the latter case, although the sites themselves are not patches, they implicitly define patches if the pattern of sites and, therefore, the configuration of the space between sites, is significant. Thus, a generic methodology for addressing optimal patch design is relevant to a range of site location and landuse planning problems.

1.1.2 Applications

Patch configuration is a relevant criterion in nature reserve and habitat design and potentially in other applications such as watershed management, forestry, the design of electoral district boundaries and local authority planning. Patch size is important for population viability (Morrison *et al.*, 1992; Gilpin & Soule, 1986), disturbance regimes (Baker, 1992) and ecological diversity (Probst & Weinrich 1993). Shape affects the role of patches as corridors (Fahrig & Merriam, 1985; Selman & Doar, 1992; Soule & Gilpin (1991) or habitats (Laurance 1991) and the amount of interaction with adjacent areas (Buechner 1987). Inter-patch connectivity affects the diversity of metapopulations and the behaviour of territorial species (Hanski and Thomas 1994; Murphy et al. 1990; Carroll and Lamberson 1993). Finally, patch composition determines suitability of territories for species with a variety of needs and diversity within a patch increases the chances of a population surviving short or long term fluctuations in climate. Analogies can be made in other situations. In urban planning, a compact industrial zone will have minimal disturbance effect on surrounding areas whereas a linear park may maximise accessibility from neighbouring residential areas. The shape of electoral districts is

important to avoid suspicion of gerrymandering. When locating facilities at sites the catchment may need to include diverse social elements. Configuration can affect physical systems in various ways. The distribution of landcover patches in a watershed will affect runoff and consequently erosion, sedimentation and flooding. Forests are managed for efficient timber production, for which large uniform stands are advantageous, and conservation and recreation which require a pattern of smaller more diverse stands. The spread of forest fires are also affected by planting patterns.

1.2 Background

1.2.1 Context

The idea that pattern and process are interrelated is common to many spatial disciplines and it is fundamental in landscape ecology. Landscape ecology is the study of landscape processes and landscape structure (Forman & Godron, 1986). The contribution of landscape ecology in this research is to provide a set of concepts for formulating optimal patch design problems. Patch is a term used in landscape ecology to denote spatial units in a landscape. Configuration refers to the shape and spatial distribution of patches. Composition is the amount of different elements within a patch or landscape.

Landscape ecology is holistic and the links between form and process are implicit and hard to quantify. In other disciplines the link is clearer. Many natural processes are influenced by both the composition and configuration of patches. Consequently, alternative configurations have different value with respect to particular objectives and criteria. For example patch configuration affects:

- ecological functions and the value of habitats;
- surface flow routing in watersheds and catchments with implications for erosion and flooding;
- the spread of fires; and
- the economic productivity and recreational value of managed forests.

In part, the motivation for this research comes from an interest in conservation and that is why conservation biology was chosen to exemplify the effects of configuration on value. Ecologists have studied the effects of spatial factors on habitat quality and have been able to draw up guidelines for the design of nature reserves (Smith & Theberge, 1986; Morrison *et al.*, 1992). These guidelines can be translated into evaluation functions within a decision-making context. Analogous ideas can be used in other contexts, as outlined in above (1.1.2).

Optimal patch design is a type of landuse planning problem. In landuse planning many of the relevant attributes such as vegetation, slope, soil type and so on, vary continuously over space. This spatial variability is better handled using raster data structures than vector. Raster GIS functions are perfectly adequate for the suitability assessment that this type of problem demands (Eastman *et al.*, 1993; Pereira & Duckstein, 1993; Jankowski, 1995). If configuration is irrelevant then suitability analysis is sufficient. However, when configuration is relevant the problem is more complex and other methods are needed.

GIS do not have the functionality for what Tomlin calls prescriptive modelling with holistic criteria (Tomlin, 1990). Holistic criteria, as opposed to atomic criteria, can only be evaluated when the solution is known. In this thesis the terms dynamic and static criteria are used instead of holistic and atomic respectively because they express this fundamental difference better. Tomlin's solution is to use *ad hoc* heuristics. He proposes some example problems and heuristics that can be implemented in GIS. An alternative approach is to use GIS as a support tool, coupled with other tools in an interactive decision-making environment (Fedorowick, 1993; Barrett & Peles, 1994; Chuvieco, 1993). GAPD is a single, autonomous system that handles the data, criteria and objectives pertinent to optimal patch design.

Configuration is a factor in other spatial planning problems including optimal routing, location-allocation and site location. Some GIS incorporate algorithms for these applications. The algorithms represent the spatial entities by points and lines and the

spatial criteria are the location of points and the connections between them. The difference between optimal patch design and these other types of problems is that the spatial entities, the patches, have a spatial extent and, therefore, an explicit shape as well as a location.

A number of computational techniques have been developed which exploit the power of digital computers. These techniques are now being used in GIS and image processing applications (Openshaw & Abrahart, 1996). Of most relevance to optimal patch design are genetic algorithms which are powerful generic search methods for finding solutions to complex problems. The main problem with their application to optimal patch design is to find suitable data structures that represent two dimensional entities and at the same time are amenable to manipulation by the genetic algorithm.

Decision-making systems require data. Developments in remote sensing technology, both in sensor hardware and image processing software, mean that the necessary data for landuse planning can be acquired in sufficient quantity, detail and with regular updates.

1.2.2 Previous research

A key component of GAPD is developed from an original piece of research which I undertook as part of a Masters degree at Leicester University (Brookes 1993). I developed a parameterised region-growing (PRG) program that operates on a raster suitability map and is driven by a string of control parameters. The program generates regions, or patches, whose form depends on both the underlying suitability map and the control parameters. This research addressed the question of incorporating patch characteristics, as equally important criteria with cell-based suitability analysis, in a multi-criteria decision-making process. The original PRG program has two major limitations. Its operation is problematic because the program has no search capability and it must therefore be supervised by an operator. Furthermore, the program is only able to grow simple symmetrical shapes.

1.3 Research outline

1.3.1 *The research question*

In optimal patch design the objective is to produce spatially explicit landuse plans that optimise both the configuration and composition of patches. Optimal patch design is a complex spatial problem. A number of key features require particular attention.

- The attributes that contribute to patch composition are distributed spatially. They vary continuously and, consequently, are best represented in raster data structures.
- Patch configuration is a spatial property whose measurement is greatly facilitated by the use of GIS which should therefore be integrated within the chosen methodology.
- Patches are evaluated against multiple criteria.
- Some criteria are static: a single evaluation applies to all possible solutions. Criteria relating to composition are mainly static.
- Some criteria are dynamic: they must be re-evaluated for each alternative solution. Criteria relating to configuration are dynamic.
- Spatial decisions frequently involve multiple conflicting objectives.
- There are an infinite number of possible patch configurations and subtle changes in patch location or configuration can have disproportionate effects on utility. Thus, the size and complexity of the search space demands an efficient search mechanism.
- Most aspects of spatial decision-making involve uncertainty. There is not only uncertainty in the attribute measurements but also in the criteria and objectives.

While uncertainty can be addressed by analytical methods it is better handled by appropriate use of tools and regard to the decision-making context. This thesis is not concerned with issues of uncertainty other than to acknowledge them and to recommend that any operational tool, such as GAPD, be used in an appropriate context. The other aspects of the problem listed above are incorporated into the design of GAPD.

1.3.2 Aims and Objectives

The principal objective of the research is to build and test an autonomous computer system for the design of optimal patch configurations. A secondary objective is to link GAPD with GIS and consider alternative ways of achieving this. The end product, GAPD, will be evaluated against the following considerations:

- The patches generated by GAPD must show that GAPD responds to different criteria and different objectives.
- The patch configurations should look right. For example, in simple problems it may be obvious that patches should be compact.
- GAPD must be useable and flexible. It must be adaptable to different problem types.
- GAPD must be efficient: it must make better use of computer resources than an exhaustive search.
- GAPD must be effective. The solutions should be better than random guesses.

Working toward these goals will involve developing an understanding of the nature of the optimal patch design problem and the limitations of the system.

1.4 Implementation and test

The research strategy is to:

1. Produce a conceptual system design.
2. Develop an appropriate spatial search mechanism.
3. Implement the system in a suitable programming language.
4. Evaluate the system using realistic data and problems.

1.4.1 Conceptual model

The genetic algorithm for optimal patch design, GAPD, comprises four logical functions: search, translation, measurement and evaluation. The search function is performed by

a genetic algorithm: genetic algorithms are capable of efficiently exploring complex decision spaces. Measurement and evaluation are performed by raster GIS functions, that measure patch attributes, and MCDM methods, that convert attribute values to utility scores. These three components employ conventional techniques. The translation function is performed by parameterised region-growing (PRG): PRG serves as a bridge to link existing techniques in a single autonomous system. No single component of GAPD is adequate in itself but GAPD exploits the advantages of each.

The potential problem with search is that generic search algorithms, such as genetic algorithms, are aspatial and operate on internal data structures, typically strings. Patch structure is spatial and evaluation functions operate on raster maps. A code is needed to translate between structures used by the search algorithm and the spatial representation of patches. PRG provides the link: the PRG algorithm grows patches on a raster map under the control of a string of numeric parameters. The parameter strings are easily incorporated into a genetic algorithm and the raster maps are in a suitable format for GIS operations. Thus, PRG translates from string structures in the search domain to spatial structures in the problem domain.

GAPD addresses the main problem in optimal patch design - the handling of dynamic criteria. It explicitly evaluates dynamic criteria for each alternative patch configuration. Methods that explicitly evaluate atomic criteria only, rely on *ad hoc* problem-specific heuristics. GAPD uses a universal search heuristic that is independent of the problem domain. Only the criteria and objectives, and hence the measurement and evaluation functions, are problem specific. Another feature of GAPD is that it can operate directly on multiple data layers and, therefore, goes beyond traditional suitability mapping.

1.4.2 *Implementation*

As a preliminary to implementing GAPD, the PRG program was evaluated using objective tests. The PRG program was shown to generate patches that were superior to regions generated using a different heuristic (Brookes 1997a). However, before PRG

can be used as a translation function within a search algorithm it must be able to generate all possible patches. The first step in the implementation of GAPD was to improve the original PRG algorithm to grow a wider range of shapes.

The translation component dictates the form of the genetic code used in the genetic algorithm. Basic genetic algorithm source code is in the public domain: a version published by Ribeiro Filho et al. (1994) was adapted and extensively modified. Genetic algorithms are conceptually simple and of proven value in solving complex, non-linear problems (Goldberg, 1989; Davis, 1991) but the application of genetic algorithms to specific real world problems is non-trivial.

The measurement and evaluation components consist of a number of problem specific routines. The utility functions are based on two theoretical models taken from the conservation biology literature. The models use simple concepts and are simple to implement - hence they run quickly and are easy to debug. In the core area model, patch shape has a large effect on patch utility (Laurance, 1991); and in the minimum cumulative resistance model, both shape and position relative to other patches are important (Knaapen *et al.*, 1992).

GAPD was implemented as a series of linked functions coded in the C programming language (Schildt, 1991). GAPD was initially programmed to run on a micro-computer (i.e. a PC) and, with minor modifications, was later converted to run on a UNIX workstation. GIS raster functions for patch analysis were coded in C as part of GAPD. GAPD is loosely coupled with the IDRISI GIS which is used to produce geographic datasets for input to GAPD and for display and further analysis of the outputs.

1.4.3 Testing

Testing was done in two phases. The feasibility tests were designed to evaluate the basic concept of optimal patch design in raster GIS and the GAPD methodology. The first series of tests looked at issues of representation and measurement of patch configuration

in raster data structures. Further tests looked at the representational power of the PRG algorithm and the efficiency of the genetic search driver.

GAPD was further evaluated by testing against a number of artificial trial problems of varying complexity. Although the complexity of a problem is not strictly quantifiable high spatial variability in the data, complex or compound utility functions and multiple patch solutions are characteristics of complex problems. Simple problems have low spatial variability, simple utility functions and are constrained to have single-patch solutions. Four broad types of problem are identified:

- Single patch-single objective (SPSO);
- Single patch-multiple objective (SPMO);
- Multiple patch-single objective (MPSO); and
- Multiple patch-multiple objective (MPMO).

To give some idea of the efficiency and effectiveness of the system, the results of the trials were compared with results that were obtained using other heuristics. Results were also evaluated by considering the size of the search space and by estimating upper bounds to the solutions.

1.5 Structure of the thesis

The conventional components of GAPD are discussed in the opening chapters. Chapter 2 deals with multi-criteria decision-making and search. Chapter 3 is a review of landscape ecology and conservation biology. Chapter 4 reviews some applications of GIS to decision-making and sets out the optimal patch design problem. Genetic algorithms are dealt with in Chapter 5.

The logical and physical models of GAPD and its implementation are described in Chapter 6. GAPD was developed first as a prototype for solving single patch problems using a single input layer and then as a fully functional system. The full system can solve multi-patch problems and can handle multiple objectives and multiple data layers.

GAPD was tested in two phases. Feasibility testing of the GAPD concept is described in Chapter 7. The second phase, the evaluation tests, are described in Chapter 8. Five hypothetical problems, using real datasets, were designed to test the effectiveness and efficiency of GAPD in a range of situations. The fifth test is a variation on a case study in Kathmandu originally done by Eastman *et al.* of the IDRISI project (1993).

The final chapter summarises the achievements of the research and sets out an agenda for further work.

1.6 Summary

Designing optimal patches in raster GIS is a complex problem because the size and shape of patches and the spatial relations among them are all criteria. When landuse planning can be done without regard to the composition and configuration of patches the conventional decision-making methods can be applied because there are no dynamic criteria. Suitability analysis using GIS is sufficient because each cell is a viable alternative and the problem is simply a matter of scoring each cell. This is no longer the case when configuration is relevant. In the simplest case there will be a constraint on minimum patch size and individual cells are no longer viable alternatives. Instead, the viable alternatives are patches or clusters of cells.

The techniques and ideas used in the thesis are drawn from GIS, genetic algorithms, decision-making, landscape ecology and conservation biology. The key feature of GAPD is the use of PRG to translate between the string domain of the genetic algorithms and the spatial domain of the GIS. GAPD was tested using a number of hypothetical problems. In each case a unique utility function was devised and integrated into GAPD. The test problems relate largely to conservation but GAPD can equally well be applied to other spatial problems such as multi-objective landuse planning, particularly in forestry and watershed management, urban planning, and facility location in cases where the shape of the catchment is a criterion.

2. DECISION-MAKING

2.1 Introduction

This chapter is an overview of decision-making methods particularly those which are relevant to the design of patch configurations. Optimal patch design is a complex spatial problem involving the integration of numerous spatial and aspatial criteria. There are an infinite number of possible patch configurations and the key issue in decision-making terms is the generation of good alternative candidate solutions. An efficient search method is needed to address this aspect of the problem. Conventional multi-criteria decision-making methods can be applied to evaluate alternative solutions and choose between them, thus driving the generation process. The spatial nature of the problem demands the use of raster GIS and this chapter also includes a review of some of the ways GIS are used for decision support.

In conservation applications, patches designed by decision makers must be realised through management practice, for example periodic felling, restoration of vegetation, fencing off to prevent grazing and so on. Chapter 3 says more about how patches as design artefacts relate to and differ from physical patches resulting from natural ecological processes. The special nature of optimal patch design as a spatial problem is set out more thoroughly in Chapter 4.

2.2 Overview of decision science

Decision science is concerned with the application of logical and rational techniques to complex real world problems to facilitate good decision-making. Applying formal methods also makes the decision-making process explicit and repeatable so that interested parties can see how and why particular decisions have been made. Decision-making occurs in two stages: (1) the strategic level where the goals and objectives are defined and the criteria are identified and (2) the operational level where a methodology is chosen and applied. This research is concerned with the operational level, with

developing, implementing and evaluating a methodology for designing optimal patch configurations.

Early decision-making techniques were developed to address logistical and economic problems and were highly deterministic (Wagner, 1972). With the help of fast digital computers, decision science is now applied to a broader range of situations. The problems tackled have become more complex and decision-making tools are now more varied and less deterministic (Minch & Sanders, 1986). Computer packages for decision-making are now integrated in decision support systems that allow a more flexible, exploratory approach to decision-making (Densham, 1991). The range of spatial problems that are addressed using formal methods and GIS includes site location, optimal routing, location-allocation and suitability assessment. Many aspects of spatial decision-making using a combination of GIS and formal decision-making methods are discussed in great detail in the literature (Jankowski, 1995; Pereira & Duckstein, 1993; Carver, 1991; Eastman *et al.*, 1995).

Complex decision problems are characterised by multiple incompatible criteria, multiple conflicting objectives and uncertainty. A well-structured problem is one that always yields the same result whereas in a loosely-structured problem the result will vary depending on the people solving the problem, because of their assumptions, values and judgements, and on the methods used (Minch & Sanders, 1986). In such cases there is no single best solution and in some situations it may be sufficient to find a satisfactory solution instead of a global optimum. There are other factors to consider in choosing a solution method particularly the time and resources available. In real applications as opposed to theoretical situations these are important pragmatic and practical considerations.

At the operational level all decision-making methods involve three steps:

- identification of alternatives
- evaluation of alternatives
- selection of a single alternative as the solution.

Decision science has produced many multi-criteria decision-making (MCDM) methods to evaluate and compare alternatives. When there are large numbers of alternatives search techniques are used to identify promising alternatives. Efficient search techniques rely on feedback information about the relative value of alternatives. MCDM and search methods are discussed more fully in subsequent sections.

Before discussing techniques in detail certain other issues must be mentioned. Risk and uncertainty are important considerations in any decision-making context. There is uncertainty in the physical attribute measurements, the evaluation functions that convert measurements to scores, the relative weighting or ranking of criteria and possibly also in the objectives. Certain methods make an assumption that criteria are independent and when this is not valid there is added uncertainty. If the criteria are not independent there will be epistatic or non-linear effects that magnify the uncertainty because small changes in attributes can generate disproportionately large changes in resultant utility (Goldberg, 1989). Finally, there are temporal considerations. Most decisions have consequences and dependencies that extend into the future. For example, when deciding where to locate a nature reserve the decision may be influenced by predictions of climate change and about future landuse requirements, which are important because nature reserves are long term features. Consequently there is uncertainty in the final decision and, therefore, a risk that the decision is not actually the best.

Methods for dealing with risk and uncertainty include techniques using probability (Goodchild et al., 1992), fuzzy sets (Leung, 1988; Burrough, 1989; Burrough & Heuvelinck, 1992; Van Gaans & Burrough, 1993) and artificial intelligence (Chen et al., 1994). However, these are all limited because they ultimately rely on subjective judgement. Perhaps the best way to handle uncertainty is within a decision support context in which software packages facilitate interactive cooperative exploration of problems and allow multiple users to cooperate in a structured and controlled way. Spatial decision support systems (SDSSs) make recommendations and help users to make decisions but do not take decisions (Densham, 1991; Sprague Jr & Watson, 1993). They can be used to generate promising alternatives for further examination (Dibble &

Densham, 1993).

This thesis is not concerned directly with uncertainty issues except to say that the operational context of any decision support tool such as GAPD must be considered.

2.3 Multi-Criteria Decision-Making

2.3.1 Overview

The terminology in the literature is not always consistent and in the context of this thesis MCDM is used to embrace all techniques designed to handle situations with multiple criteria or multiple objectives. Other related terms used in the literature are multi-criteria evaluation, multi-objective programming and goal programming. These terms are not synonymous and a single term is used here for simplicity. Essentially MCDM is about the evaluation and comparison of disparate entities.

In MCDM the problem is how to compare alternatives based on seemingly incompatible criteria. Since any method of comparing alternatives inevitably involves arbitrary judgements, all methods are open to criticism. However, formal methods provide a framework for decision-making which can be useful in justifying decisions as well as aiding the decision-making process. Such a framework gives a structure, a degree of repeatability, and breaks the problem into components where arbitrary judgements are easier to make because they involve a single criterion. Judgements are explicitly stated along with any assumptions, so that the decision can be explained and justified to interested parties.

The first step is to assign a score for each alternative against each criterion. There are two types of criteria: factors, which modify the suitability of an alternative and can be given a score on an interval or ratio scale; and constraints, which distinguish between feasible and infeasible alternatives so that an alternative either satisfies or violates a constraint (Eastman *et al.*, 1995). Once all criteria have been evaluated the next step is

to compare alternatives. There are two basic approaches: **compensatory** methods and **non-compensatory** methods. The compensatory approach assumes that criteria are compatible and that high scores for one criterion can compensate for low scores against another criterion. Non-compensatory methods do not allow a trade-off between criterion scores. Both approaches employ a decision rule to identify the best alternative. In both approaches alternatives that do not satisfy constraints, i.e. ones that are infeasible, are eliminated from further consideration.

Multiple objectives add another level of complexity to problems. As with multiple criteria the difficulty lies in the fact that objectives are expressed in different terms. Methods for tackling multi-objective problems are similar to those used for multiple criteria. Objectives can be ranked so that the problem is reduced to a series of single objective problems. Compensatory methods can be used to combine utility scores against individual objectives into a single utility score. Multi-objective decision-making methods are well documented (for example, see Tamiz, 1996).

2.3.2 Compensatory methods

In compensatory methods, the scores for each criterion are arrived at by converting criterion values to standard scores. Standard scores should be on an interval scale to allow for meaningful trade-offs and can be arrived at in several ways (Pereira & Duckstein, 1993). Criteria relate to physical attributes such as slope, rainfall or altitude. Attributes are measured on various scales and in different dimensions; measurements are converted to utility scores on standard scales. Using standard scales means that the scores are independent of the units of measurement and can be directly compared with other scores. The relationship between attribute values and standard scores will vary for each criterion and need not be linear or monotonic. The conversion between attribute measurements and standard scores can be achieved using mathematical functions, look up tables or graphical methods. Utility scores typically vary from totally unsuitable (usually 0), to perfectly suitable (usually 1). Standardisation is problematical because the utility is an interpretation or judgement of a physical measurement. Standardisation can

be done by defining a conversion function in consultation with experts. A step-wise function to convert interval scale attribute measurements can be found using the following iterative process from Pereira and Duckstein (1993).

- First, identify the attribute values that represent the maximum and minimum utility, and the midpoint between them.
- Second, identify another set of midpoints between the points already defined.
- Repeat the second step until the form of the function is sufficiently well defined. A similar process can be used to define conversion functions for attributes measured on categorical or ordinal scales. A non-monotonic function can be built up out of linear sections, each of which is derived using the above method. Figure 2.1 shows a non-monotonic stepwise utility function.

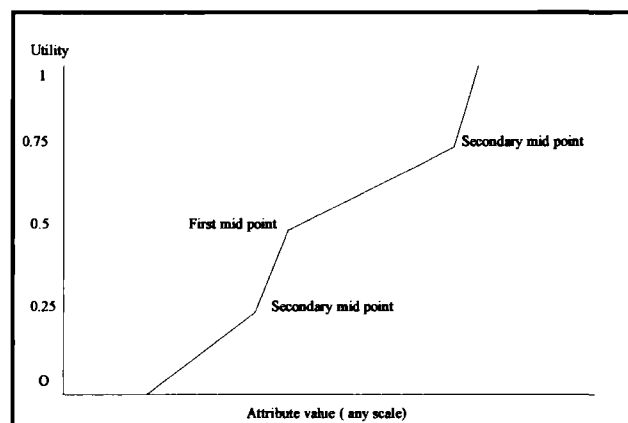


Figure 2.1 A non-monotonic stepwise linear utility function.

In raster GIS the most common compensatory method for comparing alternatives employ some kind of weighted summation technique (Jankowski, 1995; Pereira & Duckstein, 1993; Eastman *et al.*, 1995). One such technique is compromise programming. **Compromise programming**, also known as **ideal point analysis** (Pereira & Duckstein, 1993), employs the concept of a multi-dimensional suitability space with a unique ideal point in that space. The suitability space has one dimension for each criterion. The ideal point is defined as having ideal suitability for all factors and represents an ideal solution. There is an implicit assumption that each factor is independent but this may not always be true. If the ideal solution does not actually exist then the best alternative solution is the one that is nearest to the ideal in suitability space using some metric. The distance

from the ideal is a measure of relative suitability for each alternative. Alternatives that violate any constraints are eliminated.

Once the attribute measurements have been standardised, the distance from the ideal point can be calculated by combining the criterion scores into a single score using a weighted combination. Some factors may be more important than others in determining combined suitability, in which case equal distances in suitability space are not equivalent in different dimensions. The relative importance of individual factors is reflected in the weights assigned to them. Determining weights is again problematical and Eastman *et al.* (1993) describe a method and a function in the IDRISI GIS (Eastman, 1993), that determines weights based on a series of preference choices.

Taking together the ideas of a utility space, standardised utility scales and relative factor weights, the score S_a of an alternative a , is calculated by equation 2.1. The utility score for factor i is X_{ia} for alternative a and X_{i*} for the ideal point. The weights are denoted by W_i and there are N factors. The alternative with the minimum score is the winner.

$$S_a = \left[\sum_{i=1}^N W_i^p \cdot (X_{i*} - X_{ia})^p \right]^{\frac{1}{p}} \quad \text{Equation 2.1}$$

The weights are normalised so that

$$\sum_{i=1}^N W_i = 1 \quad \text{Equation 2.2}$$

Inclusion of the exponent, p , generalises the equation. The value of p controls the relative contribution that distance from the ideal point makes to the overall suitability. When $p=1$ the utility is linearly related to distance so that the difference in utility between two points is the same as the difference in their distance from the ideal point. There is

a perfect trade-off between factors because an increase in distance on one factor can be compensated by an equivalent decrease in another factor. If $p > 1$ then utility decreases more rapidly with distance, so that large distances are relatively more important. As $p \gg \infty$ then the evaluation approximates to a minimax rule where the suitability of an alternative is only as good as its worst score against all criteria (Pereira & Duckstein, 1993). Pereira and Duckstein (1993) found that $p=10$ was high enough to be effectively ∞ . Often the scale is such that the optimal score is 0 and the worst score is 1.

In ideal point analysis, the alternative with the smallest score is the winner. A mathematically equivalent technique simply takes a weighted combination of standard scores to derive a single overall score for each alternative. The alternative with the highest score is the solution. This method is called weighted summation.

An alternative compromise technique is **Concordance-Discordance** analysis (Carver, 1991). This method performs a pairwise comparison of each alternative against all the others to derive a dominance ranking for each alternative. The concordance measures the degree to which one alternative is superior to another for each criterion and the discordance measures the degree to which it is inferior. By combining the concordance and discordance scores for all criteria each alternative is given an overall ranking. The highest ranking alternative is the optimal solution. Because of the combinatorial explosion, concordance-discordance analysis is sensitive to the number of criteria and alternatives. It is not usually used where there are large numbers of alternatives and for this reason is not recommended for use in connection with raster GIS (Jankowski, 1995).

2.3.3 Non-compensatory methods

There are several MCDM methods that do not allow any trade-off between different criteria. These include lexicographic, hierarchical, conjunctive, disjunctive and dominance methods (Jankowski, 1995; Carver, 1991). Each criterion score is examined individually and a decision rule is used to compare alternatives.

The lexicographic and hierarchical optimisation methods require that the criteria be ranked in order of importance but do not assign relative weights. They are iterative techniques whereby the most important criteria are used first. In the lexicographic method the highest scoring alternative on the most important criterion is the winner. If there is a tie then all other alternatives are eliminated and the contest continues with the next ranked criterion until there is a winner. In the hierarchical method alternatives are eliminated if they fail to meet the threshold score for the current criterion.

The conjunctive and disjunctive methods also use thresholds but without ranking criteria. An alternative is eliminated if it fails to meet the threshold on a single criterion (conjunctive) or on all criteria (disjunctive).

The Dominance method also treats each criterion as equal. An alternative, A, dominates another alternative, B, if A scores at least as highly as B on all criteria and higher than B on at least one criterion. B is then dominated by A and is a dominated alternative. All dominated alternatives are eliminated and the non-dominated alternative is the winner. It may be that there is no solution using the dominance method since there may be more than one non-dominated alternative.

In general non-compensatory methods are applicable to problems with relatively few alternatives and these methods often will not identify a unique solution.

2.4 Search methods

Decision-making problems can be represented by a decision space. Each alternative in a decision-making problem can be represented by a point in an N -dimensional decision space, where N is the number of relevant attributes. The decision-making process is then a search through the decision space looking for the best alternative. Typically, each alternative has a single utility value so the utility can be thought of as an N -dimensional surface. For a utility function using two attributes, the decision space can be visualised by analogy with a topographic surface. The x and y coordinates are the criteria and the

height values are the utility. Peaks are local optima and the highest peak is the global optimum. Search spaces with many local optima are difficult to solve.

Search methods can be classified according to how they search the decision space. There are four basic methods: enumerative, random, deterministic and heuristic (Goldberg, 1989).

The **enumerative method** simply visits every point in the decision space and then chooses the best. It is an exhaustive search and is bound to find the optimum eventually. The enumerative method is suitable for problems when there are few alternatives. However, for many search problems it is inefficient and, therefore, impractical.

Random search visits a number of points at random and chooses the best. Random search has no guarantee of finding the best or even of finding anything much better than average. Enumerative and random methods are both blind searches.

Deterministic and **heuristic** methods make use of some knowledge of the decision space and are useful when the search space has a high degree of regularity, ie. when neighbouring points have more similar utility values than distant points. Using the topographic analogy again, the surface is smooth and continuous not jagged. Deterministic methods partition the search space, discarding segments until only the optimal point is left. Heuristic methods also partition the space but do not guarantee to find the optimal value. There is always a chance that the true optimum has been discarded. For many problems, there are no deterministic methods because the search space is too complex. Heuristic methods are often the only feasible approach.

The form of the decision space is a guide to choosing an appropriate solution method: it also helps to assess the difficulty of a problem. The difficulty of a problem depends on the complexity, the granularity and the extent of the decision space. Complexity is a measure of the degree to which neighbouring points, which have similar attributes, have similar utility scores. The extent of the decision space defines the range of possible

solution values and the granularity dictates how finely solutions can be distinguished. Thus, the extent and granularity determine how big the space is, that is, how many different possibilities there are. When digital computation is used a theoretically continuous space becomes discrete with a finite granularity. The size of the decision space is an important practical consideration. Complexity is critical when deciding on a solution method because each method works better in different types of decision space.

Many heuristics are based on the **hill-climbing** concept. Starting with a single guess, for which a utility value is calculated, small changes are made to the guess and the utility is re-evaluated. If the changes improve the solution they are accepted and new changes are tried. This process continues until further changes do not improve the utility. Hill-climbing methods work for fairly simple search spaces but in very complex spaces they are prone to find local optima and get stuck without finding the global optimum.

In recent years a number of new generic search and optimisation methods have been developed. Neural networks (Grossberg, 1988), simulated annealing (Cagan & Mitchell, 1993) and genetic algorithms (Goldberg, 1989) are methods which mimic natural processes within computers. Neural networks mimic the way the brain processes information and they have proved valuable in applications such as pattern recognition. Simulated annealing is analogous to cooling metals to toughen them. Simulated annealing algorithms allow random jumps in the decision space to explore different possibilities. The size of the jumps are slowly decreased so that the algorithm converges on a solution. Genetic algorithms use a universal heuristic based on evolution and natural selection. These methods have all proved superior to conventional methods in some applications because they are able to avoid being trapped by local optima. Genetic algorithms will be discussed more fully in Chapter 5.

It is usually assumed that each point in the search space has an associated utility score. This means that if multiple criteria are involved then a compensatory method such as compromise programming must be used to calculate a single utility value. With genetic algorithms this is no longer necessary as will be shown later. The genetic algorithm can

operate with feedback which simply ranks two alternatives without giving absolute utility values. The computer system developed in this research project does not use non-compensatory methods but there is no reason in principle why it should not. In the chapter describing the implementation of GAPD it will be seen that a compensatory method fits in more easily with other GAPD components. Alternative implementations of GAPD could equally well use non-compensatory evaluation methods.

2.5 Decision-making and decision support in GIS: A review

GIS can be used for decision-making in two ways, either as decision support systems in their own right, or as auxiliary analysis tools loosely coupled with decision support software. GIS do not have strong decision-making functionality and the loose coupled approach is often used. One advantage of loose coupling is that the GIS is used for spatial analysis, where it is strong, and the output can be fed into a number of different, specialist decision support packages. Thus, MCDM methods which are not easy to implement in GIS, or which are already well supported by existing software, can be used. The most appropriate method rather than the one most convenient to implement in GIS can be chosen.

GIS functionality for decision-making is variable. The main areas in which GIS have some functionality are: optimal routing, location-allocation, site location and suitability assessment. The **optimal routing** problem is how to find the best route, usually the quickest or cheapest, which connects a number of given points. Optimal route problems can be solved for networks in a vector GIS, and to a lesser extent on rasters, using conventional algorithms that are incorporated in GIS. The ARC/INFO GIS (Environmental systems research institute, 1992) has optimal routing functions for networks and IDRISI (Eastman 1993) has a function for finding the minimum cost path between two points on a raster.

In a **location-allocation** problem the objective is to select supply sites from a number of potential sites to serve a number of demand locations. The supply sites must satisfy the total demand and the objective is to minimise the total cost of satisfying the demand. The location-allocation problem can be solved using a network data structure i.e. the supply and demand sites are nodes connected by lines. There are a number of heuristic solution methods (Densham & Rushton, 1992). ARC/INFO includes functions for location-allocation (Environmental Systems Research Institute, 1992).

In **site location** the objective is to find the best site according to a number of criteria. Generally, GIS have the functionality to measure the attributes of sites but are lacking in functions for converting attribute values to scores and for applying decision rules to identify the best site. IDRISI, however, has functions for the weighted summation compensatory technique plus modules which help in assigning weights and in assessing risks.

Site location problems usually involve suitability assessment. **Suitability assessment** uses MCDM techniques to assign a score to each location. The score is a measure of the location's suitability for some purpose. Suitability analysis can be performed using the basic GIS buffering and overlay operations. The IDRISI GIS includes a number of functions that facilitate MCDM. The IDRISI MCE function is a user friendly interface to an ideal point analysis using weighted overlay. Other routines are available to help users to assign relative weights to different criteria and to handle risk and uncertainty. The routines facilitate suitability assessment but do not apply generally to more complex planning problems. Suitability assessment may be the first step in a patch design problem.

More generally, GIS are integrated with other decision-making packages. Carver (1991) coupled the vector ARC/INFO GIS with purpose-written FORTRAN decision support software. In Carver's problem the objective is to identify the best location for a waste disposal site subject to various criteria and the conflicting views of a number of interested parties. GIS overlay functions are used to identify feasible alternative sites and

each site is scored against each criteria. The scoring process creates a matrix of rows (sites) and columns (criteria). The matrix elements S_{ij} are the normalised weighted utility scores for site i according to criterion j . The utility matrix is input to the FORTRAN program which can perform ideal point analysis, hierarchical optimisation or concordance discordance analysis.

Chuvieco (1993) describes a planning problem where the objective was to determine an optimal land reallocation strategy given constraints on the total amount of land that could be reallocated. A GIS was used at two points in the analysis. First the GIS was used to identify the total area of land available. This data was fed into a linear programming software package called QSB that determined how much land should be reallocated. However, the package did not specify which land should be allocated. The GIS was involved again to identify actual locations using simple functions such as distance buffering.

Xiang (1993) has also used GIS with linear programming. In this example there are two conflicting objectives and Xiang uses the technique of mitigation costs, or a penalty function, to convert the problem into a single objective. The linear programming component identifies land parcels and the GIS is used to identify conflicts. The system runs in a DSS context.

Sharifi (1992, 1993) used ARC/INFO in a novel way to solve a land allocation problem by formulating the problem as a network distribution problem. His technique is innovative but does not extend the functionality of GIS. Eastman *et al.* (1993) demonstrate the use of IDRISI decision-making functions in a landuse application. They wish to assign a number of cells in a raster to one of two uses in such a way as to optimise overall landuse. To do this they first assess the suitability of each cell for each of the two alternative landuses. They then employ an heuristic to resolve conflicts between the objectives.

2.6 Summary

Methodologies exist for handling complex decision spaces and problems with multiple criteria and multiple conflicting objectives. To some extent these methodologies have been coupled with or integrated with GIS. However, available techniques still fall far short of addressing the optimal patch design problem. Individual techniques have a valuable contribution to make but they need to be linked together in a single system that can exploit the advantages of each. The crucial element, the functionality to address patch configuration as a planning criterion, does not exist. Spatial arrangement is handled adequately in network analysis and location-allocation but in both these cases the spatial elements are points and lines. Algorithms for these problems can be implemented in vector GIS. The patch design problem requires a raster structure because criteria scores vary continuously through space and because the physical extent and shape of patches has an important influence on their utility.

3. OPTIMAL PATCH DESIGN AND CONSERVATION

3.1 Introduction

GAPD is a computer system and it treats optimal patch design as a problem of locating optimal clusters on raster maps. This chapter looks at real world situations where configuration and composition of spatial units is critical. Conservation biology provides many examples of these types of problems. A key issue in conservation is habitat fragmentation which reduces habitats to isolated patches. The utility of a habitat patch is a function of its composition, its configuration and its context in the landscape. The quality of the habitat at the location is critical but the spatial properties of size and shape, and the spatial interactions with other patches, are also significant. Studies in conservation biology, ecology and landscape ecology have lead to the identification of a number of principles and guidelines for nature reserve design. These guidelines can be translated into formal criteria which can be evaluated using GIS functions and incorporated into a decision support tool.

Most conservation biology and landscape ecology is descriptive rather than prescriptive (Shrader-Frechette & McCoy, 1994). This is natural since there is still a lot of uncertainty surrounding conservation objectives and ecological processes. Where efforts are made to apply formal decision-making techniques to issues such as reserve planning they are essentially aspatial.

3.2 Ecological background

3.2.1 *The fragmentation problem*

Habitat fragmentation and loss of habitat are major threats to conservation (Saunders *et al.*, 1991; Morrison *et al.*, 1992; Doak *et al.*, 1990; Lord & Norton, 1990; Quinn & Harrison, 1988). Throughout the world, agricultural development and forestry has led to a situation in which natural and semi-natural habitats are becoming fragmented into

small patches which are surrounded by a matrix of developed land. The consequences of this have not been quantified but there is a widely held belief that many species and habitats are under severe threat.

Much research in ecology shows that in fragmented landscapes, spatial factors affect ecosystem functions in many important ways. Different researchers have identified a number of guidelines concerning the size and shape of reserves and the spatial relationships between them. These guidelines are uncertain, specific to particular species or ecosystems and sensitive to the conservation objective. However, the guidelines can be used as a basis for designing evaluation functions which can be implemented in raster GISs and used in conjunction with decision-making methodologies. There are examples of GIS and decision-making techniques being employed to improve nature reserve selection strategies (Davis *et al.*, 1990; McKenzie *et al.*, 1989; Margules & Stein, 1989). These methods concentrate on aspatial factors and, in view of the importance of spatial factors, there is a need to develop tools which incorporate spatial factors in nature reserve planning.

3.2.2 Conservation strategies and goal setting

Conservation problems cover a wide range of objectives, strategies and scales, from management of individual species at a small scale to strategies for preserving biodiversity at a global scale. Ecosystems are complex and dynamic; ecosystem functions are modified by spatial and temporal factors. Inevitably, uncertainty surrounds conservation objectives and the means to achieve them.

One major strategy for conservation is to set up nature reserves and other protected areas. Nature reserve design is a spatial problem; it is concerned with the location of reserves and with complex spatial relationships and their effects on ecosystem functions. Typically, nature reserves are sited in areas where the habitat is relatively unspoiled, and often where there is little pressure for other economic land use, such as agriculture, forestry or mineral extraction. Increasingly, habitat fragmentation is a major factor in

decreasing the value of otherwise suitable habitat. The main consequences of habitat fragmentation which are detrimental to overall habitat value are the reduction in size of habitat patches such that they are not able to support viable populations, the isolation of patches from each other, and the exposure of habitats to interactions with surrounding areas. Habitat patch size and spatial relationships, both with other patches and with the surroundings, should all be taken into account in evaluating habitat suitability and, hence, in deciding where reserves should be located.

Nature reserves are areas which are set aside for conservation and managed in a particular way. Initially, a nature reserve may be physically indistinct from its surroundings but in many cases the administrative boundaries will correspond to physical boundaries and the reserve will be a recognisably distinct physical unit. Reserves that start out as indistinct from their surrounding are likely to become distinct over time. Reserves can be put into three categories depending on the landscape structure: (1) Selected reserves, (2) Rescued reserves and (3) Restored reserves. In the first category there is more potential habitat than is going to be protected and the planners can select areas to be protected. Other areas will be used for forestry, agriculture, etc., with the result that the selected reserves eventually become isolated. Initially, suitable habitat is the matrix surrounding patches of non-habitat, but once the reserves are designated the situation is reversed. In the second category, the potential habitats are already isolated and there is little choice: habitat is patchy in a contrasting matrix. In the third category, the amount of habitat is less than is needed so areas must be restored. As the restoration takes effect, for example by planting to create woodlands or by removing drainage to recreate wetlands, the contrast between patch and matrix increases and over time the reserve will become a physically distinct patch.

Conservation can be, and is, addressed in aspatial ways, through legal protection of species, economic incentives to follow certain practices, and publicity campaigns aimed at changing people's behaviour. These aspatial strategies may be independent of, or related to, a nature reserve strategy. While they are important, they cannot adequately address the issue of conserving natural habitats and ecosystems.

3.3 Applications of decision-making techniques to conservation

Conservation planning is done at the strategic and operational level. There are numerous conservation objectives and appropriate strategies (Smith & Theberge, 1986; Usher, 1986; Lomolino, 1994; Leader-Williams *et al.*, 1990; Game & Peterken, 1984; Theberge, 1989).

Two well-documented reserve selection methodologies, **gap analysis** and **representative reserve selection**, are based on the premise that all species should be preserved and, consequently, a network of reserves should be established which includes habitat for every species. The gaps in gap analysis refer to those species which are unprotected, or are insufficiently protected (McKendry & Machlis, 1993; Davis *et al.*, 1990). Unprotected species are prioritised for protection and GIS are used to identify areas of habitat which should be protected. By overlaying data layers in a GIS it is possible to map habitat suitability and highlight areas of rich potential. The total area of different habitat types can be calculated. Where existing nature reserves coincide with a habitat it can be concluded that the habitat and associated species are protected. Where rich habitat lies outside nature reserves the species are unprotected. Habitat types that are relatively unprotected are "conservation gaps" and are targeted for protection. The primary use of the GIS is to produce suitability maps for the priority habitat types that then serve as the information basis for locating reserves.

The representative network approach is based on the same principle but employs a different method (McKenzie *et al.*, 1989; Kirkpatrick, 1983; Pressey & Nicholls, 1989; Nicholls & Margules, 1993). The idea is that reserve networks should be planned as an integrated package rather than in an *ad hoc*, one-by-one, manner to optimise the protection of species. An iterative selection algorithm chooses a new reserve at each step until all species are protected. Every time a reserve is selected the list of unprotected species is modified so that each reserve is chosen on the basis of which species are still unprotected. This algorithm can be used to select reserve networks which protect all species at least once or some minimum number of times. The actual

algorithm used has been criticised on technical grounds with a suggestion that integer programming should be used instead (Underhill, 1994).

Neither gap analysis nor representative reserve selection places much importance on spatial (as opposed to locational) factors, although Nicholls and Margules (1993) do use spatial relationships as a tie-breaker between otherwise identically suitable reserves. The gap analysis GIS system could also take account of spatial relationships by scoring reserves according to their isolation from other sites and their size. Gap analysis and representative reserve selection are strategic whereas optimal patch design is at a lower, more detailed, tactical level. Objectives identified using strategic methods should ideally be implemented using spatially explicit planning tools.

3.4 Reserve Design - designing optimal patches

3.4.1 *Composition and configuration*

In optimal patch design the problem is to design networks of patches subject to spatial and aspatial criteria. There are two aspects: composition and configuration. Composition is aspatial, or locational, and can be addressed by mapping. The spatial component, configuration, requires knowledge of ecosystems and landscape dynamics derived from conservation biology and landscape ecology. Landscape ecology provides a number of concepts and principles which are useful in formulating the problem. Specific models which can be applied as utility functions are provided by conservation biology.

3.4.2 *Spatial factors*

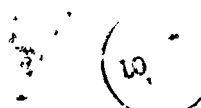
The recognition of the importance of spatial factors in habitat quality has stemmed largely from an interest in conservation among biologists and ecologists. Landscape ecology and conservation biology are two different perspectives on the same problem. The term landscape ecology is used in two senses. In the first sense, it is a holistic study

of the landscape as an entity which has structural and functional components. In this meaning landscape ecology places greater emphasis on human factors, on the role of people in the landscape and their response to it. In the second sense, landscape ecology is taken to be ecology studied at the landscape scale. In this thesis it is used in its first sense. Conservation biology applies to all scales and, therefore, encompasses ecology at the landscape scale.

3.4.2.1 Landscape ecology

Landscape ecology is an holistic study of the landscape as a complex system of interacting components. Forman and Godron (1986) distinguish landscape ecology from ecology by saying that whereas much of ecology is focused on vertical (within patch) relationships, landscape ecology concentrates on horizontal (between patch) relationships, and is thus concerned principally with spatial relationships. The holistic view is that the functional interactions between and within landscape elements are too complex to be analysed and, therefore, preserving the overall landscape pattern is the best way to conserve habitats and species.

Some of the terminology used in landscape ecology is inconsistent. The following basic definitions are taken from the FRAGSTATS program documentation (McGarigal & Marks, 1994). A patch is the basic element of a landscape. Patches are areas of "relatively homogenous environmental conditions". Patch boundaries are "discontinuities in environmental character ...". The matrix is "the most extensive and connected landscape element" within which smaller patches are embedded. Landscape structure is the spatial relationships between components. There are two aspects of structure: landscape composition and landscape configuration. Landscape composition is "the presence and amount of each patch type", an aspatial property. Landscape configuration is "the physical distribution or spatial character of patches". Configuration is spatial and relates to shape and pattern. The most problematic definition is the definition of landscape. There are different interpretations but, in the context of spatial planning, the landscape is equivalent to the area of study.



The concepts of landscape ecology are useful for describing and measuring certain aspects of pattern and spatial interaction using numerous indices of landscape structure. However, the indices are general and lack meaning in the context of specific applications. Landscape ecology does not address the question of site specific spatial planning because it is concerned with landscapes as a whole. Numerous alternative configurations will preserve the character of a landscape but the utility of the individual patches will vary enormously. Landscape ecology addresses the spatial aspects of the problem but not the locational aspects.

There are several theoretical bases for landscape indices. Dominance, diversity and contagion are based on information theory (Shannon & Weaver, 1962). Fractals have been applied to landscapes (With, 1994; Russell *et al.*, 1992; Leduc *et al.*, 1994; Benson & Mackenzie, 1995). Measurement methods based on percolation theory are useful for estimating an index of connectivity for landscapes as a whole (O'Neill *et al.*, 1992; Gustafson & Parker, 1992). There are other *ad hoc* indices such as the fragmentation index (Johnsson, 1995) and the proximity index (Gustafson & Parker, 1994). The various indices, including the fractal dimension, have been found to be dependent on scale and resolution (Cullinan & Thomas, 1992; Leduc *et al.*, 1994). A number of landscape ecology measurement functions are included in the GRASS GIS package (Baker & Cai, 1992). The FRAGSTATS software package includes many routines for measuring landscape indices (McGarigal & Marks, 1994).

To date the major contribution of landscape ecology to conservation lies in attempts to measure landscape characteristics and monitor changes. To a lesser extent, there is research into relating measures of landscape pattern to ecological indicators such as species presence and abundance and species diversity. Some attempts have been made to use landscape ecology in a prescriptive mode (Barrett & Peles, 1994; Fedorowick, 1993; Hansson & Angelstram, 1991; Knaapen *et al.*, 1992). Barrett and Peles (1994) applied the principles of agro-landscape ecology in an exercise to design habitats and land management regimes in an integrated area plan. In applying principles they do not use formal optimisation techniques, but work in an interactive framework. Fedorowick

(1993) applied landscape ecology principles to prioritise blocks of land but detailed planning for these areas was done conventionally. While there is a lot of support for the idea of applying landscape ecology principles to conservation and management they have not been applied in a truly prescriptive sense nor integrated into planning systems (Selman & Doar, 1992; Hobbs *et al.*, 1993; Forman & Godron, 1981; Theobald & Gross, 1994).

For GAPD the terminology of landscape ecology has been generalised across different scales. GAPD deals with four levels of spatial arrangement. The cells in a raster are the lowest level and the raster itself (i.e. the landscape) is the highest level. Contiguous cells with shared properties make up patches which are located within a context in the area of study. Individual patches are at the second level and networks of patches are at the third level. Each spatial element at each level has both a composition and a configuration. Cells have identical configuration but their composition can vary. A cell's composition may be a single value from a suitability layer or a vector of attribute values from multiple data layers. Patch configuration encompasses shape and spatial relations with other patches. Patch composition can be expressed in numerous ways that summarise the cells composing the patch. Landscape composition and configuration have the usual landscape ecology meanings. These ideas are developed further in Chapter 4.

3.4.2.2 Conservation biology

Conservation biologists have used empirical studies, computer simulations and ecological theory to derive some principles on which to base conservation decisions. Conservation biology is concerned at a small scale with individual species and communities and at a large scale with biodiversity overall. A number of theories and studies reveal the importance of spatial factors in determining habitat suitability.

A **metapopulation** is a group of separate sub-populations that are distinct from each other but which interact to a greater or lesser extent. Actual metapopulations exist

somewhere in a spectrum of varying degrees of interaction between sub-populations. At one end of the spectrum there is no, or very little, interaction, and the sub-populations are almost isolated, while at the other there is almost as much interaction among sub-populations as within sub-populations. The degree of interaction depends largely on spatial factors and is thought to affect the long term viability and genetic make up of the whole population (Hanski & Thomas, 1994; Murphy *et al.*, 1990). Compared with single populations, sub-populations have greater genetic diversity and are less vulnerable to disease, predators, disturbance such as floods and fire, and stochastic events. Patches where sub-populations become locally extinct can be recolonised from surviving patches.

Studies of the life history of a **territorial** species, the northern spotted owl (Carroll & Lamberson, 1993), suggest that fragmentation of habitats may be beneficial given that the total area of available habitat is limited (Carroll & Lamberson, 1993; Lamberson *et al.*, 1994). Individuals of territorial species naturally disperse to find breeding sites and will not breed within a certain distance of other individuals. A single large patch will not support as many pairs as the same area divided into several small patches. The patches need to be close enough to allow individuals to move from one patch to another, but far enough apart to satisfy the territorial requirements.

At the interface between a patch and its surroundings, interactions result in a number of **edge effects**. For example, the micro climate of woodland patches can be quite different from their surroundings and there is a temperature and humidity gradient from the interior to the boundary (Laurance, 1991; Laurance & Yensen, 1991). Laurance (1991) quantifies the edge effect as the distance, d , over which the influence of the surroundings extends into the patch and has made empirical studies to determine values for d . Around the edge of each patch is a strip of modified habitat. If conditions in this strip are not suitable for interior species then the effective area of the patch is reduced. Laurance's core area model shows how the effective area varies with total area and particularly that the effective area becomes zero when the patch shrinks to a critical size. The ratio of core area to total area depends on the shape of the patch. Compact shapes have a larger proportion of core than elongated shapes or ones with contorted boundaries. Other edge

effects are the penetration of animals and plants into the patch from the matrix and disturbance by outside factors such as traffic noise and agricultural pollution.

Buechner (1987) has looked at a number of variables associated with edge effects including edge permeability, edge configuration, environmental gradient across the edge and directionality of interactions.

Heterogeneity is an important factor in habitat suitability for two main reasons. Firstly, some species require several different resources, for feeding, breeding, nesting and so on, and may not find all the resources in one patch. Individual patches may be homogenous, so individuals will need to access a number of resource patches to satisfy their needs, or the patch may be heterogeneous and contain all the resources. The second aspect is that species tolerate a range of conditions. When a species range covers an environmental gradient it may be advantageous to include as much of that variety as possible to preserve genetic diversity and guard against future changes, particularly climatic fluctuations and possible climate change. An example of heterogeneity being important is described by Hanski & Thomas (1994) in a study of butterfly habitat. Their investigations revealed that the butterfly laid its eggs on slopes with different aspects. In certain weather conditions one batch of eggs would survive and the other would not. Under another set of weather conditions the reverse happened. Thus, if the butterflies were confined to using only one slope aspect, adverse weather conditions in one year could cause extinction of the whole population.

The subject of **corridors** has caused some conservation biologists to debate whether they are desirable or effective (Shrader-Frechette & McCoy, 1994). An almost equally problematic concept is **connectivity** which is a functional relationship between patches and is distinguished from connectedness which is a measurable physical property (Fahrig & Merriam, 1985; Selman & Doar, 1992). **Isolation**, the converse of connectivity, is often used as an alternative expression of the same concept. **Corridors** are linear patches that function as conduits along which species can travel between patches. The habitat quality of corridors is important because some species require plenty of cover and

other resources while in transit. The spatial properties of corridors have been investigated using computer simulations by Soule & Gilpin (1991). They deduced that corridor width, length and sinuosity are significant. Width is important because of edge effects and the maximum length depends on the mobility of the target species. Sharp bends can deter certain species. Simulations also showed that directionality is important: a funnel shaped corridor with a wide entrance and a narrow exit was very poor while a horn shape, with a narrow entrance and a wide exit, was good. Corridor utility is highly species-specific to the extent that what constitutes a corridor for one species may be a barrier to another. Simulations by Henein & Merriam (1990) indicate that corridor quality may be a critical factor and that poor quality corridors may actually be worse than no corridors at all.

The **minimum viable population** for a species is the smallest number of breeding pairs needed to guarantee the long-term survival of that population (Morrison *et al.*, 1992; Gilpin & Soule, 1986). If the population is too small then it can be eliminated by stochastic population events, such as breeding all male offspring, or by loss of genetic health by inbreeding. A habitat patch must be of a minimum size to support a minimum viable population, hence the idea of a **minimum viable area (MVA)**. MVA is a critical size below which a habitat cannot support a viable population. This concept is also problematic since there is little empirical evidence to show what the MVA is for different species.

Natural systems are dynamic and disturbances, such as fires, floods and wind-blows, are now looked on as natural elements of ecosystems rather than as exceptional events that can be disregarded. **Disturbance regimes** can create equilibrium systems in which the pattern of vegetation remains constant although individual patches change (Turner *et al.*, 1993). The effect of a disturbance regime on a particular species, Kirtlands Warbler, has been studied by Probst & Weinrich (1993). Kirtlands Warblers inhabit patches of 5-23 year-old woodland. The survival of Kirtlands Warbler populations depends on regular fires to clear space in mature woodlands so that regeneration can occur. There are several implications in reserve design (Baker, 1992):

- reserves have to be large enough to maintain a natural disturbance regime and should be several times larger than the expected disturbance size;
- they should include disturbance initiation zones, so that disturbance can be managed, and export zones so that political consequences outside the reserves are mitigated; and
- buffer areas can be set up around reserves to prevent disturbances filtering out from a core area.

Disturbances are spatio-temporal phenomena and their spatial extent and the temporal interval between events are stochastically distributed. Both the expected interval and the size should be considered in designing reserves (Turner *et al.*, 1993).

Island biogeography relates the size and isolation of islands to the diversity of species found on the islands. According to the theory, several small islands support more species than one large island of equivalent total size. Quinn and Harrison (1988) found evidence of the same effect in National Parks in the USA. Others (Shrader-Frechette & McCoy, 1994) question whether the island biogeography model can be applied to fragmented habitats such as woodlots in agricultural areas.

3.4.3 *Summary of spatial influences*

The preceding discussion shows some of the ways that spatial factors affect ecological functions. It shows how landscapes are made up of elements defined by their environmental characteristics and their functions. The principal spatial factors are patch size, shape and pattern in the landscape. Landscape elements can be categorised by their form and function as: habitat patch, resource patch, corridor, buffer, barrier and matrix. Buffers are areas which protect habitat areas from interactions. Barriers prevent interactions between habitat patches. The spatial characteristics and functions of the different landscape elements are summarised in table 3.1.

| Landscape element | Function | Important spatial characteristics |
|-------------------|--|--|
| Habitat patch | Supports a viable population | Size Compactness Isolation/Accessibility |
| Resource patch | Utilised by populations for some needs | Size Accessibility |
| Corridor | Connects habitat and resource patches | Position Length and width Shape |
| Buffer | Mitigates matrix effects on a habitat patch | Position Width Orientation |
| Barrier | Protects patches from matrix effects. Isolates patches from each other. | Position Length and width Orientation |
| Matrix | Isolates patches | Not applicable |

Table 3.1 Landscape elements: their functions and spatial characteristics.

The basic spatial properties of patches and their influence on habitat suitability are summarised below.

- Size affects the suitability of habitat and resource patches. To be useful, patches must be above a certain minimum size. If total area is restricted, then it may be better to have several smaller areas instead of one large one. Utility may increase with patch size up to a limit beyond which there is no further improvement.
- Shape is the configuration of a single patch. It includes several aspects of a patch's dimensionality. Two important aspects of shape are compactness and elongation. Compact shapes have a short perimeter length, a large core area and are good for interior species. Long perimeters increase the relative amount of edge habitat to interior habitat. Elongated shapes are suitable as corridors and buffers. Length, width, sinuosity and orientation are important features of corridors. Elongation and orientation can also affect patch heterogeneity, since a shape which runs parallel to an environmental gradient will include a wider range of conditions than one which runs across it. The orientation may mitigate edge effects, for example the patch may lie parallel to the prevailing winds, or

be oriented north-south and thus less subject to insolation.

- **Pattern** is the configuration of a network of patches or of a landscape. The spatial arrangement of patches affects their connectivity and conversely their isolation. Isolation can be useful to protect against disease and pests and promote genetic diversity. Connectivity mitigates against local extinction by facilitating recolonisation, allows species to exploit distributed resource patches and allows dispersal of territorial species.

Each of these parameters has a theoretical optimal value which depends on the conservation objective, the species concerned and the environmental characteristics.

In discussing size, shape and pattern it is assumed that there is a relatively sharp contrast between the two sides of a patch boundary. This assumption is justified by considering the circumstances in which reserves are demarcated, as discussed in section 3.2.2.

3.5 Summary

Spatial factors influence ecosystem functions and habitat suitability in many different ways. Guidelines from conservation biology and concepts of landscape ecology can be used together to formulate optimal patch design problems and to specify objective criteria. Later chapters will show that two basic measurements, size and distance between patches, can be used as building blocks for evaluating patches. It will be shown that these building blocks can be implemented very easily in a raster GIS and integrated into GAPD.

The spatial criteria and models used in conservation biology can be transferred to other application domains. Mention has already been made of other physical systems such as forests and river basins. GAPD is also relevant to social and economic applications. Catchment areas for facilities can be thought of as patches with criteria relating to total size, composition and maximum distance from the site. Using a raster layer of population and other socio-economic data the size of the patch can be measured as geographic area or population served. The catchment is analogous to a territory or

habitat patch but instead of containing various resources it must satisfy various demands. As for habitats the patch must be of sufficient size to be viable. Travel distance to the facility can be a factor limiting the maximum distance of the boundary from the site. The composition may be more complex than simply total population, for example if different socio-economic groups need to be included in the catchment, as might be politically desirable with educational facilities. The same idea can be extended to locating multiple facilities serving different types of catchment, for example different types of store to cater for different markets. Each catchment would have a different composition and different constraints on travel distance. Another possibility arises if there are not sufficient facilities: they should be dispersed so that there is no bias towards certain areas and, consequently, criteria relating to distance between sites become relevant. GAPD could also be used analytically for example in crime analysis. Here, the objective is to identify patches of activity that already exist and then designate those areas for special action. Various criteria could be used to define what constitutes a patch and where large patches should be dis-aggregated or small patches merged. Patches should be homogenous in other attributes besides crime. A distance rule could be used to determine when small neighbouring patches should be merged and when they should be treated separately.

4. OPTIMAL PATCH DESIGN ON RASTER MAPS

4.1 Introduction

This chapter describes the patch design problem, its complexity and shows how it differs from other GIS decision-making problems. The problem is broken down into manageable components which are discussed in detail. Three key issues are identified: generating promising solutions, evaluating solutions, and searching a complex search space. These issues are dealt with in general terms. The final section describes the conceptual design of an operational system which utilises novel techniques and integrates search algorithms with GIS. The components used to implement the conceptual design in GAPD are identified. Subsequent chapters describe the implementation and test of GAPD in greater detail.

4.2 The patch design problem

Designing optimal patch configurations using a raster GIS is a complex problem. It is both a MCDM problem, because the utility depends on multiple criteria, and a search problem because there are many alternatives which cannot be identified *a priori*. When the size of a patch is much greater than the cell size of the raster then the shape of the patch becomes a meaningful criterion. Size, shape and pattern are relevant criteria in many ecological contexts and should be taken into account when planning nature reserves (Morrison *et al.*, 1992). Similar criteria can be significant in other contexts, such as planning housing developments (Tomlin, 1990).

The simplest patch design problem is to find a single patch which optimises a single objective. The patch's utility is a function of its composition and configuration. Finding the best patch involves a trade-off between aspatial and spatial criteria.

The single patch problem is a specific case of the general patch design problem. In general, the problem is to locate multiple patches to optimise multiple objectives.

Patches will vary in type, size and configuration and the number of patches will be unknown. Utility will depend on the environmental attributes and spatial features of the patches and the spatial relationships between them. The solution will depend on the application and on the data.

Tomlin (1990) touches on the patch design problem implicitly when discussing the application of map algebra to decision-making, or prescriptive modelling. He distinguishes between atomic and holistic criteria. Atomic criteria relate to each cell independently. They are locational, in that they depend on the physical attributes of cells, which vary with location, but they are aspatial, in that they are not affected by the suitability scores of other cells. For example, distance from a river is a locational criterion because the distance of a cell from a river is independent of the distance of other cells from the river. Holistic criteria on the other hand are spatial because the suitability score of a cell is influenced by the scores of other cells. For example, if the objective is to plan a housing estate and a waste treatment works, and one of the criteria relates to the distance between the two sites, then there is a problem. The distance of the works from the housing estate can only be computed once the housing estate is sited. Therefore, the distance criterion cannot be evaluated without knowing the solution. Atomic criteria can be handled relatively simply using MCDM methods whereas holistic criteria are much more problematic. Tomlin sets out some other examples of this type of problem and proposes some heuristics for solving them. However, these are rather crude and are not likely to give high-quality solutions. The decision-making exercises described by Eastman *et al.* (1995), Carver (1991) and Chuvieco (1993) (see Chapter 2 section 3) are all based on atomic criteria.

In their discussions of MCDM and GIS, Jankowski (1995), Pereira and Duckstein (1993) and Eastman *et al.* (1995) have all assumed that individual cells are alternatives but this is not the case for patch design. Feasible solutions are clusters of contiguous cells and, therefore, there is at least a contiguity constraint. There may also be criteria relating to compactness and shape which are even more complex to handle at the individual cell level. If the suitability of a cell depends on it being a member of a contiguous group of

cells of a certain total size, then the overall score for cell x depends on the score for its neighbouring cell y; but the score for y cannot be determined without knowing the score for x. The same circular argument applies to all cells so that in principle the problem cannot be solved. This circularity means that an heuristic approach should be adopted. It is likely that the cells with the highest atomic suitability are not contiguous and, therefore, the best patch will incorporate lower-scoring cells. Hence there needs to be a trade-off between cell suitability and patch suitability.

In Tomlin's (1990) terminology atomic criteria are static and holistic criteria are dynamic. The terms static and dynamic criteria are preferable because atomic and holistic may suggest a distinction between cell and patch. The real distinction is between criteria which can be evaluated once, and are therefore static, and criteria which are re-evaluated for each alternative and are therefore dynamic. Static and dynamic criteria must be treated differently. The most obvious way of processing static criteria is to generate a suitability map in which the value of each cell is a suitability score. Raster GIS are well provided with the necessary functions and there is an established tradition of solving planning problems via suitability mapping. Many problems involve both static and dynamic criteria and suitability mapping is a good first step. However, suitability mapping is not an adequate technique for dynamic criteria and must be considered as a pre-processing step to generate input to another method.

For dynamic criteria there is a choice between two approaches: problem specific heuristics and generic search algorithms. Patches are groups of contiguous cells. The number of possible patch configurations in a raster is huge compared to the number of cells because of the combinatorial explosion (see Appendix IV). Because the number of alternatives is so large, search methods that evaluate all possible alternatives are impractical. Neither heuristics nor generic search methods necessarily guarantee optimal solutions but they should produce solutions better than random guesses. The problem with heuristics is that they are application specific, are difficult to design and may make use of poor assumptions. The problem with search algorithms lies in the representation of alternatives.

An efficient search algorithm requires a utility function to evaluate the dynamic criteria and provide feedback to guide the search process. Application-specific search heuristics which handle dynamic criteria adequately can be difficult to develop and the same may apply to utility functions. However, the problem is actually simplified by using a generic search algorithm because the search process is separated from the evaluation function. The benefits of this arrangement are explained further in section 4.5.

4.3 Cell suitability assessment

Suitability assessment can be performed using the compromise programming or weighted summation technique described in Chapter 2. In a raster suitability map, a cell which has the ideal value for each factor is an ideal cell, whereas a cell which has less than the ideal value for any factor is not an ideal cell. The GIS weighted overlay function combines scores for different criteria and a Boolean overlay, or mask function, eliminates cells which fail to satisfy a constraint. Compromise programming is the preferred method for suitability assessment in raster GIS because it assigns a value to every cell and is simple to implement using GIS map algebra (Tomlin, 1990) functions. Concordance-discordance analysis would be computationally expensive since every cell would have to be compared with every other cell for every criterion to allocate a rank. Non-compensatory techniques are not applicable because they do not assign a relative suitability to each cell.

Suitability can be evaluated using methods other than multi-criteria evaluation. Knowledge-based methods and expert systems can predict areas where certain species are most likely to occur. Aspinall and Veitch (1993) have used Bayesian inferencing to identify red deer habitat. Fabricius and Coetzee (1992) employed an artificial intelligence technique ("iterative dichotomising") to habitat mapping. Neural nets have been applied to suitability assessment by Wang (1994). Another method of land suitability evaluation uses fuzzy sets (Burrough, 1989; Burrough & Heuvelinck, 1992; Van Gaans & Burrough, 1993). Fuzzy suitability scores and error estimates on suitability values are important ideas. However, apart from acknowledging their relevance, the work

presented in this thesis makes exclusive use of crisp values without errors. The question of error handling is a major GIS research issue (Burrough, 1986; Burrough & Heuvelinck, 1992; Goodchild *et al.*, 1992; Drummond & Ramlal, 1992) and beyond the current scope of this project.

In the context of optimal patch design problems, those methods which assign a score to each cell are not only the simplest to apply but also the easiest to integrate with dynamic criteria. However, an advantage of search over application-specific heuristics is that it is easier to integrate other MCDM methods for assessing static criteria into a single generic system.

4.4 Heuristics

Dynamic evaluation is problematic when trying to develop problem-specific heuristics. Tomlin (1990) proposes some examples in which he essentially substitutes static criteria for dynamic ones. For the housing estate and waste treatment example quoted above, one could generate a layer showing distance from the cells most suitable for housing. This could then be treated as a static layer. This sort of approach uses intelligent guesses about the form of the solution. The assumption is that the housing estate will be located where the suitability is greatest. Making such assumptions limits the potential for trade-offs between different criteria. In more complex situations, heuristics can be difficult to develop.

An interesting recent development is a novel technique using cellular automata to simulate the emergence of zones in a bottom up manner (Garriga & Ratick, 1996). The method tends to produce acceptable landuse patterns. Currently the algorithm lacks formal evaluation of different patterns and a stopping criterion but development of the method is continuing.

4.4.1 *The Iterative Relaxation heuristic*

The land allocation problem referred to earlier, described by Eastman *et al.* (1993), is a multi-objective problem in which a cell can be allocated to only one of two (or more) possible landuses, A or B. A certain number of cells must be allocated to each landuse. When a cell scores highly for both uses there is a conflict and the possibility that allocating the cell to its most suitable use, say A, will result in a much inferior cell being allocated to B, with a consequent reduction of overall utility. The problem is actually complex but a simple relaxation heuristic tends to find good solutions. The heuristic works by setting suitability thresholds for each landuse, and selecting all cells which exceed the thresholds. When the thresholds are too high, not enough cells are selected to satisfy all objectives so the thresholds are lowered until a solution is found. The heuristic, which can be called iterative relaxation (IR), is aspatial; it optimises composition but configuration is irrelevant.

The IR method can be adapted for the patch design problem. In the simplest case a contiguity constraint is added which states that cells must constitute a single patch. At each iteration the selected cells will be distributed in contiguous clumps of different sizes in a pattern which depends on the spatial correlation between the various factors and on the threshold. The threshold is lowered until at least one group of contiguous cells satisfies the size constraint. This heuristic maximises the minimum cell suitability in a patch so the clusters will have high utility for aspatial criteria but it ignores spatial criteria other than contiguity. Consequently, this algorithm has several limitations which become more apparent as stronger configuration criteria are applied: (1) it only considers size and ignores shape and pattern; (2) there is no trade-off between cell suitability and patch suitability; (3) it treats size as a constraint when it may be better treated as a factor; (4) lowering the threshold can have a catastrophic effect in which several large patches appear, all of which are larger than the desired size; (5) the large patches may be too large if there is a maximum size constraint; (6) the patches may contain holes which are not acceptable. The relaxation algorithm will tend to find patches that have high cell suitability but may score very poorly for patch suitability, especially when shape and

pattern are important criteria. The IR method is used in some examples in Chapter 8 in comparison with GAPD.

4.4.2 *Region-growing heuristics*

A more promising approach is to use region-growing which overcomes the problem of overly large clusters and holes. There are numerous region-growing algorithms used in image processing (Sonka *et al.*, 1993). Edge-detection techniques identify region boundaries by finding edges where the difference between neighbouring cells is great. Two complementary methods, top-down and bottom-up, use a hierarchical structure. Top-down algorithms start with large regions (possibly the whole image) and successively divide into smaller regions using a set of rules. Bottom-up approaches start with small regions, or seeds (possibly a single cell), and aggregate them together. In image processing the rules are intended to partition the image such that the cells within a region are similar to each other and different from the cells in other regions. The region-growing approach can be adapted as a patch design technique by using rules which generate promising regions. Promising regions will have high suitability scores for individual cells, will satisfy size constraints and will have good shape properties. A simple distance rule would force patches to be compact.

Parameterised region-growing (PRG) is a technique for generating patches with high utility for both spatial and aspatial criteria (Brookes 1993, 1997a, 1997b). PRG combines two components: Simple Region-Growing (SRG) and Parameterised Shape-Growing (PSG). SRG is an aspatial, iterative, bottom-up, region-growing process. Starting with a single cell, the region grows by adding the neighbour with the highest suitability score until the desired size is reached. PSG uses the same process but the scoring procedure is different. Each neighbour cell is given a score which reflects how well that cell would contribute to the desired shape of the region. In PRG the overall score for a cell is a weighted combination of its cell score and its shape score. The PRG program is controlled by a number of parameters which control the size and shape of the region, the initial cell for region-growing, and the trade-off between patch configuration

(dynamic criteria) and composition (static criteria).

PRG was designed and implemented as a stand-alone program and shown to find better patches than the IR algorithm for some hypothetical problems (Brookes, 1997a). However, there are several limitations to this supervised mode of operation. Choosing a parameter set which optimises the region is problematical. Parameters cannot be determined *a priori*. Optimal parameter settings will depend on:

- the texture of the raster,
- the perceived relative importance of configuration and composition,
- the resolution and extent of the raster.

Parameter settings are also interdependent since, for example, an optimal size at one location will not be optimal at another because of spatial variation over the raster. A further limitation of the original PRG is that it applied only to single patch problems. PRG could be used iteratively to solve multi-site problems but this approach is rather unsatisfactory. PRG is a generic technique because the cell and shape scoring rules can be adapted to many different situations.

4.5 Search

An approach using search avoids some of the difficulties associated with heuristics. Search methods were discussed in Chapter 2. Efficient search methods rely on feedback about the quality of candidate solutions to guide the search towards an optimal solution. Therefore, search algorithms depend on evaluation functions. The use of evaluation functions is actually an advantage of search methods over other heuristics because dynamic criteria are handled in a straightforward way. Search examines alternatives and re-evaluates the criteria for each one. To take the same housing example quoted above, each alternative is a map with a housing estate patch and a waste treatment patch. The distance between the two is measured exactly. The use of a search technique circumvents problems associated with trying to evaluate dynamic criteria at the cell level and means that they are evaluated at an appropriate level. A number of primitive evaluation functions which could be used as building blocks for more complex functions are

described in Chapter 6. A further advantage of search is that different MCDM methods can be employed. Suitability mapping is no longer an automatic starting point. Although the early implementations of GAPD, described later, operate on suitability maps, later versions work on multiple data layers and some do not involve any static criteria at all.

The main difficulty in applying search is to find a suitable structure that represents candidate solutions in the problem domain and is tractable by the search algorithm. Within GAPD, PRG is the link between the search driver and raster GIS evaluation functions.

4.5.1 Measuring spatial properties

Dynamic criteria relate to both composition and configuration and to all spatial scales. They relate to cells, patches, patch networks and whole landscapes. Composition and configuration are properties which exist at all spatial levels. In a raster all cells have the same shape but the configuration of a cell can be construed as its contiguity with other cells with the same composition. Measurements of spatial properties can be divided into simple measurements and spatial indices. Size and boundary length are examples of simple measurements, fractal dimension is a spatial index. Table 4.1 summarises some examples of the types of measurements which might apply at different scales.

| Spatial level | Composition | Configuration |
|---------------|--|---|
| Cell | A single suitability value Multiple attribute values | Contiguity |
| Patch | Size (number of cells) Sum of cell scores Minimum of cell scores Diversity of cell attributes | Shape Fractal dimension |
| Patch network | Number and type of patches Size of each patch type Summary of component cell values | Inter patch distances Patch isolation |
| Landscape | Dominance index Diversity index | Contagion index Percolation index Fractal index |

Table 4.1 Levels of spatial arrangement

At all scales, composition is a function of the composition of individual cells. Cell composition may be a single value, as in a suitability map, or it may be a vector of attribute values from multiple data layers. At larger spatial scales, composition is a summary of component cell values. It may be a total, minimum, maximum or other summary value such as diversity.

The fundamental configuration property is cell contiguity. A patch is a set of contiguous cells which share some property. Contagion, percolation and fractal landscape indices are functions of cell contiguity. Spatial properties at the patch level are size, shape and relation with other patches. Superficially, these properties may seem easy to measure because there are many standard indices in general use. However, apart from size, which has a straightforward meaning and can be measured in a raster GIS simply by counting the number of contiguous cells in a patch, the situation is not so simple. The problem is in defining exactly the property to be measured in a way which relates to the real world. Shape and pattern are difficult; they are both difficult concepts to define exactly and are usually reduced to a number of properties. For the purpose of evaluating habitat patch shape the following properties are important: boundary configuration, elongation and orientation. Boundary configuration is usually measured in terms of how well the shape approximates to a circle of the same area - a circle is the most compact shape and

has the simplest boundary. Three expressions of boundary configuration are the PI index, the roundness and the development. Elongation and orientation relate to long thin shapes. Elongation expresses the ratio of length (the long axis) to width (the short axis). Orientation is the direction of the long axis. Ecologically significant aspects of pattern are: isolation and connectivity, which are properties of individual patches, and diversity, dominance and contagion which are properties of landscapes as a whole.

4.5.2 *Evaluating patch utility*

There are numerous options for measuring composition and configuration. Converting these measures into criterion scores is problematic. Various methods for converting measurements into scores and evaluating alternatives using MCDM methods were discussed in Chapter 2. However, it is questionable to what extent the various measurements relate to actual, meaningful, spatial characteristics and, in turn, to utility.

Often the relationship between composition and utility is clearer than that between configuration and utility. A key insight is that configuration affects composition. Therefore, meaningful utility functions can be derived from composition measurements. Configuration affects utility but is not measured directly. This is demonstrated in the examples described in Chapter 8.

The conceptual design of GAPD does not limit it to any particular MCDM methodology or type of spatial measurement. In practice, the implementation of GAPD for this research made use of simple measurements rather than indices. The chosen measurements relate to criteria derived from the conservation biology literature.

4.6 Overview of the GAPD conceptual design

It has been shown in the preceding discussion that the patch design problem is both a MCDM and a search problem. A generic tool for optimal patch design must be based on an efficient search algorithm. Searching for optimal patches on raster maps raises

three issues: (1) how to measure patch properties; (2) how to evaluate patch properties; and (3) how to generate feasible alternatives. These three issues have been addressed in the previous sections.

At the conceptual level GAPD combines four elements in a single powerful, robust and flexible unsupervised method. The four elements are: a search driver, a translation function, a measurement function and an evaluation function. This design is shown schematically in figure 4.1. The search driver navigates the decision space using feedback information to converge on an optimal or near optimal solution. Feedback is provided through the other three components. The translation function converts between the internal representation used by the search engine and the raster representation used by the measurement function. The measurement function measures attributes and the evaluation function awards a value based on those attributes. Each component is a black box to the others. Alternative implementations of each component are possible without affecting the others.

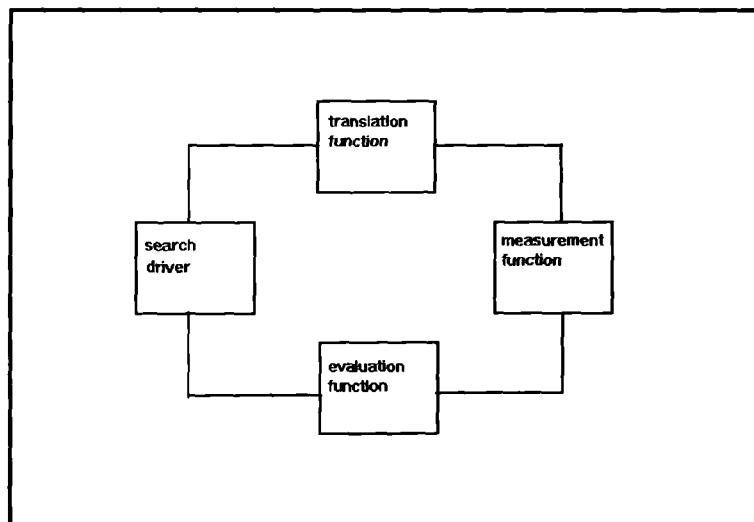


Figure 4.1 GAPD conceptual design

PRG is the key to implementing GAPD. PRG is an heuristic for generating promising alternatives rather than a true optimisation technique. Using PRG, a string of numeric parameters is mapped to a raster representation of a patch. PRG can, therefore, form

a bridge between search techniques which operate on strings of numbers, and a raster GIS which has the functionality to evaluate sites.

GAPD integrates PRG, raster GIS functions, MCDM methods and a genetic algorithm search driver. Genetic algorithms were chosen because they are robust, general-purpose problem solvers which can be successful in the kind of complex search space associated with patch design. Integrating a genetic algorithm with a raster GIS via PRG combines the three essential elements for solving the problem: an efficient search mechanism, a means of generating alternatives, and tools for evaluating alternatives. Thus, with reference to figure 4.1, the genetic algorithm is the search driver, PRG fulfils the translation function, GIS fulfils the measurement function and MCDM fulfils the evaluation function. The genetic algorithm approach is flexible and can be generalised to tackle multi-patch and multi-objective problems.

The next three chapters describe the components of GAPD more fully. Evaluation functions are problem specific whereas the genetic algorithm and PRG are more generic. Chapter 5 describes the theory and operation of genetic algorithms. The details of the implementation of GAPD are contained in Chapter 6, which includes a number of examples of evaluation functions and their implementation. Chapter 7 discusses some of the issues of measuring configuration on raster systems.

5. GENETIC ALGORITHMS

5.1 Introduction

Like many innovations, genetic algorithms have a longer history than might be supposed from the recent explosion of interest. Genetic algorithms were originally developed in studies of artificial life and simulated evolution (Holland 1975). When their power as general purpose search engines was recognised they were applied to real world optimisation problems (Goldberg, 1989). In the process there have been many enhancements including hybrid algorithms which combine genetic algorithms with other search techniques. Genetic algorithms are a promising technique to apply to optimal patch design because the search heuristic is both efficient in complex decision spaces and independent of the problem domain.

The canonical genetic algorithm, which is described below, forms the basis of all genetic algorithms and encapsulates the principles of adaptive search. Krzanowski (1997) has provided a taxonomy of evolutionary algorithms which classifies alternatives to the canonical genetic algorithm. The canonical genetic algorithm was used in GAPD for the pragmatic reason that it is the simplest and most researched model. In comparison to other variants it is simple to implement, is domain independent and has few operational variables. Recently Krzanowski (1997) has reported a different genetic algorithm approach to a similar spatial problem. Although in theory genetic algorithms are powerful general-purpose problem solvers, their application to problem solving is not always straightforward and there are a number of operational issues.

5.2 History and background

The first genetic algorithm and the early theory of genetic algorithms was developed by Holland (1975). Originally the aim was to simulate evolution by natural selection, through the operation of genetics, and hence to understand mechanisms of adaptation.

In natural organisms the information that determines how the organism develops and its form is encoded in genes. Genes are physically stored in string-like structures called **genotypes**. The external form of the organism is the **phenotype**.

Organisms pass on modified versions of their genes to their offspring. In sexual reproduction, both parents contribute genetic material and the genotype of the offspring is a mixture of the genotypes of the two parents. There are two mechanisms whereby the offspring's genes differ from the parents. The first mechanism is **crossover**. During reproduction genotypes from each parent line up side by side, and a number of crossover points divide the strings into segments. Between crossover points the segments swap over to make two new strings, each with elements from the originals. After crossover there is no new genetic material, because all the genes come from the parents, but there are new structures and, therefore, new genotypes and new phenotypes. Extra variety arises through the second mechanism, **mutation**. A mutation is a random change to an individual gene. The probability of a mutation is low and most change in the genotypes occurs through crossover.

Differences in the genes of the offspring and parents lead to differences in their phenotypes. Similarly, there are differences between all individuals in a population. Individuals which are better suited to their environment have a better chance of breeding and producing offspring. By producing more offspring, well adapted individuals contribute more genetic material to subsequent generations than do poorly adapted individuals. In this way, the quality of genetic material in subsequent generations rises.

The power of crossover arises because genetic codes are epistatic; combinations of genes rather than individual genes are important. Crossover creates new combinations of genes; if these combinations lead to better individuals, then they will be passed on to succeeding generations. Crossover is the principal mechanism in adaptation by natural selection and mutation is relatively unimportant by comparison (Goldberg, 1989).

5.3 Genetic Algorithm Fundamentals

Holland (1975) modelled genes using binary strings and used computer programs to perform crossover and mutation. There are theoretical justifications for using binary strings but it is possible to use other strings or even other data structures (Holland, 1975; Goldberg, 1989; Davis, 1991). Simple crossover occurs at a single point with a fixed probability and mutation occurs at any point with a much lower probability. Given two strings, A and B, and a crossover point, X, two new strings, A' and B', are given by:

| | |
|--------------------|---------------------|
| String A; 10100111 | String A': 10100001 |
| String B: 11001001 | String B': 11001111 |
| | ^ |
| | X |

A genetic algorithm starts with a population of strings and generates successive new populations known as generations. Each generation consists of survivors from the preceding generation together with offspring from breeding pairs in the previous generation. The population size is kept at a constant level, so some individuals are eliminated to make room for offspring. Pairs of individuals are chosen from the population using a stochastic selection procedure and bred with each other using a stochastic reproduction procedure. Survivors are also chosen using a selection procedure. The selection procedures operate on phenotypes while the reproduction procedures operate on genotypes. The analogy with natural selection is in the use of a **fitness function** that operates on the phenotype and determines a relative fitness for that individual. The relative fitness is used by selection procedures to choose which individuals breed and which survive. The final element in the genetic algorithm is a conversion function which translates a genetic code into a phenotype.

The **schema theorem** of Holland (1975) is a useful model for understanding the power of genetic algorithms and the reason why binary strings are preferred. The **cardinality** of a string is the number of values each string position can take. Thus a binary string has

a cardinality of 2 because each element can be 0 or 1. A **schema** is a pattern which can be matched by many actual strings. A schema is expressed using a wild-card or don't care symbol, *, where the * can be matched by any value (either a 1 or a 0 in a binary string). Thus the schema 1*0* can be matched by four strings:

1101 1100 1001 1000

Just as a schema can be matched by different strings, each actual string belongs to more than one schema. This idea is important because even though an offspring differs from both its parents it shares schema with them. Furthermore, multiple schema descend from the parents to the offspring. For example, if the parents are 111000 and 101101 and the offspring are 111101 and 101000, then four schema 111***, ***000, 101*** and ***101 have survived from the first generation to the second. If any of those schema imparts high fitness then one of the offspring inherits high fitness from the parents.

According to Holland (1975), any system in which schema with above average fitness have a proportionately higher representation in the succeeding generation is an adaptive system. Although the genetic algorithm explicitly operates on strings it implicitly operates on schema. In this way genetic algorithms are implicitly parallel because, each operation on a pair of strings actually operates on many schema. Holland (1975) showed that, for a binary alphabet, in processing N strings, the number of useful schema processed is proportional to N^3 . This theorem is also very well explained in Goldberg (1989). Both authors demonstrate that using a binary alphabet instead of one with higher cardinality means that each string belongs to more schema, and consequently there is more implicit parallelism.

Schema can be thought of as building blocks and the schema theorem can be seen intuitively. To take a trivial example, an alphabet with cardinality 8 and a string length of 1, can represent 8 phenotypes: 0,1,2,3,4,5,6,7. These all belong to the same schema *. To represent the same phenotypes with an alphabet of cardinality 2 (i.e. in binary) would require a string of length 3: 000, 001, 010, etc. Each of these belongs to several schema: **0, *00, *0*, etc.

The building block idea can be translated directly into a decision-making context. Each schema defines a section of the decision space. As highly-fit schema spread through the population, then the search becomes concentrated in a smaller section of the decision space.

A comparison of binary and higher cardinality alphabets also illustrates another point. Assuming that crossover is the only operator (since mutation occurs with very low probability) then, in order to have a chance of finding a global solution, the initial population must contain enough genetic variety to be capable of generating all possible phenotypes. With a binary alphabet and a string of length 4, an initial population of 2 can contain enough genetic variety to generate all 16 phenotypes; if the initial population is 1111, 0000 then subsequent generations can be: 1000, 0111; 0100, 1011; 0011, 1100; etc. With an alphabet of cardinality 4, an initial population of 2 could not generate all phenotypes using simple crossover: for example two genotypes 14 and 23 can only generate 13 and 24. In practice, because populations are initially generated at random, a population greater than the notional minimum will be needed to guarantee total genetic diversity.

A major problem with genetic algorithms is that highly-fit schema are likely to be destroyed by crossover. This is a serious drawback, because it applies more to long schema than to short schema. The length of a schema is the maximum separation of fixed values; ****11****** has a length of 2 and **1****1**** has a length of 6. Obviously there is a greater chance that crossover will disrupt the schema of length 6. This is especially important because genetic algorithms are meant to be good at dealing with epistatic effects. High fitness for a schema like **1****1**** is implicitly epistatic, since if it is the combination of 1's in the appropriate places that creates the high fitness then neither **1******* nor *******1**** have high fitness. It follows that short, highly-fit schema are more likely to survive than long schema and this has implications for genetic algorithm design. Holland (1975) and Goldberg (1989) have more to say on schema length, fitness and epistatic effects.

5.4 The canonical genetic algorithm

Holland's original genetic algorithm is now also known as the **canonical genetic algorithm** (Whitley, 1994) or the simple genetic algorithm (Ribeiro Filho *et al.*, 1994) to distinguish it from numerous, subsequent variations. The canonical genetic algorithm is presented formally as follows:

Step 1. Initialise a random population of size P .

Step 2. Calculate the fitness of each individual in the population.

Step 3. Select N pairs of individuals at random with a probability proportional to their fitness (where $P > 2*N$).

Step 4. Breed offspring from each pair using crossover and add to the new population.

Step 5. Apply random mutation to each offspring.

Step 6. Select $(P-2N)$ individuals from the old population at random, with a probability proportional to their fitness, and add them to the new population.

Step 7. Replace the old population with the new population.

Step 8. Calculate the fitness of each individual in the population.

Step 9. Test the stopping criteria.

Step 10. If the stopping criteria are not met, return to step 3.

Different stopping criteria can be used, but usually the algorithm stops when there has been no increase in average utility for a set number of generations or when a certain number of generations have been processed. There are a number of variants in the way that individuals are selected for breeding and survival.

5.5 Genetic Algorithms for decision-making

5.5.1 *How genetic algorithms are applied in decision-making*

Genetic algorithms are a heuristic search technique that is analogous to natural selection. The idea of using genetic algorithms to solve optimisation and decision-making problems follows naturally. Organisms are well adapted to their environment and also extremely

complex. If they have evolved through natural selection, then the mechanisms of natural selection must be powerful, certainly much more powerful than random chance.

The genetic algorithm problem-solving strategy can be described as a directed random search. It is directed because fitness applies a selection pressure that guides the search to promising areas of the decision space. It is random because crossover and mutation are stochastic processes; they allow the search to make large jumps in the decision space and escape from local optima. The combination of directed and random elements is what makes genetic algorithms powerful.

In decision-making terms, a schema can be thought of as a hyperplane in a decision space; as a schema spreads through the population, the search becomes concentrated in smaller partitions of the search space. Genetic algorithms proceed through a trade-off between exploration and exploitation. The art of successful genetic algorithm design is in striking the right balance. The other key point about genetic algorithms is that they operate on a population of individuals and, therefore, search many parts of the decision space at the same time.

When genetic algorithms are used in decision-making, the number of generations, G , and the population size, P , must be chosen carefully. The population size is important for the theoretical reasons outlined above. The number of fitness evaluations is proportional to $(P \cdot G)$ and is important because it has a large impact on run-time. When run-time is constrained there is trade-off between population size and number of generations.

Transforming the genetic algorithm from a tool for studying adaptation into a decision-making tool introduces a number of implementation problems. Firstly the problem has to be expressed in suitable terms. In simulating adaptation, experimenters are free to choose fitness functions and can select phenotypes that are easy to represent in genetic codes. In many textbook examples the phenotype is a real number and the genotype is its binary equivalent. In decision-making, the phenotypes are defined by the problem. Finding a suitable genotype and conversion function is not simple; in fact it is possibly

the major difficulty. The genetic algorithm search heuristic operates on the genotype and is independent of the phenotype. The fitness function is in the problem domain where it operates on the phenotype independently of the genotype.

The key to successful genetic algorithm implementation lies in finding a suitable genetic code. Firstly, the code must be complete: that is it must be able to represent a sufficient variety of alternatives. Secondly, it should have low cardinality (ideally binary). Thirdly, it should lend itself to meaningful schema or building blocks of short defining length. There appears to be no rigorous theory that determines how to satisfy this condition; it is a matter of intuition, and experience. Having designed a suitable code, the next consideration is the utility function. This must obviously discriminate between optimal and sub-optimal solutions. It should also give some indication of the relative utility of sub-optimal solutions in order to apply selection pressure. If sub-optimal solutions are all equally bad, then there is no feedback to guide the search which would have to rely on hitting the target with a single shot.

5.5.2 When genetic algorithms are applied in decision-making

Looking at things from another point of view, what problem characteristics suggest that a genetic algorithm approach might be useful? One characteristic is uncertainty in the model or process. Another is a large, highly-complex decision space. The only information from the problem domain that a genetic algorithm makes use of is the fitness, or utility, function. Thus, decision-makers only need to be able to specify the attributes that make for a good alternative. In practice, many applications use hybrid algorithms that use problem specific auxiliary techniques to improve the performance of the genetic algorithm.

It is tempting to see genetic algorithms as a substitute for more conventional search algorithms whenever conventional methods are computationally intensive. Genetic algorithms do not guarantee optimal solutions and, therefore, exhaustive search or deterministic methods should be preferred when they are practical. Genetic algorithms

are capable of searching more efficiently than exhaustive search and, therefore, can be preferred on performance grounds. Because they are efficient, genetic algorithms can also be useful when uncertainty dictates that the same problem must be solved many times with different scenarios.

5.6 Review of some genetic algorithm applications

In many applications, the canonical genetic algorithm has been modified and improved to incorporate various tricks which improve the effectiveness and efficiency of the algorithm. By their nature, genetic algorithms do not guarantee to find optimal solutions. In part, this is because of the stochastic element and because of performance problems in processing large populations and many generations. A number of failings occur frequently. One such failing is premature convergence. It often happens that a genetic algorithm will evolve towards a sub-optimal solution and become stuck there. This occurs when the sub-optimal solution has high fitness relative to other members of a random, initial population and its genes quickly spread through the population causing a loss of genetic diversity. A number of techniques have been invented to cope with this. The following review of some applications illustrates several improvements to canonical genetic algorithms.

Genetic algorithms have been applied to a number of spatial problems. Pham and Onder (1992) tried to optimise the layout of components on a workbench and Mosetti *et al.* (1994) looked at siting wind turbines. Both used small matrices to model space. Andrey and Tarroux (1994) used a genetic algorithm to segment images with a classifier method. Classifiers are production rules which have a condition part and an action part. A classifier is rewarded by being activated and the utility function takes account of successful activations. The population of classifiers was distributed over a raster image. Each individual was only allowed to breed with a neighbour and this imposes spatial auto-correlation. Andrey and Tarroux found that the algorithm was only successful when they seeded the initial population with classifiers that were likely to give good results. Results with a randomly generated population were poor. Bhandarkar *et al.*

(1994) used a genetic algorithm for edge detection in raster images. They employed a two-dimensional genetic structure, a binary edge image, and also used a novel two-dimensional crossover operator. They evaluated a number of genetic algorithms and concluded that more effective genetic operators and an alternative genetic structure which would allow larger images to be processed were needed. Dibble and Densham (1993) used a genetic algorithm to generate many near-optimal alternatives in the context of a decision support system. The objective was to present decision-makers with alternatives from which to make a choice. Another approach to spatial problems was adopted by Dibble (1994) in an experiment to build an adaptive classification scheme that is capable of learning to recognise, among other things, suitable habitats for elephants and that automatically generates rules that vary depending on the spatial context.

Genetic algorithms are computationally expensive and can have long run times on sequential processors. Solution time can be dramatically reduced by using parallel processing. Genetic algorithms are naturally suited to being parallelised because, at each generation, the same utility function is applied to a number of individuals and this can be done in parallel. There are a number of models for parallelising genetic algorithms (Koza, 1992). Major considerations are the relative amount of data traffic and management overheads against the saving in elapsed processing time.

Genetic programming is an offshoot of genetic algorithms in which the genes are computer programs (Koza, 1992; Koza, 1994; Kinnear, 1994). The main difference between genetic algorithms and genetic programming is the structure of the genes. Selection and reproduction are performed in the same way and the utility function is a measure of how well the individual programs perform a certain task. However, whereas genetic algorithms typically use a string structure, genetic programs typically use a tree structure. The tree structure is made up of branches and nodes. Terminal nodes are atomic executable routines or variables. Other nodes are operators. The point about using tree structures is that, using a suitable notation, each tree structure can be guaranteed to be syntactically correct. Crossover is performed by selecting a node and swapping branches at that node. Thus, the offspring are guaranteed to be syntactically

correct. This has obvious advantages when the trees are executable programs that must be run in order to evaluate them. A second feature of tree structures is that they are of variable size and the parent trees may be different sizes from the offspring. This feature is very attractive and could be used by other genetic algorithms to replace the fixed length string structures which place limitations on the flexibility of solutions.

5.7 Summary of Operational issues

The successful implementation of a genetic algorithm is by no means a simple task. The theoretical justification for using genetic algorithm methods is persuasive, there are many examples of practical genetic algorithm applications, and yet there is a gap between theory and practice. Successful implementation is dependent on the effectiveness and the efficiency of the algorithm. An effective algorithm is one which is capable of converging to an optimal solution given sufficient resources. An efficient algorithm is one which will converge, in an acceptable timescale, given available resources. Although efficiency and effectiveness are highly inter-related, effectiveness relies on the choice of a coding and the way in which the problem is modelled, whereas efficiency depends on other operational factors.

5.7.1 Coding

A suitable genetic code is of primary importance. The coding must be representative and capable of manipulation. It must be able to represent the range of possibilities and have sufficient granularity to distinguish intermediate possibilities. The cardinality influences how the coding can be manipulated and the schema theorem suggests that the smaller the cardinality the greater the power of the algorithm to exploit implicit parallelism.

5.7.2 Fitness function

Of equal importance is the fitness function. The function must be truly representative of the system being modelled. It must distinguish the optimal solution from sub-optimal

solutions and be a reliable estimate of the relative merits of sub-optimal solutions. For an effective genetic algorithm, the fitness function must be accurate, reliable and sensitive. For an efficient genetic algorithm the function must be evaluated quickly.

5.7.3 Genetic Operators

The function of genetic operators is to explore new genetic structures but in doing so they destroy existing (potentially good) structures. The operators are applied in a stochastic manner and the rate at which they are applied can have a profound bearing on effectiveness and efficiency. If the rate is too high, then good structures are destroyed too quickly and the solution cannot converge. If the rate is too low, then the solution will converge too slowly and there may be insufficient resources to reach an optimal solution. The principle operators are crossover and mutation but a number of others have been used. The choice of which operators to use and how they are implemented can also have some bearing on performance.

5.7.4 Initial population generation

The question of how the initial population is generated is of more theoretical importance to experimenters in artificial life than to those who seek to solve optimisation problems. From a practical point of view it can be a good idea to seed the initial population with good guesses, which can be derived from more conventional methods. A purist would maintain that the population should always be random. A random population has more genetic diversity and, thus, can explore more of the solution space, but a seeded population can converge more rapidly. A hybrid population which is partially random and partially seeded is a compromise often adopted. The size of the population is very significant but there is no sound theory to say what the population size should be. A population which is too large means many evaluations at each generation, whereas one which is too small will not contain enough genetic diversity. There is a trade-off between population size, P , and number of generations, G , since the number of times the utility function is evaluated is $P \cdot G$. An argument in favour of seeding the population is that a

seeded population can be smaller than a random one. The disadvantage of using seeded populations is that the search is restricted and may not find a global optimum.

5.7.5 *Number of generations*

The number of generations required depends partly on population size (see 5.5.1). Fewer generations allow a larger population and hence greater genetic diversity. However, there is an intuitive argument that with more generations, a greater proportion of the individuals evaluated will be better than average. Thus, a greater number of more promising alternatives are explored. The trade-off between generations and population size may be problem dependent.

5.7.6 *Selection methods*

The selection strategy for generating a new population can affect performance. The two problems are over-selection and under-selection. Over-selection selects too many of the best individuals and may converge prematurely to a sub-optimal solution. Under-selection, on the other hand, selects too few and does not converge at all. Different selection strategies have been tried but it is not conclusive which is the best. Some algorithms vary the selection strategy as the solution evolves and attempt to match the strategy to the current population. In early generations, over-selection can quickly eliminate hopeless cases, whereas later on under-selection prevents single alternatives from swamping the population.

Fitness scaling, fitness ranking and tournament selection have all been tried to overcome over-selection and under-selection. In the simple genetic algorithm, the utility of the phenotype maps to the fitness of the genotype in a linear way. Parents are selected in proportion to their fitness. This technique is known as *roulette selection*. If fitness is proportional to utility then the term *utility is fitness* is used. By contrast, in *fitness scaling* the utility of the phenotype is mapped to a fixed predefined range of fitness values so that the relative fitness of the best and worst individuals is controlled. With

fitness scaling, roulette wheel selection can be used to choose parents with known relative probabilities of picking the best and worst individuals. This reduces the likelihood of over-selection when there are great differences in utility, or under-selection when there are small differences. Fitness ranking places each individual in order of fitness and then assigns a selection probability according to rank. Tournament selection is an alternative strategy for determining parents: pairs are selected at random and the individual with the highest fitness becomes a parent. In the simple genetic algorithm, all parents are replaced by their offspring.

One aspect of purist genetic algorithms which is not acceptable in decision-making is the possibility of the best individual being eliminated by chance. A number of methods have been proposed to prevent this. One is to heavily bias selection in favour of the best, by fitness scaling, but this may result in over-selection. The simplest techniques use a kind of insurance policy which guarantees that the best individual(s) survive. Another group of techniques use sub-populations. The total population is divided into a number of groups which, for the most part, breed amongst themselves but occasionally interact with other groups. The idea is to ensure the survival of the best while allowing the maximum diversity amongst the rest. Potts *et al.* (1994) developed GAMAS which uses 4 sub-populations, each with different characteristics. One sub-population consists of the champions which are periodically creamed off from the other sub-populations and put into an elite group. The places left by the elite members taken from the other groups are made up by new, randomly generated individuals. A similar configuration was tried by Dibble and Densham (1993) with an "Olympic games" held between the champions of the sub-populations.

5.7.7 Hybrid algorithms

Hybrid algorithms incorporate some knowledge of the problem domain into the genetic algorithm. The simplest hybrids are those that use seeded populations but otherwise function as pure genetic algorithms. More sophisticated hybrids make use of techniques like local hill-climbing. Periodically, individuals, usually the best N , are selected for local

searching. The genetic code of these individuals is perturbed in a small way and the utility re-evaluated. If there is an improvement, the new code is retained, otherwise the individual is left unaltered.

5.8 Application of genetic algorithms to optimal patch design

Genetic algorithms are general-purpose tools because the search engine is independent of the problem domain. The search engine only operates on strings and fitness values. Strings are fed into a black box translation component and fitness values are returned from a black box evaluation component. Applying a genetic algorithm to a real problem requires domain-specific translation and evaluation components. Within GAPD, the PRG process translates numeric strings into raster images that can be evaluated using GIS functions, as indicated in Chapter 4. Successful implementation of a genetic algorithm depends largely on a suitable string representation. A string should be capable of representing a very large number of alternatives. It should also be made up of building blocks, or sub-strings, which should correspond to different properties of the solutions. The parameter set of the PRG program satisfies these requirements as will be shown in Chapter 7. Consequently, the genetic algorithm can be applied to patch design on raster maps by integrating a genetic search engine with raster GIS functions and PRG.

The genetic algorithm approach is, therefore, practical. It is also promising because genetic algorithms have been proven successful in many complex problems. Patch design on rasters is complex with epistatic properties. As shown in Chapter 4, conventional and heuristic methods are inadequate even for problems with simple spatial criteria. For problems involving more complex spatial criteria, it is relatively easy to produce evaluation functions but much harder to invent heuristics. There are actually many variations of the optimal patch design problem and, consequently, no single heuristic method will be suitable for all cases. For this reason, a general-purpose technique is desirable.

Pragmatically, using a genetic algorithm approach for large, complex problems is

becoming easier. Genetic algorithm programs are now widely available either as source code or in packages. The code used in this research project is based on a C program published in a journal article by Whitley (1994). The SUGAL package from Sunderland University is a comprehensive package for researchers using genetic algorithms. The package allows users to control the genetic algorithm using selections from a wide range of options. Users code their own evaluation functions. The source code is also available so that if the standard options are not adequate, users can modify the programs to suit their needs. Computer processing power is still increasing, especially in parallel processing, so that potential performance problems are diminishing.

The implementation of GAPD is described in the next chapter. The design of the genetic algorithm component was kept deliberately simple, to make both the initial development and subsequent evaluation easier by avoiding unnecessary complexities. In keeping with the main objectives of the research, to implement a working system and evaluate it, enhancements similar to those described in this chapter are left as topics for further research.

6. DESIGN AND IMPLEMENTATION

6.1 Introduction

GAPD is a computer system for solving optimal patch design problems. Previous chapters have explained how the optimal patch design problem differs from other spatial planning problems and how conventional GIS and decision-making methods are inadequate. A conceptual model of GAPD was set out at the end of chapter 4. GAPD combines a genetic algorithm to search the decision space, a parameterised region-growing (PRG) process to translate one-dimensional genetic codes into two-dimensional patch maps, raster GIS functions to measure patch attributes and MCDM techniques to evaluate patch utility.

This chapter describes the logical design of GAPD and the physical designs that were implemented in evaluation versions of GAPD. GAPD is made up of four logical components, the genetic search driver, the PRG component and the measurement and evaluation functions. The innovative part of GAPD is the use of PRG as a translation function between aspatial and spatial representations of alternative solutions. Measurement and evaluation are performed by conventional GIS and MCDM methods.

The structure of GAPD is described in section 6.2. The physical design and implementation of the genetic driver and PRG components of the prototype are described in section 6.3. Section 6.4 describes the transition from the prototype to full GAPD, which is described in section 6.5. The MCDM and GIS components of GAPD are dealt with in section 6.6. Finally, some ideas for improving GAPD are introduced in section 6.7.

GAPD was implemented in stages, first as a prototype with reduced functionality and then as a fully functional system. The step up from the prototype to the fully functional GAPD is large and so during development and test an intermediate step was used to make the transition easier. In the intermediate stage the number of patches is predefined.

Making the transition from the prototype to full GAPD involved changes to the PRG component and the search driver. Different physical designs were used to implement the full GAPD to test it against a range of problems. Objectives and criteria are application specific, so a unique evaluation function is needed for each problem. The PRG component was adapted for different evaluation functions and to operate with different numbers and types of data layers. Some modifications to the PRG component necessitated changes to the search driver. The tests described in chapter 8 include examples of various implementations of GAPD. GAPD was implemented as a number of interchangeable library functions linked into a single module.

6.2 GAPD structure

GAPD couples a genetic algorithm to a raster GIS via PRG. The PRG component grows patches on a raster under the control of a string of numeric parameters. For any given input map, a set of PRG parameters precisely specifies a patch and, consequently, an optimal PRG parameter set corresponds to an optimal patch (Brookes 1997a). It follows that a search for an optimal parameter set is equivalent to a search for an optimal patch. The genetic driver operates on a population of PRG parameter strings and uses PRG to translate strings into patches that are evaluated using raster GIS and MCDM functions.

The logical and physical components of GAPD are illustrated in Figures 6.1 and 6.2. Both diagrams show a cyclical flow through the components. The search and measurement functions operate in different domains using different data structures. The search domain is aspatial and alternative solutions are represented by numeric strings. The problem domain is spatial and alternatives are represented by raster maps. The PRG and MCDM components are the interfaces between the two domains. The cycle is shown in more detail in the flow diagram, Figure 6.3. The genetic driver constructs strings that are passed to the translation function where they are converted to raster maps. GIS functions operate on the maps to measure spatial and aspatial attributes of patches. Attribute values are converted to scores and then to utility values by the MCDM

evaluation functions. Utility values are passed as fitness scores to the search driver. Within the search driver, fitness scores bias the selection of individuals to form a breeding population and genetic operators modify individuals to create a new population. The cycle continues until the algorithm converges - that is until the solution ceases to improve.

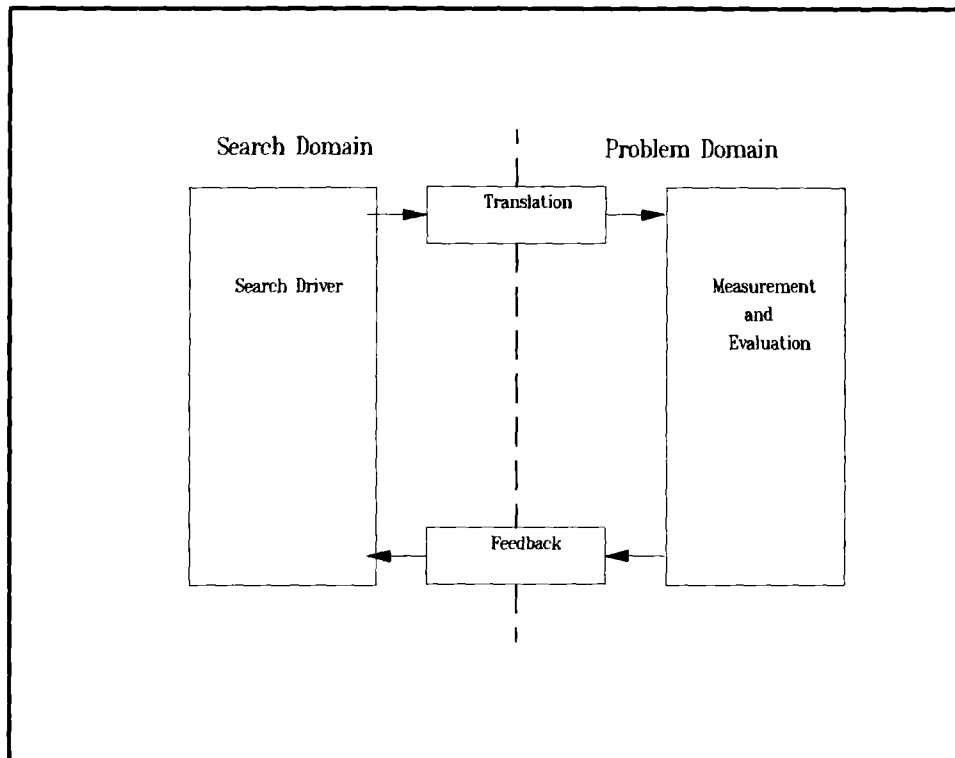


Figure 6.1 Logical structure of GAPD

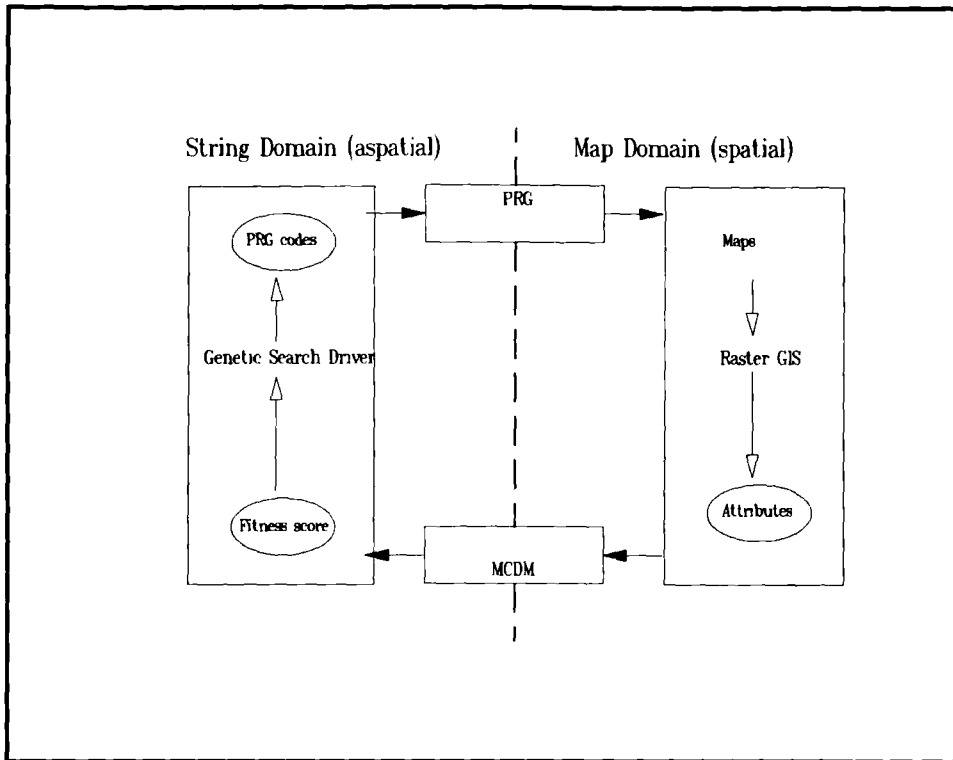


Figure 6.2 Physical structure of GAPD

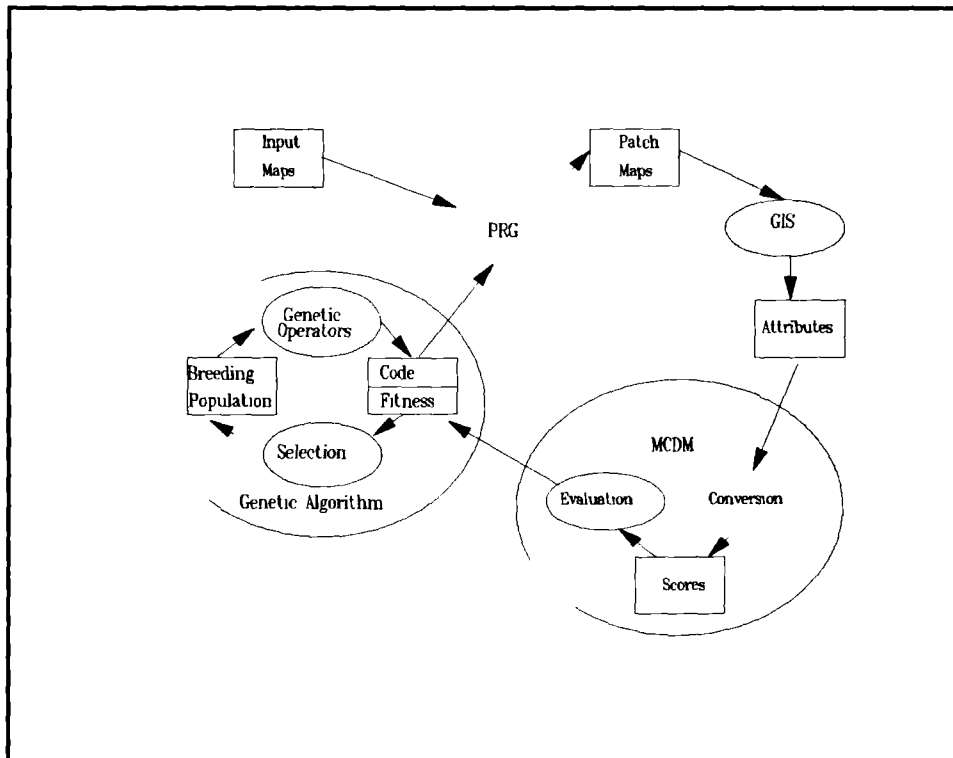


Figure 6.3 GAPD flow diagram

The logical structure of GAPD could be implemented with completely different physical components. Other search techniques could be used instead of a genetic algorithm and there are many options for implementing genetic algorithms as was shown in chapter 5. Each GAPD component was implemented using a variety of physical designs. In theory, the driver and PRG are generic components because they do not need to be changed for each problem, unlike the measurement and evaluation functions which are specific to each application. In practice, different designs were used for pragmatic reasons. One avenue of further research is to investigate to what extent the PRG and driver functions can be made truly generic. It is conceivable that another technique could be devised to translate strings into raster maps. Obviously the criteria, and, consequently, the functions to convert attribute measurements to scores, are application specific. GAPD could also be implemented using different MCDM techniques, including both compensatory and non-compensatory methods. The construction of evaluation functions depends on the MCDM methodology and the relevant criteria.

6.3 The prototype GAPD

6.3.1 Introduction

The prototype contains the essential elements of GAPD, it has the same logical components and the same structure. It is in fact a special case of GAPD but significant effort went into developing GAPD from the prototype. Also the final design of GAPD builds on components developed for the prototype so they are described first. The differences between the prototype and full function GAPD are summarised in table 6.1. This section describes only the generic components, that is the PRG and the driver. Measurement and evaluation are application specific and they are dealt with in a separate section. As previously stated, the key to the successful application of genetic algorithms to decision-making is the genetic code and the mapping from the search domain to the problem domain. In GAPD the genetic code is a set of PRG parameters. Since the form of the genetic code dictates the form of the genetic search driver, the PRG is described

first, followed by the search driver.

| | |
|--------------------------|---|
| Prototype | Full function GAPD |
| Single patch | Number of patches is unspecified |
| Single patch type | Multiple patch types |
| Single objective | Multiple objectives |
| Single suitability layer | Multiple input layers: Suitability, Cost, Raw data |
| Compromise MCDM method | Any MCDM method |

Table 6.1 Comparison of the prototype GAPD and the full function GAPD

6.3.2 The PRG component

The PRG component grows patches with different compositions and configurations determined by a string of numeric parameters called a **patch definition code**, or **PDC**.

Originally the PRG technique was developed as a heuristic for locating optimal sites on raster suitability maps (Brookes 1993, Brookes 1997a). However, it is of limited value in this role because it has no search capability. The key insight for GAPD is that the PRG technique provides a means of precisely encoding a two-dimensional shape as a one-dimensional code. It is this aspect of PRG, its powerful representational capability, which is exploited by GAPD.

As a first step in the development of GAPD, the original PRG algorithm was enhanced to increase its representational power and then tested as a stand-alone program. The program was evaluated using hypothetical problems based on ideas from the conservation biology literature. The results have been published elsewhere (Brookes, 1997a) and are not reported in detail here. The main conclusions were that PRG:

- can be used as an interactive, stand-alone decision-support tool;
- generates feasible patches;
- generates promising patches;
- operation is problematic even in simple problems; and

- operation will become more problematic in more complex problems.

The PRG algorithm generates a single patch, of an exact specified size, with near-optimal spatial characteristics and near-optimal cell suitability. Patches grown by PRG are feasible because they are of the right size and they are promising because they are composed of cells with high intrinsic suitability and because they have good shape characteristics.

PRG differs from other region-growing techniques, such as those used in image processing for the segmentation and classification of images (Sonka *et al.*, 1993), in several ways. In PRG the objective is to grow patches rather than to partition the whole image - this implies that difference criteria, which measure the similarity between cells inside the region and the differences between cells in different regions, are irrelevant. The area surrounding the region can be similar to it because, once the region reaches the required size, it ceases to grow. The homogeneity of the region is not critical because the region will grow across natural edges if that is necessary to attain the desired size and shape. Also the spatial relationships among cells are taken into account as well as the characteristics of the cells themselves.

PRG is a fusion of two heuristics: (1) **simple region-growing (SRG)** which optimises for intrinsic cell suitability while maintaining contiguity; and (2) **parameterised shape-growing (PSG)** which optimises patch suitability by controlling shape. By trading off these two heuristics, PRG incorporates configuration and composition as equally important criteria to optimise patch utility. PRG, PSG and SRG all use different patch definition codes (PDCs).

6.3.2.1 Simple region-growing - SRG

SRG is a bottom-up process that starts with a single cell, **the seed**, and merges neighbouring cells one at a time into the region. SRG operates on a raster data layer in which the value of each cell is a suitability score. The SRG algorithm is an incremental

process which iteratively adds the neighbouring cell with the highest score until the region has grown to the required size. If two or more cells have equal value, then the one closest to the seed is chosen. This tie-break rule imposes a certain degree of compactness on the final shape. SRG will grow a region in which the suitability of the poorest cell is maximised and, given an appropriate seed, SRG will tend to optimise cell suitability.

SRG optimises patch composition only. The spatial characteristics of the regions grown by SRG, i.e. their configuration, will depend entirely on the spatial association between high ranking cells in the suitability map. The PDC for SRG controls the location of the seed (row and column position) and the patch size. The logical and physical structure of the PDC is shown in Figure 6.4.

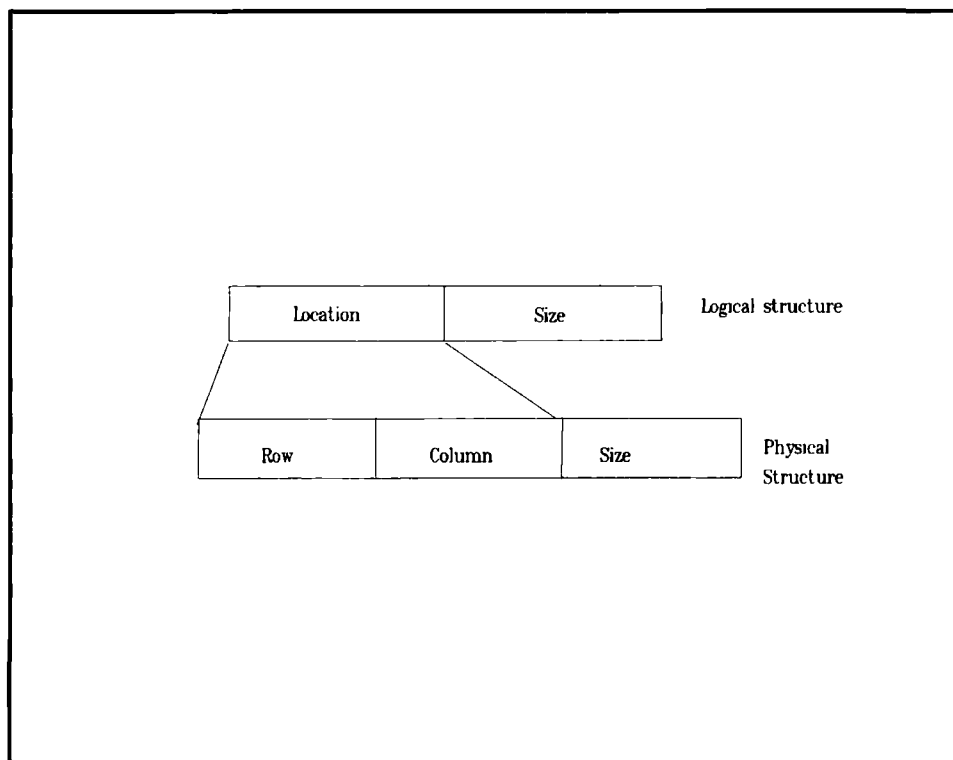


Figure 6.4 SRG parameter structure (PDC)

6.3.2.2 Parameterised shape-growing - PSG

The PSG concept is that a region-growing program that generates particular shapes can be built by substituting **shape scoring rules** for the cell scoring rules in SRG. PSG uses the same incremental process as SRG but with a different scoring algorithm. At each iteration, the cell with the highest shape score is added to the region. Shape score is based on an anisotropic distance decay function. This means that shape score decreases with distance from the seed at a different rate in different directions. In the special case when the score is an isotropic function of distance, scores decrease at the same rate in all directions and PSG grows compact circular shapes. By using different anisotropic functions PSG can generate many different shapes. Anisotropic spread is a familiar concept in diffusion, for example in fire spread modelling (Jianping & Lathrop, 1995).

The PDC for PSG includes the same three parameters as SRG (row, column and size) plus a string of **shape-control parameters**, called a **shape definition code (SDC)**, that control the perimeter outline and orientation of the final shape. The logical and physical structure of the PDC is shown in Figure 6.5. By controlling three basic shape properties it is possible to generate a large variety of shapes. The operation of PSG can best be understood by considering regular shapes although not all shapes generated by PSG are regular.

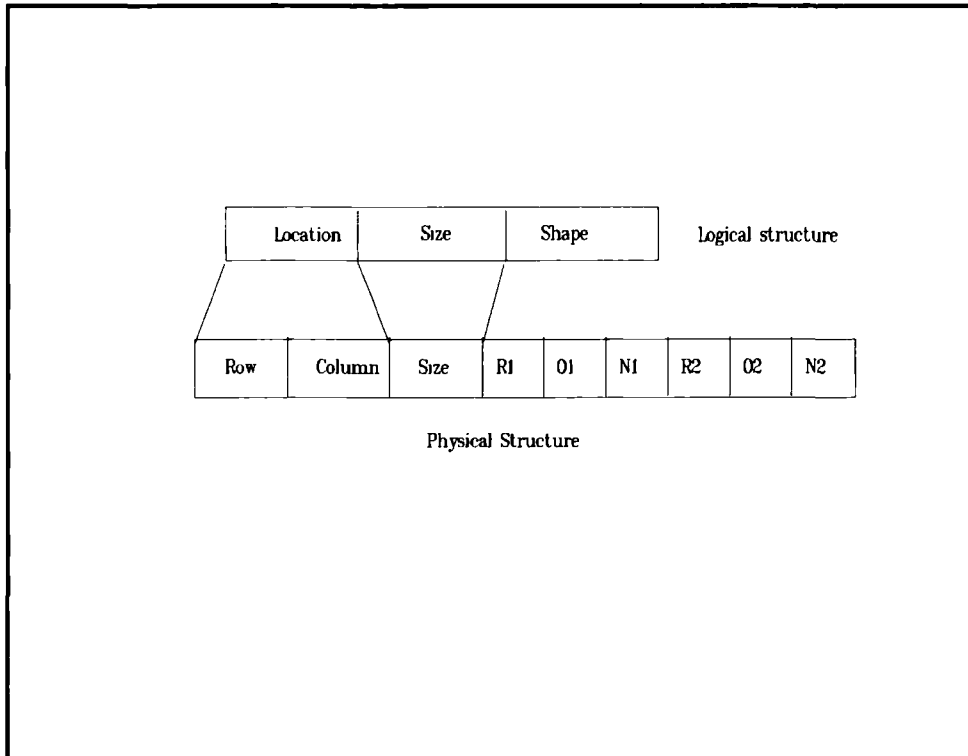


Figure 6.5 PSG parameter structure (PDC)

The three shape properties controlled by PSG can be expressed in terms of **major** and **minor axes**. An axis is a straight line joining the centroid to a point on the perimeter. The longest axes are major axes and the shortest axes are minor axes. In general shapes will have several major and minor axes.

Using the above concepts the three shape properties are:

ratio: the ratio of the lengths of the minor to major axes

orientation: the smallest positive angle between a major axis and the rows of the raster

number: the number of major axes.

The original idea behind PSG was that the perimeter of a shape is a line and can, therefore, be defined by trigonometric functions as in Fourier analysis. A single function, say a cosine, will produce a regular shape with a wavy perimeter. The sinusoidal line is

superimposed on a circle so that the radius of the circle and the amplitude of the wave control the ratio of minor and major axes. Variations in three wave properties, amplitude, frequency and phase, will result in variations in the complexity of the boundary. There is a direct correspondence between the three wave properties and the three shape properties:

amplitude <-> ratio

phase <-> orientation

frequency <-> number

The ideas behind PSG are illustrated in Figure 6.6. The shape scoring component of the original PSG was based on a cosine function but there are other possibilities.

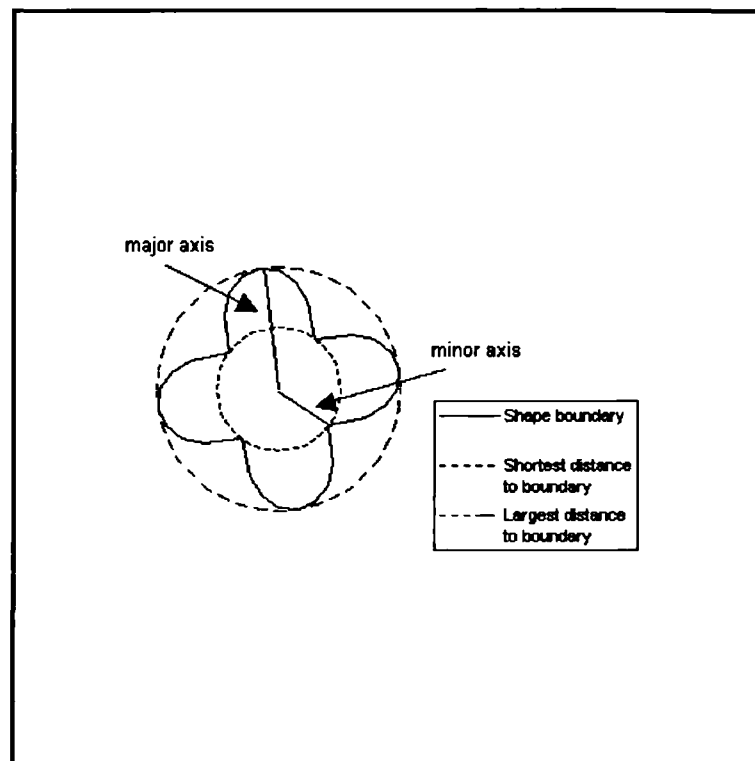


Figure 6.6 Sinusoidal shape with 4 axes and a ratio of major to minor axes of 2:1

In the initial development of GAPD, the original PRG algorithm (from the M.Sc. project) was enhanced by introducing a new SDC code and improving the shape scoring algorithm in PSG. A more complex SDC allows PSG to grow more complex shapes. A

single SDC generates a primitive shape and is called a **primitive SDC**. A **complex SDC** is a concatenation of primitive SDCs. The GAPD implementation of PRG uses a concatenation of two primitive SDCs called a **dual SDC**. The shape scoring algorithms for the single SDC and the complex SDC are described in detail below.

Original PSG cell scoring algorithm for primitive SDC

The original shape scoring function uses four integer shape control parameters: the **numerator**, N , the **denominator**, D , the **orientation**, O and the **number of axes**, A . The shape score for each cell is calculated from these parameters and two measurements on the raster: the distance, d , and direction, θ , from the seed. Direction is measured with respect to the x axis (see figure 6.6a). The parameters N and D together control the ratio of the major and minor axes, A defines the number of axes, and O defines the direction of the major axis. The shape score, S , is a modified distance and it is calculated in two steps. The **distance modifier**, M is calculated by equation 6.1 and S is given by equation 6.2.

$$M = \frac{N + (D - N) \cos\left(\left(\theta * \frac{A}{2}\right) + O\right)}{D} \quad \text{Equation 6.1}$$

$$S = M * d \quad \text{Equation 6.2}$$

The cosine term varies between 0 and 1 so M varies between N/D and 1. The cell with the lowest score, S , is added at each iteration.

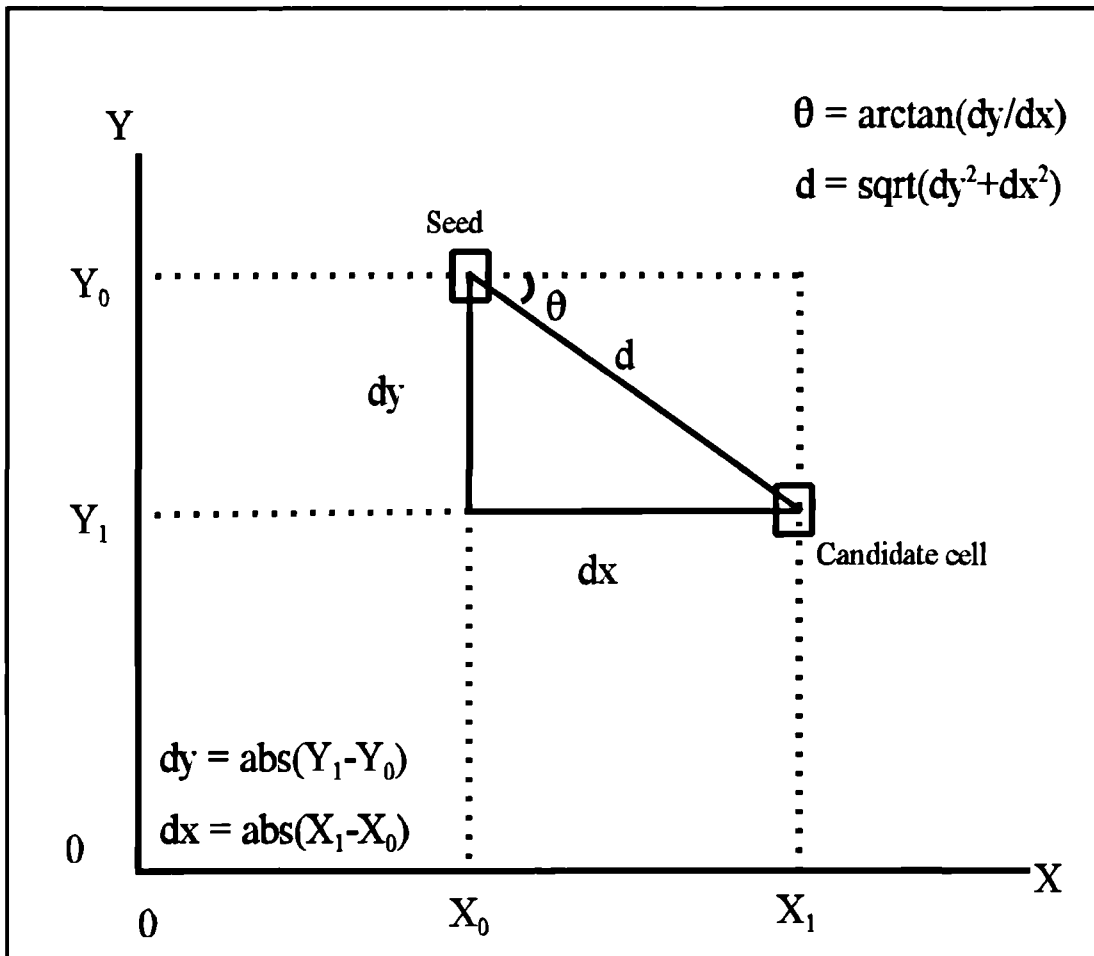


Figure 6.6a Diagram showing the meanings of the measurements d and θ in equations 6.1 and 6.2 and how they are calculated.

New PSG cell scoring algorithm for primitive SDC

The second version of the cell scoring algorithm has a number of improvements over the original. The main change is to allow non-integer parameter values. In the original version, the shapes are all symmetrical because the number of axes, A , is an integer. With the new version A can be less than 1 with the result that shapes need not be symmetrical. The two integers N and D are replaced by a single non-integer ratio parameter, R . The new calculation of M is given by equation 6.3.

$$M = R + [(1 - R) * \cos((\theta * A) + O)] \quad \text{Equation 6.3}$$

R varies between 0 and 1, the cosine term varies between 0 and 1 and the modifier, M , varies between R and 1. The shape suitability, S , of each cell is again given by equation 6.2.

Cell scoring algorithm for complex SDC

The range and complexity of possible shapes can be further increased by generalising PSG to use complex SDCs. A shape score is calculated for each primitive SDC using equation 6.2. The scores for each primitive SDC can be combined in various ways, for example by taking the average, the sum, the minimum or the maximum of the scores. The average score gives more variety than the maximum or minimum. If S_i is the score using shape control parameters R , A , and O , the resultant score, S , is given by equation 6.4.

$$S = \frac{\sum_{i=1}^n S_i}{n} \quad \text{Equation 6.4}$$

Some sample shapes grown by PSG (with dual SDCs) and the shape control parameters used to generate them, are shown in Figure 6.7 and Table 6.2.

| Shape | Shape control parameters | | | | | |
|---------------|--------------------------|----------|----|----|----|----|
| | R1 | A1 | O1 | R2 | A2 | O2 |
| Top left | 1 | 0 | 45 | 0 | 0 | 45 |
| Top centre | 0.5 | 1 | 0 | 0 | 0 | 0 |
| Top right | 0.5 | 1 | 45 | 0 | 0 | 0 |
| Middle left | 0.3 | 2 | 0 | 0 | 0 | 0 |
| Middle centre | 0.3 | 2 | 0 | 3 | 2 | 45 |
| Middle right | 0.3 | 2 | 0 | 1 | 2 | 0 |
| Bottom left | 0.3 | 0.5 | 0 | 0 | 0 | 0 |
| Bottom centre | 1 | 0.2 5 | 0 | 0 | 0 | 0 |
| Bottom right | 0 | 1.5 | 0 | 0 | 2 | 45 |

Table 6.2 Shape definition codes for figure 6.7

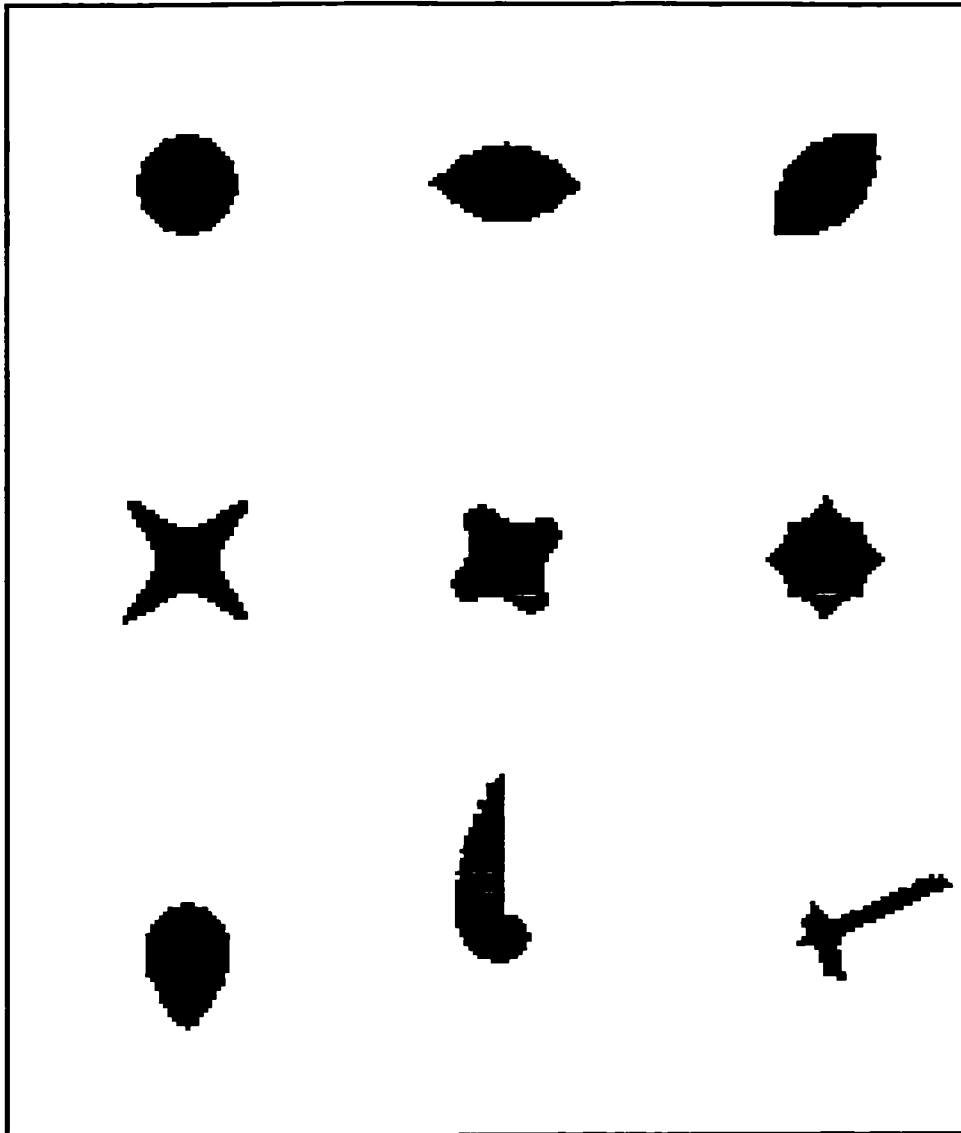


Figure 6.7 Shapes grown by PSG with dual shape definition code (SDCs are listed in table 6.2)

6.3.2.3 PRG - the fusion of SRG and PSG

PRG grows patches which tend towards a particular shape and at the same time, include high-value cells by fusing the ideas of PSG and SRG. PRG calculates a **resultant patch score** for each cell by combining the shape score from PSG and the cell score from SRG. The resultant patch score is a compromise between the two scores. The trade-off between shape score and cell score is achieved by taking a weighted average of the two

scores. The resultant score for a cell i , U_i , is given by equation 6.5 where C_i is the intrinsic cell score, S_i is the shape score and W is the relative shape weight or **shape bias** ($0 \leq W \leq 1$).

$$U_i = W.S_i + (1-W).C_i \quad \text{Equation 6.5}$$

In the original PRG program the shape bias was fixed at 0.5. In the version used in GAPD, the shape bias is variable. How far the actual patch deviates from the ideal shape that would be grown by PSG depends on the shape bias parameter and the underlying suitability raster. A bias of 1 equates to pure PSG and a bias of 0 equates to pure SRG. The PDC for PRG is the same as that for PSG with the addition of the shape bias parameter. The structure is illustrated in Figure 6.8.

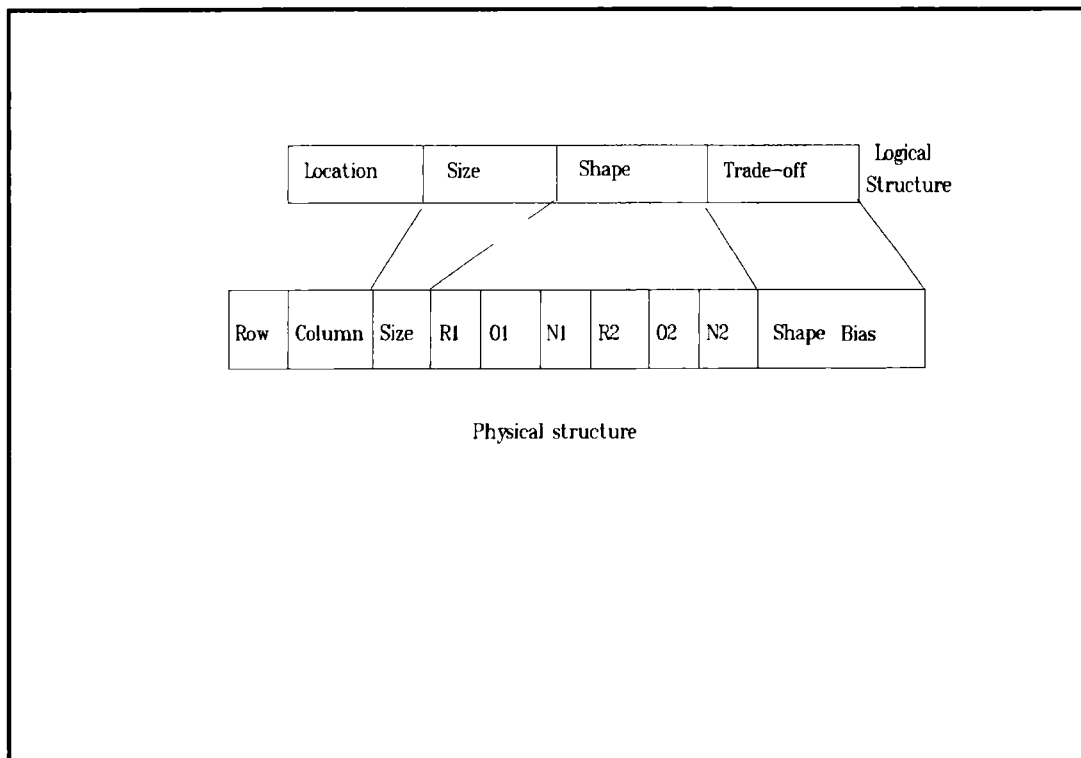


Figure 6.8 PRG parameter structure (PDC)

The shape score in PSG must be adjusted before PSG and SRG can be combined. PSG selects the lowest-scoring cell but SRG selects the highest-scoring cell. The PSG scale could be reversed by using inverse distance but this would not solve all the problems.

Absolute PSG scores would decrease with distance, without limit, so that as distance increased, SRG scores would get relatively larger and SRG would dominate irrespective of the shape bias. This problem with combining disparate entities is a typical MCDM situation and the answer is to normalise the scores. The twist is that the normalisation process must adjust as the patch grows. This is achieved by using a relative distance instead of an absolute distance.

The relative distance is derived with reference to the enclosing rectangle. The **enclosing rectangle** is defined by the points $\{ X_{max}, Y_{max} \}, \{ X_{max}, Y_{min} \}, \{ X_{min}, Y_{min} \}$ and $\{ X_{min}, Y_{max} \}$ where $X_{max}, X_{min}, Y_{max}, Y_{min}$ are the maximum and minimum x and y coordinates of all the cells surrounding the patch edge. As the patch grows, the enclosing rectangle also grows. No candidate cell can be further from the seed than the furthest corner of the enclosing rectangle. The distance from the seed to the furthest corner, *diag*, is used as a scaling factor.

The shape score is normalised as follows. The un-normalised PSG score, S , is calculated as before using equation 6.2 or 6.4. This value can be interpreted as a modified distance: it cannot be greater than the diagonal because the modifier (equation 6.3) is not greater than 1. The normalised score, S_N , is then calculated using equation 6.6. Normalisation converts PSG scores to a scale between 0 and 1, where 1 is the best and 0 the worst. Further conversion to any appropriate linear scale is then trivial.

$$S_N = \frac{diag - S}{diag} \quad \text{Equation 6.6}$$

There are two potential problems with patches grown by PRG. One is holes and the other is barriers. PRG uses a hole stopping routine that vetoes any cell whose inclusion could lead to a hole being created during subsequent iterations. Details of the hole

stopping routine can be found in a paper by the author (Brookes, 1997a), a copy of which is bound at the back of the thesis. As a measure to prevent the region from growing across narrow linear barriers without cost, the PRG algorithm only considers the 4-neighbour cells for inclusion. Thus, PRG can cross barriers but only by including barrier cells in the patch.

6.3.2.4 Summary

PRG is the key component of GAPD. The PDC is the basis of the genetic code and as such dictates the form of the genetic driver, The parameters in the PDC can be grouped into three logical categories:

- (1) the **control parameters** which specify the region's size and location;
- (2) the **shape parameters** which guide the region towards an ideal shape;
- (3) the **shape bias parameter** which determines the relative influence of shape suitability and intrinsic (static) cell suitability.

6.3.3 The prototype genetic search driver

The GAPD prototype applies to single patch problems and uses one PDC as the genetic code. Full function GAPD determines the number of patches, their types, size, shape and distribution and, consequently, it uses a more complex structure but the basic unit is the same.

6.3.3.1 Physical components

The genetic code

All implementations of GAPD in this project use a dual SDC in the PDC and the new PSG shape scoring algorithm. Altogether, there are ten PRG parameters made up of three control parameters, six shape control parameters (in two sets of three) and the shape bias parameter. The genetic code is illustrated in figure 6.9.

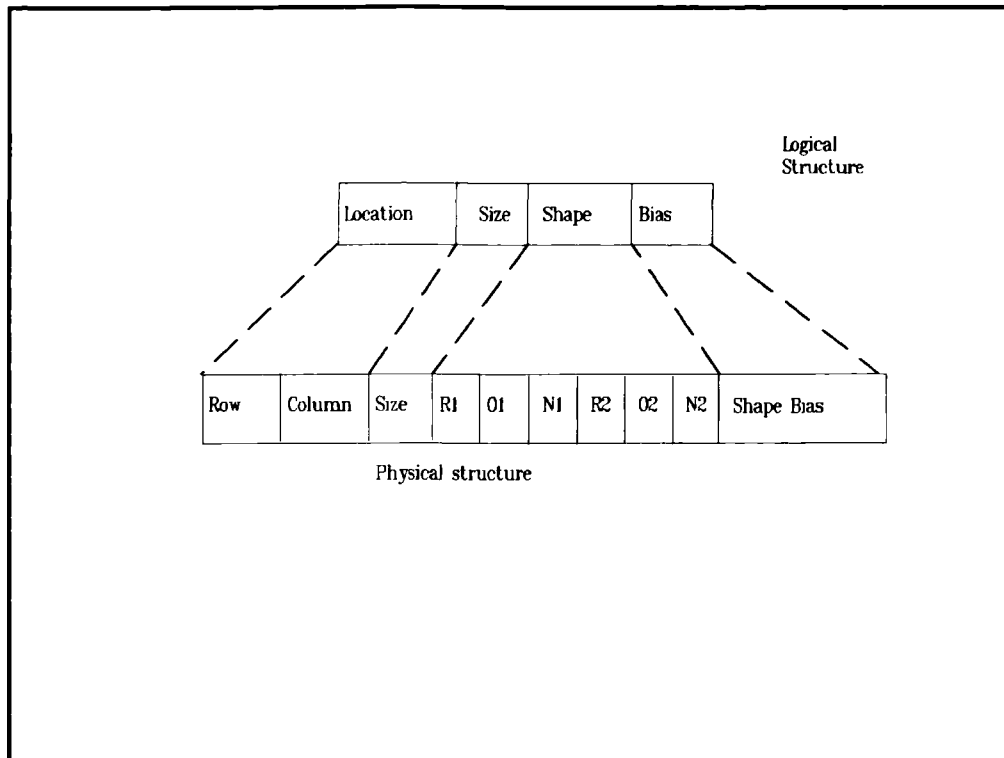


Figure 6.9 Genetic code for the GAPD prototype

The PDC is a string of numbers and there are alternative ways of incorporating such a structure into a genetic algorithm. The choice is between using binary strings, as in Holland's canonical genetic algorithm (Holland 1975), or using numeric strings. Holland's schema theorem (Holland, 1975) says that binary strings are better but Davis has argued that this is not necessarily so (Davis, 1991). Such theoretical considerations are beyond the scope of this research. The numeric string implementation was chosen for pragmatic reasons. A disadvantage of binary strings is that they would involve extra functions to map back to numeric parameter values. An advantage is that no new genetic operators are needed. For both binary and numeric strings there is the possibility that genetic operations such as crossover and mutation will sometimes generate invalid parameter sets. The numeric string option is a more obvious mapping to the underlying PRG representation and numeric genetic operators are quite easy to implement. A further, serendipitous, advantage of real number strings is that they make debugging the code easier because it is possible to view genetic codes while GAPD is running using a debugging tool and read the actual values that are passed to PRG directly.

PRG parameters can be integers or floating point. In the genetic code all numbers are restricted to integers to make the genetic operators simpler. Integers are converted to real numbers by dividing by 10. Each number has a **domain** which defines the valid values. Domains are expressed as a tuple specifying the type, the minimum and maximum values, and the increment. For example the domain of the orientation parameter is {int,0,180,1} meaning that the value is any integer from 0 to 180. If the domain were {int,0,180,2) only even integers would be legal. Other domains are shown in Table 6.3.

| Genetic code for a single individual | | | | | | | | | |
|--|--------|------|--------------------------|----|-----|----|----|-----|------|
| Control parameters | | | Shape control parameters | | | | | | Bias |
| row | column | size | R1 | A1 | O1 | R2 | A2 | O2 | |
| Parameter domains (minimum and maximum values) | | | | | | | | | |
| 5 | 5 | 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 75 | 75 | 150 | 1 | 10 | 180 | 1 | 10 | 180 | 1 |

Table 6.3. Genetic code structure and sample parameter domains: the values for size, row and column depend on the problem and the dimensions of the raster

The genetic operators

GAPD uses five genetic operators. **Crossover** and **mutation** are numeric equivalents of the binary string operators of Holland's original genetic algorithm (Holland 1975). **Sum**, **average** and **creep** are three new arithmetic operators, developed as part of GAPD, specifically for numeric strings. Crossover exchanges one or more numbers between strings but does not generate any new values. The other operators do generate new numbers but change only one number in a string. Crossover, sum and average operate on two strings while mutation and creep operate on a single string. The genetic operators are described more fully in figures 6.10 and 6.11. Davis (1991) suggests a number of operators for numeric genetic codes.

Crossover creates two offspring from two parents by chopping the parents at a random point and swapping the tails. There are 10 numbers so there are 9 possible places to cut the string and 9 possible tails.

parent1 = 12,36,55,1,12,76,8,4,108,7

parent2 = 64,23,107,5,2,45,3,6,25,2

chop tails at position 4

child1 = 12,36,55,1,2,45,3,6,25,2

child2 = 64,23,107,5,12,76,8,4,108,7

Mutation creates one offspring from a parent by copying the parent with one number modified. The number is chosen randomly and the new value is randomly chosen from the number's domain.

parent = 12,36,55,1,12,76,8,4,108,7

mutate position 5

child = 12,36,55,1,74,76,8,4,108,7

Creep is similar to mutation but the modification is limited to plus or minus 10% of the domain of the selected number.

parent = 12,36,55,1,12,76,8,4,108,7

mutate position 5

child = 12,36,55,1,13,76,8,4,108,7

Figure 6.10 Genetic operators (a): crossover, mutation and creep, from Brookes 1997b

Average creates two offspring from two parents with one number changed. In each offspring the new number is a different weighted average of the parent's numbers so for the i th number the new numbers C_{1i} and C_{2i} are given by equations 6.11.1 and 6.11.2.

$$C_{1i} = \frac{(2 * P_{1i}) + P_{2i}}{3} \quad \text{Equation 6.11.1}$$

$$C_{2i} = \frac{P_{1i} + (2 * P_{2i})}{3} \quad \text{Equation 6.11.2}$$

parent1 = 12,36,55,1,12,76,8,4,108,7
 parent2 = 64,23,107,5,2,45,3,6,25,2
 average at position 7
 child1 = 12,36,55,1,12,76,6,4,108,7
 child2 = 64,23,107,5,2,45,5,6,25,2

Sum is similar to average but the new numbers are the sum and difference of the original numbers.

parent1 = 12,36,55,1,12,76,8,4,108,7
 parent2 = 64,23,107,5,2,45,3,6,25,2
 sum at position 9
 child1 = 12,36,55,1,12,76,8,4,133,7
 child2 = 64,23,107,5,2,45,3,6,83,2

Figure 6.11 Genetic Operators (b): average and sum, from Brookes 1997b

Every genetic operation honours the parameter domains and always generates a valid string. Each domain is a torus. When a value exceeds the maximum it loops around to the minimum, like the hands on a clock, and vice versa. For example an orientation of 210 translates to 30, and an orientation of -20 translates to 160. Genetic operators act at a single point in the code. This point is chosen randomly with equal probability at all points.

Selection and population generation

Roulette selection is used to select individuals from the old population. Individuals are

randomly selected with a probability proportional to their utility. When an individual is selected a genetic operator is chosen randomly. If no operator is selected, the individual is copied unchanged. If crossover, sum or average is chosen, then another individual is selected to be the second parent. The individual with the highest utility in each generation is automatically selected and copied unchanged to the next generation. This is done because the objective is to find the best individual and otherwise it would be possible to lose the best individual through stochastic events.

Stopping condition

The GAPD stops when either a specified number of generations has passed without any improvement in average utility or the total number of generations allowed has been exceeded.

6.3.3.2 Operational context

The GAPD program code is based on the simple genetic algorithm code published by Ribeiro Filho *et al.* (1994) with the modifications outlined above. Links to PRG and other routines are hard coded. This means that GAPD must be recompiled for different applications otherwise the program size gets too large. More efficient coding techniques would overcome these problems. GAPD does, however, have a lot of flexibility in other respects. This flexibility is achieved by using a control file. The control file allows the genetic algorithm to be tuned to a particular problem without changing the program code. Larger and more complex search spaces require larger values for population size.

Different operator probabilities can give better results in different problems. The population size, the number of generations, the stopping criteria and the genetic operator probabilities are all read from the control file. The use of a control file proved to be very beneficial when implementing GAPD.

6.4 The transition from the prototype to GAPD

6.4.1 Introduction

The extension of the prototype to a fully functional GAPD was difficult to accomplish. A number of alternative designs were considered and experimented with. Initially, efforts were concentrated on modifications to the genetic driver and the genetic code using ideas published in the genetic algorithm literature while the PRG algorithm and the PDC were left alone but these efforts were not successful. A second line of investigation concentrated on the PRG component. This proved more fruitful and the final design for GAPD incorporates an enhanced multi-patch version of PRG.

The most significant difference between the prototype and GAPD is that GAPD addresses multiple patch problems. The major difficulty which arose in multi-patch problems was the control of total patch size. For single patch problems, the PDC size parameter matches the constraint but for multiple patches the constraint is on the sum of all patch sizes. Several alternative designs exhibited an inherent bias against multi-patch solutions. This bias was overcome by modifying the PRG algorithm.

6.4.2 Possible alternative drivers

The first idea for extending the prototype to full functional GAPD was to use an unmodified PDC as the basic genetic code or genotype. Each PDC would correspond to a single patch. A multi-patch network would be specified by multiple codes and generated by multiple calls to PRG. This scheme could be implemented by incorporating individual PDCs into higher level structures in a number of ways. There are many examples of genetic algorithm applications using novel structures (Bhandarkar *et al.*, 1994; Buckley & Hayashi, 1994; D'Angelo *et al.*, 1995; Hobbs, 1994; Potts *et al.*, 1994; Roth & Levine, 1994; Srikanth *et al.*, 1995).

6.4.2.1 Variable length string

A variable length string is a simple and obvious structure. Individual PRG codes are concatenated together. This method was implemented and some initial trials carried out. The number of patches was controlled by allowing the size domain to have a zero minimum value, thus allowing for the possibility that some seeds would not grow. Trials of this version of GAPD gave poor results in multi-patch problems because of the size problem described above.

6.4.2.2 Hierarchic structure

An attempt was made to overcome the bias against multi-patch solutions by introducing a size budget. The size budget field stores the current total of all patch sizes. Another new field, the number of patches field, stores the number of patches. The genetic code now has a two tier structure. PRG parameters at the bottom level are grouped into PDCs that are elements in the top level.

Three new genetic operators were developed to act on the top level structure. Whole sub-strings can be inserted, deleted and swapped between individuals. When a sub-string is inserted or deleted the number of strings field is updated. Parameters within individual PRG codes are subject to the same operators as used in the prototype GAPD. The size budget and number of patches fields are updated after every operation.

Values are assigned to the size parameters of individual PRG strings in a new way. The size domain is dynamic as it now relates to the size budget field not to individual patch sizes. Every time a patch-size parameter is changed the size budget and the size domain are updated. If a new patch-size parameter exceeds the maximum value in the domain, it is adjusted in the usual way.

Trials of this version of GAPD gave poor results in multi-patch problems. The use of a size budget meant that size domain did not apply equally to all patches. As the size budget was used up the size domain shrank so that the genetic algorithm could not vary the size parameter of existing patches or add new patches. The freedom of the genetic

algorithm to vary patch sizes and number of patches was restricted.

6.4.2.3 Other possibilities

A number of other options were considered but not implemented. Genetic programming uses a tree structure as the basic genetic code (Koza, 1992). Trees are an efficient structure for variable length codes and genetic operators for trees are well documented. Some genetic algorithms use meta-populations. GAPD could use two populations. In the first, the individuals would be PRG codes as in the prototype. In the second, individuals would be strings of pointers to individuals in the first population. In this way, each patch could be a part of many alternative networks. Another option is to use a spatial structure. Sub-populations could be confined to segments of the map and selection and breeding operators could be spatially dependent. All methods based on single-patch PRG structures face the same problem of controlling total patch size.

6.4.2.4 Why this line was abandoned

This line of development was abandoned in favour of an approach based on modifying the PRG. The modified PRG approach produced better solutions and avoided the bias against multi-patch solutions. Although it would have been interesting to pursue the other investigations further that would not necessarily meet the primary objective of implementing a working system. Another reason was that additional complexity was being introduced that detracted from the aim of making GAPD simple, robust and general purpose. The simpler the mechanism the more likely it is to be applicable to a range of problems. At some point, the problems associated with a high degree of tuning of the genetic algorithm become no different to those of designing ingenious, problem-specific heuristics.

The problems with using an unmodified PRG algorithm stemmed from an inherent bias against multi-patch solutions. This is a clear example of an inappropriate genetic code preventing the genetic algorithm from finding good solutions. The key insight was the

recognition that the bias resulted from the use of a size parameter for each patch. Total size of all patches is not controlled by a single parameter in a PDC. The number of patches and the size of each patch in the optimal design is unknown, therefore, the maximum of the size domain for each patch must equal the size constraint to allow for the solution to be a single patch. For the same reason, the minimum size must be 0 so that no other patches will grow. Consequently, the average patch size will be one half the size constraint and the average number of patches in a valid solution will be two. Thus, there is an inherent bias in the data structure against multi-patch solutions because they tend to exceed the size constraint and consequently been penalised with lower utility. The solution was to reinterpret the size parameter in the PDC as relative not absolute size and then to add a new global parameter to control the size of all patches. This solution was implemented in a straightforward way by amending the PRG component as described below.

6.5 The full function GAPD

The successful implementation of the full function GAPD was achieved by modifying the PRG component. Only minor modifications were needed to the genetic search driver component.

6.5.1 Modified PRG

A multi-patch PRG algorithm with a new parameter structure was devised. Multi-patch PRG grows patch networks from a single parameter string. The PDC for multi-patch PRG is basically a concatenation of PDCs for single patch PRG. The new PDC is used as the genetic code by GAPD.

The **multi-patch PDC (MPDC)** has a header and a tail. Information that applies to all patches is stored in the header and the information that applies to individual patches is stored in the tail. The tail is a concatenation of single patch PDCs with some modifications to the parameters and their meaning. What was formerly the size

parameter becomes a relative size. Total size is stored in a new parameter in the header. Denoting total size by T , relative size by R_i , and the number of patches by N , the absolute size, S_i , of patch i is given by equation 6.7.

$$S_i = T * \left(\frac{R_i}{\sum_{i=1}^N R_i} \right) \quad \text{Equation 6.7}$$

GAPD solves problems where the solution is a network of patches of different types. Type applies to each patch independently and therefore a type parameter is included in each PDC in the tail. The full genetic code for GAPD is shown in figure 6.12.

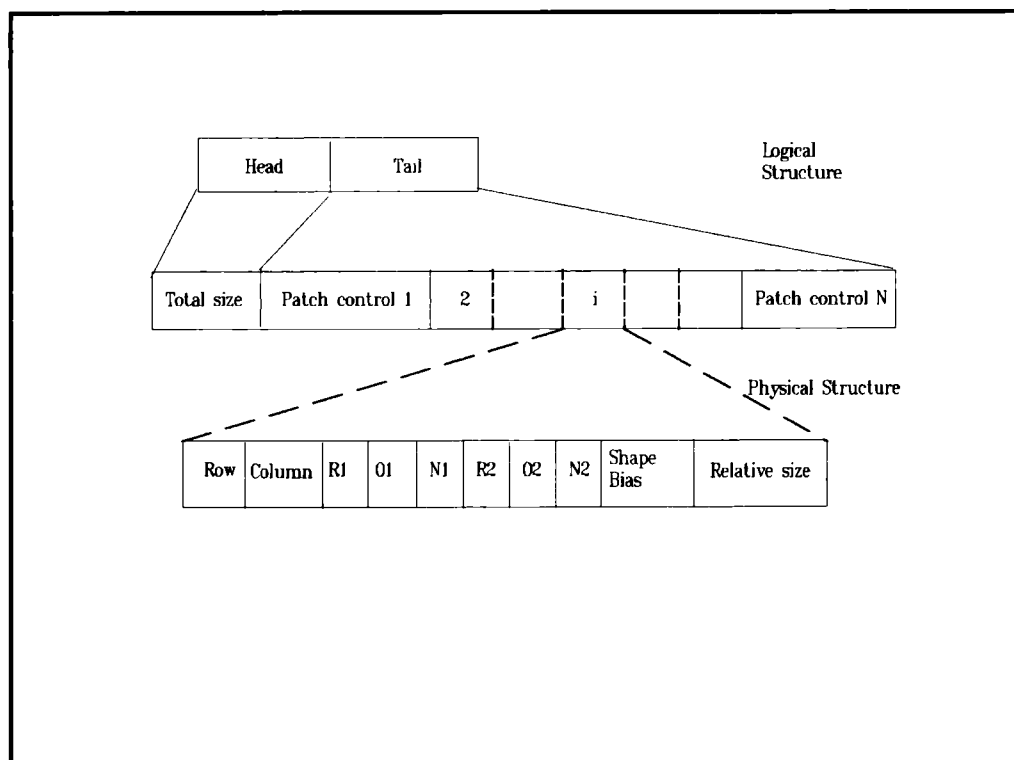


Figure 6.12 Genetic code for the full function GAPD

Ideally, GAPD should be implemented with a variable length genetic code so that the number of patches can vary without limit, but in practice a fixed-length code can be used.

A fixed-length code limits the number of patches but this is not a problem if there is actually an upper limit in the real world. This is the case if there is both a total-size constraint and a minimum-size constraint for patches. Then the maximum number of patches can be computed exactly by dividing the total size by the minimum patch size. Even if there are no exact constraints, similar considerations can be used to estimate an upper limit that is unlikely to be exceeded by a good solution. The length of the genetic code then fixes the maximum number of patches. Up to this limit the actual number of patches is variable if patch relative size can be zero. A fixed structure uses more computer memory than a variable length structure but is simpler to implement.

How multi-patch PRG works

Multi-patch PRG works in basically the same way as single-patch PRG. Each iteration adds a single cell to the patch network but this time there are multiple seeds, each of which can grow into a patch. A cell can be a candidate for inclusion in more than one patch.

The method of scoring cells is similar to single-patch PRG. Each cell has a patch score for any patch that it can join. Shape scores are normalised using the diagonals of the patches concerned. Patch scores are adjusted using the absolute patch size, computed using equation 6.7, as a weighting factor. In each iteration, if the current size of the patch is less than its absolute size the shape score stands but if it exceeds the absolute size a penalty function is imposed to reduce the patch score. The penalty function can be adjusted to allow patches to exceed the absolute size by a greater or lesser amount. A cell can only be added to one patch and, once added, it cannot be removed. All patch growth terminates when the total size is reached.

Each patch is identified by the position of its PDC in the MPDC. PRG first grows patches with unique identifiers. When patch growth is complete the patch type is substituted for the identifier. Where patches of the same type are contiguous they merge into a single patch.

6.5.2 The GAPD driver

The integration of multi-patch PRG into GAPD requires only minor changes to the genetic driver. No entirely new genetic operators are needed. The parameter domains and other control variables are read in from the control file as before.

Extra processing is needed so that the domains are applied correctly. There are fewer domains than there are parameters in the genetic code because parameters in the different PDCs share the same domain. When a genetic operator is selected to act at a randomly chosen point, P , in the genetic code, it is no longer obvious which domain applies. If H = the length of the header, T = the length of the tail, and N = the number of patch codes, then the total length of the genetic code is $H+(T*N)$, the number of domains is $H+T$ and the program calculates which domain, D , applies using equation 6.8 or 6.9. Note that the modulus % operator yields the remainder from a division.

$$D=P \quad \text{if } P \leq H+T \quad \text{Equation 6.8}$$

$$D=H+(P-H)\%T \quad \text{if } P > H+T \quad \text{Equation 6.9}$$

The same kind of processing is used to apply the crossover operator. If the crossover point, P , is in the tail, then the identifier, I , of the PDC in which P falls is found by equation 6.10.

$$I=(P-H)/T \quad \text{Equation 6.10}$$

The parameters in this PDC are swapped individually. All other PDCs and the header are swapped as single units. The final number of patches can be less than the number of

PDCs, for example, if some patches have zero relative size or if patches merge together.

6.5.3 Pragmatic variations in the implementation of GAPD

For pragmatic reasons it can be beneficial to use different physical structures for the PDC. When computer resources are limited, it may be worth using shorter genetic codes because they reduce the search space and, therefore, reduce execution time. There is a risk that the quality of solutions will be compromised because the representational power of PRG is reduced. If, however, the search space is too large and the population size too small, then the genetic algorithm is likely to converge on a sub-optimal solution in any case. Some of the tests of GAPD used shortened PRG codes. The tests are described in chapter 8 and it is made clear when and why different codes were used.

6.6 The measurement and evaluation functions

The spatial factors summarised in chapter 3 can be used to build models for evaluating individual patches and networks of patches. Key attributes influenced by size, shape and pattern are:

- total patch size,
- patch core size,
- patch composition,
- patch core composition,
- heterogeneity of physical attributes,
- connectivity between patches.

These attributes can be measured using simple raster GIS functions. Converting the measurements to utility scores is then procedurally straightforward, as explained in chapter 2. Shape was never explicitly measured in any of the evaluation functions used to test GAPD but the influence of shape is implicit. This both avoids the issue of measuring shape and makes use of more meaningful concepts.

Simple metrics of area, distance and composition can be used as the basis of evaluation

functions. These simple metrics are made using primitive raster GIS functions. In real situations, meaningful attributes in the problem domain are more complex but they can be measured by combining basic metrics. Attribute measurements are converted to utility scores using evaluation functions.

The functions described below were coded as routines in a program library and linked to GAPD. This was done because it was easier to program them from scratch than to link modules from GIS systems into GAPD. The functions are not generic parts of GAPD but they are included here as examples of how GAPD works and because they were the ones actually used to test GAPD.

6.6.1 Basic raster functions

The **extract** routines are a series of functions that return summary values of patch composition. **Extract_count** operates on a single input raster and returns a count of all cells which have a particular value. **Extract_min** operates on two rasters and returns a single number. The first raster defines the patch and the second is a value layer. The function overlays the patch on the value layer and extracts the minimum value. **Extract_sum** is similar to **extract_min** but in this case it returns the sum of cell values.

Shrink removes the boundary cells from all patches in the input raster and generates a new output raster. Boundary cells are those which have at least one neighbour with a different value. Shrink can cause a patch to split into fragments or to disappear altogether. The shrink function interprets neighbours as meaning cells which share an edge (i.e. it checks 4 cell neighbourhoods). Multiple calls to shrink will strip off multiple layers of cells.

Spread generates a raster in which each cell value is the distance from the nearest patch. Patches are defined by the input layer and distance values are geometric distances. **Cost_spread** is like spread but it uses an additional cost layer. The cost layer is a friction surface in which the value in each cell is the cost of travelling through that cell.

The output raster is a `cost_distance` raster in which the value in each cell is the minimum travel distance to that cell from any patch. This is the most complex of the functions and it uses a published algorithm (Jianping & Lathrop, 1995), the others are trivial.

Patch_count counts the number of distinct patches and assigns a unique identifier to each. Two contiguous patches of the same type are merged into one patch. Output is an integer count and a new raster.

6.6.2 Attribute measurements

Patch area, measured in number of cells, is found using the function `extract_count`. `Extract_count` returns the total area of all patches of the same type. To get the area of a single patch the patch must have a unique identifier. A unique identifier can be assigned by using `patch_count`.

Pattern can be interpreted in many ways. Some of the simplest use inter-patch distance. Inter-patch distances are found in two stages. `Spread` or `cost_spread` generates a distance layer from the source patch. `Extract_min` operating on the sink patch and the distance layer returns the minimum edge-to-edge distance between the source and the sink patches. Minimum cumulative resistance (MCR) is one ecologically significant measure of the distance between patches (Knaapen *et al.*, 1992). MCR is measured using `cost_spread` and `extract_min`.

Composition has many interpretations. **Cumulative suitability** was frequently used for testing GAPD. It is found using `extract_sum`. One test used a different interpretation. Composition was measured using the total area of each different landuse type within a patch.

Attribute values for the patch core are found by first generating a new raster using `shrink`. Any size of edge effect can be modelled by iteratively stripping successive layers of edge cells. If the length of a cell edge is l , and the edge effect is d , then there are d/l

layers of edge cells. Figure 6.13 shows the core and edge areas for various shapes in a raster. Non-integer edge effects can be modelled by separating the whole number and fractional parts. First whole layers of cells are stripped away using shrink. This leaves a layer of boundary cells which are part core and part edge. Each boundary cell of the reduced patch now contributes a fraction to the core. For example, if the edge effect is 2.75 strip away 2 layers of edge cells. Each boundary cell now contributes 0.75 to the edge and 0.25 to the core. The contribution of each boundary layer cell is reduced in proportion to the fractional part of the edge effect.

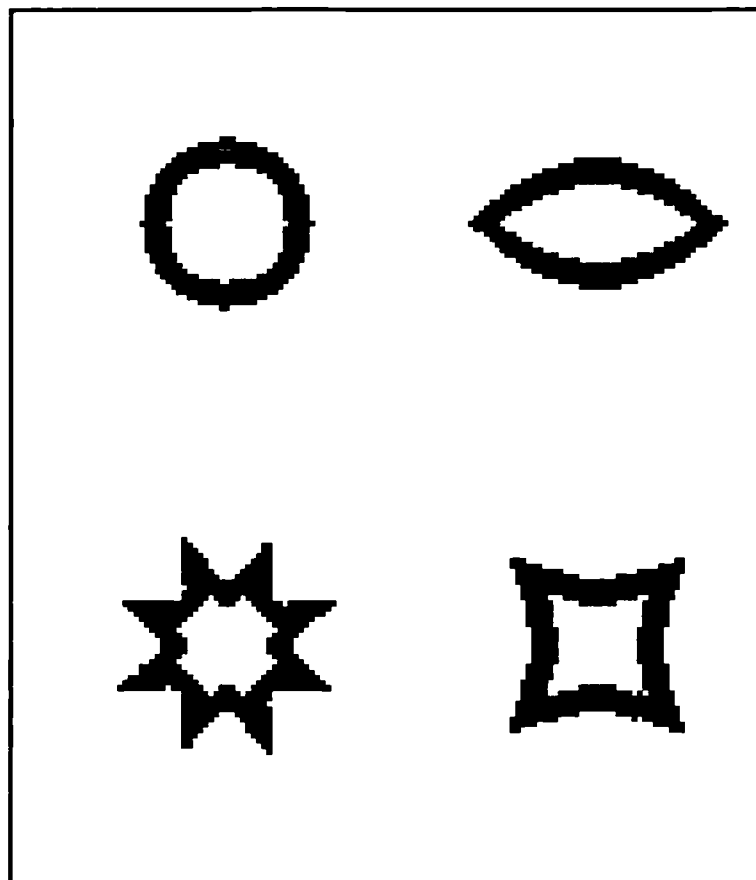


Figure 6.13 Core area and edge area for various shapes, from Brookes 1997b

6.6.3 Evaluation functions

Most aspects of MCDM relevant to GAPD were covered in chapter 2 and present no particular difficulties. However, there can be problems with search techniques when

constraints are involved and chapter 2 did not deal with costs which are often a significant factor in decision-making problems.

Attribute measurements are converted into utility scores using MCDM methods. The compromise programming method, explained in chapter 2, was used in all tests. Each attribute measurement is converted to a normalised score and the scores are combined using weighted summation.

Constraints are handled using thresholds and penalty functions. Constraints can apply to the minimum size of patches, minimum distance between patches, maximum total size of all patches and so on. Alternatives which do not satisfy a constraint can be awarded a zero score but this is not always satisfactory when using genetic algorithms. If too many alternatives score zero, then selection pressure can cause the algorithm to converge prematurely. To avoid this a **penalty function** is used to reduce the score for a criterion by a proportion which depends on the extent to which a threshold is exceeded.

An example of the use of a penalty function to model a soft constraint is given below. It is assumed that there is a size constraint of C cells. This figure is a soft upper limit; solutions which use more than C cells are allowed but are penalised. The penalty function takes the ratio of the actual size and the target size, multiplies it by a penalty factor, and divides the original utility score by the result. The penalty factor means that utility decreases as size increases once the threshold is breached. The penalty function is only applied to patches that exceed the threshold size. If the size constraint is C and the penalty factor is PF , then the penalised suitability, S_p , of a patch with area A ($A > T$) and original suitability S is given by equation 6.11.

$$S_p = \frac{S}{PF * (\frac{A}{C})} \quad \text{Equation 6.11}$$

The size of *PF* controls how hard or soft the constraint is. Larger values of *PF* mean harder constraints. Soft constraints are an example of uncertainty in decision-making.

Penalty functions can also be used to model costs. In reality there are often costs associated with an action. Many problems have two objectives: to maximise utility and to minimise costs. Penalty functions are a means of integrating constraints and costs into a single utility function. They penalise solutions that have high costs or that violate constraints. Costs are not considered further in this thesis.

6.7 Further developments

One thing that the implementation of GAPD within this thesis does not demonstrate is the use of different MCDM techniques. GAPD uses a roulette selection procedure in the search driver and this means that every alternative must be awarded an absolute fitness score. This can only be achieved using compromise MCDM methods. The option to use non-compensatory methods is available if other genetic selection procedures are used. Genetic algorithms can use different selection strategies (Potts *et al.*, 1994; Whitley, 1994) including tournament selection which requires only relative utility scores (Goldberg, 1989). In tournament selection, pairs of individuals are picked at random and a tournament decides which of the pair is selected for breeding. The tournament can be decided using non-compromise MCDM methods such as those mentioned in chapter 2. Incorporating tournament selection would involve reworking the genetic driver code.

Landscape and shape indices were not used in the evaluation functions because it is difficult to relate indices to fitness in a meaningful way. Further research is needed before indices can be used in evaluation functions. Some suggestions of applications which might make use of landscape indices are included in chapter 9. Ecosystems and landscapes are dynamic, which introduces a problem since reserves are static in their spatial location. To some extent the dynamic nature of ecosystems is already catered for in the above guidelines (eg. heterogeneity, minimum viable area, meta-populations). A more robust technique for evaluating patches might be simulation modelling. Simulation

modelling would capture the dynamic nature of the ecosystems themselves, the way populations grow and decline for example, and allow for modelling of external influences such as climate change and changing landuse outside the reserves. Using simulation models as evaluation functions is technically feasible but well outside the scope of this thesis, both because of the computer resources required and the lack of available models.

6.8 Summary

This chapter has described the logical and physical design of GAPD and some details of the implementation. Some technical notes on the implementation and some sample program code can be found in Appendix I. Stage one of the GAPD development was a reduced functionality version for single-patch problems. The transition to a fully functional GAPD presented several problems which were overcome by modifying the PRG component to grow multiple patches from a single genetic code. The next two chapters deal with testing of GAPD. Chapter 7 is concerned with conceptual issues and includes some investigations into issues of shape in raster maps. Chapter 8 describes five evaluation tests of GAPD in detail.

7. FEASIBILITY TESTS

7.1 Introduction

In the course of the development of GAPD, some studies were made to establish the feasibility of the GAPD concept. Section 2 of this chapter is concerned with issues relating to the representation of shape in raster data structures. Sections 3 and 4 describe feasibility tests to establish that GAPD is both effective and efficient.

GAPD generates and evaluates patches on raster maps. Patches are spatial objects but, although raster maps can be good for visualisation of objects, they are not good at representing objects. Previous chapters have discussed how configuration affects patch suitability in the real world and how configuration can be measured and evaluated using raster GIS. These discussions have not dealt with the external validity of measurements in the GIS: how well they relate to the real world. The first investigation into the feasibility of GAPD looked at this issue by considering the representation of shape in raster systems.

Further feasibility tests were made to investigate the effectiveness and efficiency of GAPD. To be effective, GAPD must generate good solutions. To do this, it must use a genetic code that can represent the full range of possibilities. Therefore, the PDC, which controls patch shape and is the basis of the genetic code, must be capable of representing shapes with widely differing characteristics. Also, the genetic driver must be able to manipulate the genetic codes in such a way that the full range of codes can be explored. GAPD must also be efficient: it must solve large complex problems within the constraints of the available resources. The GAPD prototype was used in all tests.

7.2 The issue of shape in rasters

7.2.1 Introduction

GAPD uses raster data structures because they are better for representing attributes that

vary continuously over space than either object or vector-based systems are. A disadvantage of rasters is that they are not so good for representing objects with sharp boundaries. GAPD generates spatially explicit patch maps in raster format. The patches are interpreted as spatial objects. Since the whole point of GAPD is to optimise patch configuration it is important to investigate how well spatial properties can be represented in a raster before attempting to incorporate them as criteria in a decision-making environment.

A number of experiments were run to investigate issues relating to how shapes are represented and how shape properties are measured in raster systems. There is no single measurement of shape, instead different properties related to shape are used to express shape (LaGro, 1991). Parameterised shape-growing (PSG) was used to generate patches of different size and shape. The experiments were used to investigate:

- how many cells are needed to make a meaningful shape;
- the accuracy of shape representation;
- the effects of PSG parameters on shape characteristics; and
- the behaviour of different shape properties.

7.2.2 The raster data structure

All representations of geographic space, whether as maps or in GIS, have three important properties: scale, resolution and extent. Extent is the size of the area being mapped and is important for large areas because the curvature of the Earth becomes apparent. Scale is the ratio between values in the real world, typically length, and their representation. Resolution is the ability to distinguish two entities. For the raster data structure, resolution is the most important. A cell in a raster represents a real area and the length of a cell side represents a real length. Therefore, resolution is an implied scale. However, a cell has no physical area in storage and can be displayed as any size, at any scale. The number of cells defines the extent.

The raster model divides space into discrete, finite square elements which are arranged

in an exhaustive and non-overlapping two-dimensional array. As a consequence, the size and perimeter length of geographic objects, and the distances between them, are discrete rather than continuous variables. The accuracy and precision with which geographic objects can be represented depends on the resolution of the raster. Raster GIS and remote sensing are closely linked, and since both use the term resolution in different ways, it is necessary to clarify the terminology. In remote sensing, resolution is the power to distinguish real objects on the ground and represent them as different objects in an image. Resolution is a property of the measurement device. In a GIS, resolution is the power to distinguish properties of the representations of objects. For example, it is perfectly valid to represent small, sub-pixel sized objects as single cells but it would then be wrong to measure their relative areas or perimeter lengths because in the raster they would be equal while in reality they are not. In this thesis, the resolution of an image is the length of the side of each cell. The cell side length obviously defines the area of each cell and consequently the smallest difference in area and the smallest difference in distance that can be measured.

In a raster, a patch or region is a cluster of contiguous cells. The usual terminology used for raster GIS is from Tomlin (1990). There are three types of spatial entities, cells, zones and neighbourhoods. Cells are the fundamental units: locations are represented by cells which have attribute values. A zone is a geographical entity that is distinct from other entities and is represented by contiguous groups of cells with equal attribute values. A zone is therefore equivalent to a patch. Neighbourhoods are defined by spatial relations among cells, such as adjacency, distance and direction. Tomlin (1990) also categorises spatial operations as local, zonal or focal. Local operations compute new values for a location based on other values at the same location. Zonal operations compute new values for a cell based on the values of other cells in the same zone. Focal operations compute values for a cell based on values of cells within a neighbourhood. The shape scoring routine in PSG is a neighbourhood function. Calculation of patch composition is a zonal operation and computation of inter patch relationships combines neighbourhood and zonal operations. For example, to find inter-patch distance the first step is to find the shortest distance from each cell in the patch to the other patch (a

neighbourhood operation) and then find the smallest of those values (a zonal operation).

The spatial properties of the patch (e.g. the size, perimeter length and shape) depend on the resolution of the raster. A cluster of cells represents an actual object in the real world; the spatial properties as measured in the raster will be different from the properties of the real object, however, because of the discrete nature of the raster GIS model. The relationship between resolution and shape is more complex than that between resolution and size. For example, the PI ratio, which is a simple measure of shape, is a function of both perimeter length and area (LaGro, 1991). The orientation of a shape relative to the raster grid will also affect the representation of that shape and hence its measured properties so that the same shape will appear differently if its orientation changes. This effect is apparent by considering a square, first with sides parallel to the orientation of the raster and then reoriented at 45 degrees. Measured by counting edges the second square will have a longer perimeter. The way that spatial attributes are defined will also affect their measurement.

7.2.3 Measuring shape on a raster

Measuring irregular shapes is always problematic but in a raster a further difficulty arises because of the resolution. The number of cells needed to represent a real patch depends on the resolution. The finer the resolution the more accurately the patch can be represented. Intuitively, it is obvious that for very coarse resolutions shape is meaningless. For example, if a patch consists of only one pixel then it must be square. Patches of two, three or four cells, will have blocky, unrealistic forms. How many cells are needed before shape becomes a meaningful property?

The two measurements used in the experiments are **PI ratio** and **core area (CA) ratio**. The PI ratio is the ratio of the perimeter of the shape to the perimeter of a circle of the same area. The PI ratio for a circle is 1. The CA ratio is the ratio of the core area to the total area for a given edge effect, d , where edge effect is that defined in the core area

model described in Chapter 3. For a circle of radius R , the edge effect can be expressed as a ratio, $r = d/R$. Then the CA ratio is given by equation 7.1 or 7.2.

$$CA \text{ ratio} = \frac{(R-d)^2}{R^2} \quad \text{Equation 7.1}$$

$$CA \text{ ratio} = (1-r)^2 \quad \text{Equation 7.2}$$

The PI and CA ratios are not linear scales and, therefore, the different values cannot be precisely interpreted and cannot be directly compared with each other. The PI and CA ratios were converted to scores. The **CA score** is the difference between the actual CA ratio and the calculated CA ratio for a true circle. Likewise, the **PI score** is the actual PI ratio minus 1. Plots of CA and PI scores emphasise the differences between values and make for clearer plots. The experiments were run with the original PSG program, hence the parameters are N, D, A and O. In all experiments the value of D is fixed at 10.

7.2.4 Experiment 1 : Effect of measurement technique

This experiment investigates how two shape properties, the PI and CA ratios, vary with patch size. Varying patch size, expressed as a number of cells, is equivalent to varying the resolution for a fixed, real size. Circular patches were used because the PI and CA ratios of true circles can be calculated.

Theoretically, if the PSG shape control parameter N equals 10, the resultant patch will be a circle irrespective of the number of axes, A , and the orientation, O . N controls the ratio between the major and minor axes: when $N = 10$ all axes are the same length (see equation 6.1). A circle is the most compact shape so other values of N should have poorer scores for both the PI ratio and the CA ratio. A number of circles were grown using the theoretical values. Because it is not certain that the assumed parameters

actually do grow the best circles, a second series of circles were generated. A program searched for the optimal value of N , A and O for each size by trying different values in a systematic manner. The optimal value was found by comparing the PI and CA scores with the values for true circles. The search was intended only to give an indication of the difference between the best value and the theoretical value, and the search was confined to a small range of likely parameter values.

Patch size was varied from 10 to 200 cells in steps of 10, then in steps of 25 up to 300 and steps of 50 up to 500. The edge effect was kept constant at one cell so the relative edge effect decreased with increasing patch size. The CA scores for each patch are plotted in figure 7.1 and the PI scores are plotted in figure 7.2. Two sets of figures are plotted on each graph. One line shows the scores for shapes grown using parameters which were assumed to grow a circle (i.e. with $N=10$) and the other shows the best score achievable by searching for an optimal parameter set.

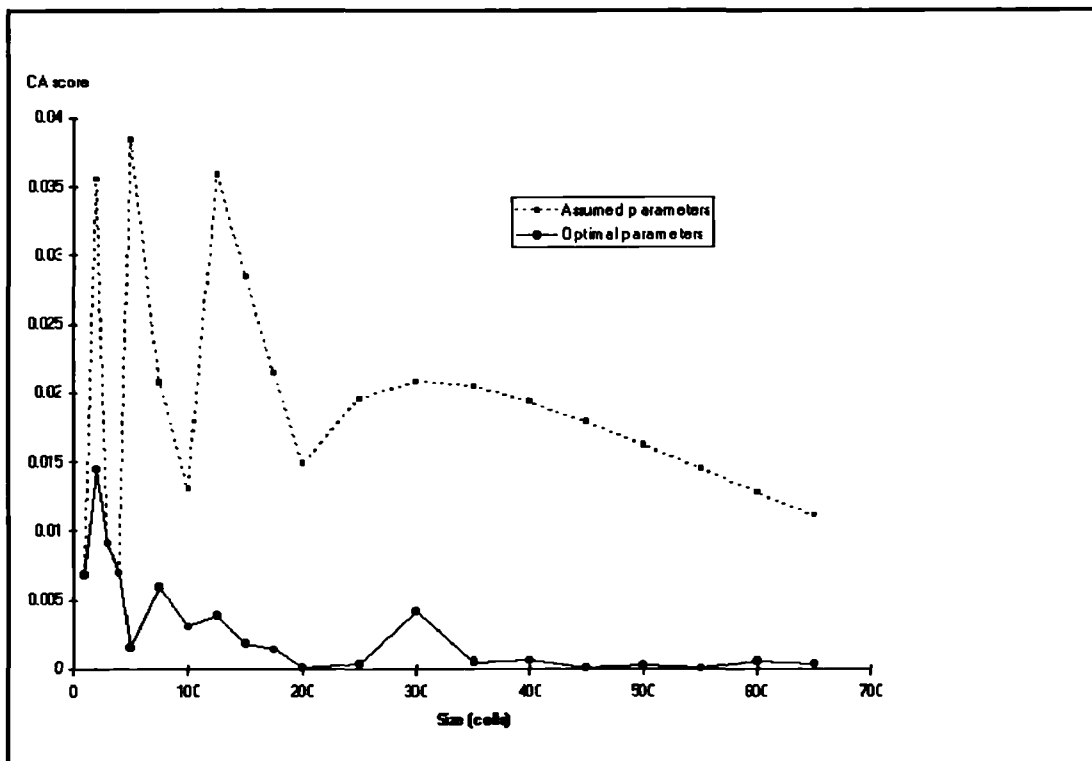


Figure 7.1 Variation of core area score with size

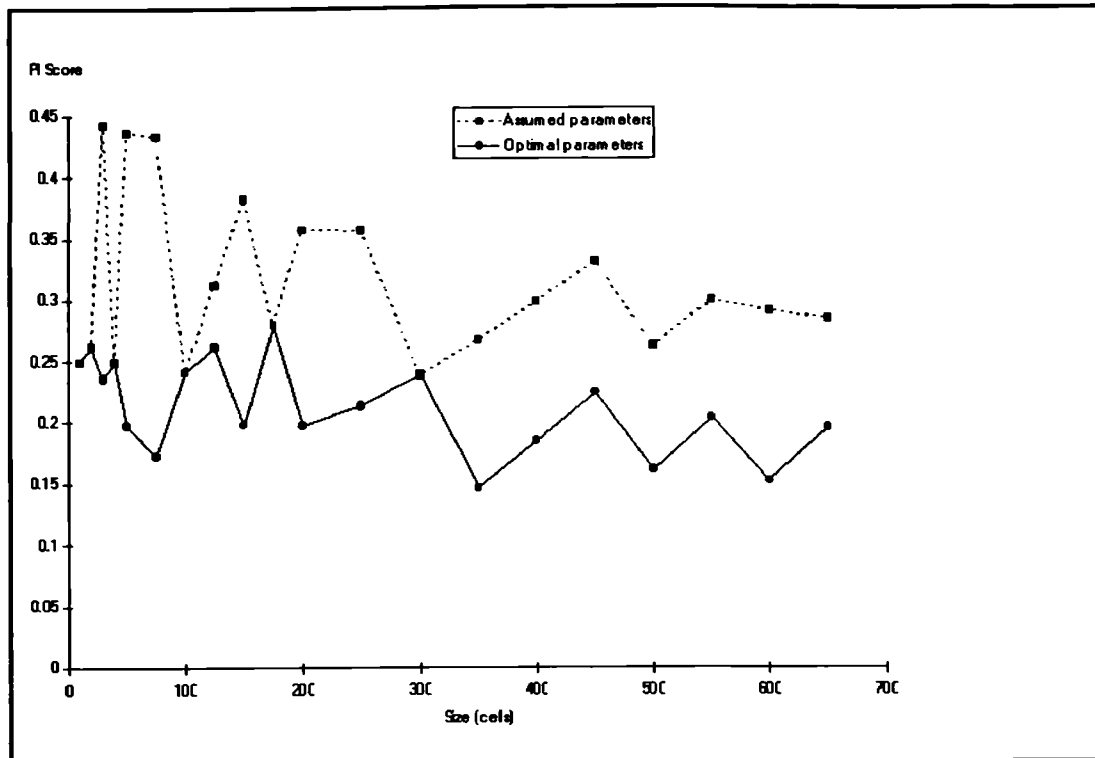


Figure 7.2 Variation of PI score with size

On the graphs zero is the optimal score and the most obvious feature of the graphs is that the assumed parameters are not optimal for all sizes. In fact for most sizes the assumed parameter solutions are sub-optimal. For shapes bigger than 50 cells none of the assumed parameters are optimal by CA but some are optimal by PI ratio until 300 cells. The results for size 300 show that for PI ratio the assumed parameters are optimal but for CA they are sub-optimal. Thus, PI and CA are not equivalent measures. If CA is used to assess circularity the best shape has a different parameter set than if PI is used. There is a lot more variability in the scores for the assumed parameters than for the optimal parameters. The optimal parameter values are different for different patch sizes.

For the PI ratio the optimal scores do not change much with increasing size though there is a slight downward trend. The relationship is not monotonic. The line has a marked saw-tooth appearance so that some small patches are more circular than some bigger patches. On average, larger shapes appear to be slightly better approximations than small ones. For CA there is a stronger downwards trend indicating that larger shapes are better approximations than small ones. However, this is probably because the edge effect

is getting relatively smaller with increasing patch size. None of the lines is monotonic: a small shape may be closer to a circle than a slightly bigger one. Small variations in size can have large effects on shape measures. This means that the resolution of the raster affects measurements of shape in non-linear way. The interplay between resolution and PSG parameters is complex. The parameters which grow the most circular patch of a certain real size depend on the resolution of the raster.

7.2.5 Experiment 2 : Varying the shape control parameters

This experiment investigates the effect on core area of varying both the number of axes and numerator parameters, A and N . The denominator, D , is fixed at 10 and the orientation, O , is fixed at zero.

The number of axes parameter controls the waviness of the shape's perimeter and the numerator controls the depth or amplitude of the waves. Figure 7.3 shows how the core area varies with number of axes for 3 values of the numerator, 1, 5 and 9. For all points the patch size is 100 and the edge effect is 1. For a circle the core area would be 68 (67.69). Each line in the figure corresponds to one value of the ratio. There are two noticeable features of the graph. First, the lines are not smooth and are not monotonic. Second, the characteristics of the three lines are quite different. The line for $N = 9$ is almost flat at a constant value of around 69 or 70 which is close to that for a circle. The $N = 5$ line follows a shallow U shape: the core area first decreases with increasing number of axes and then begins to increase again. Once the number of axes becomes large the major axes begin to merge into each other and visually the axes are not distinct. The line for $N = 1$ is a much deeper U shape decreasing until A is about 30 then increasing more slowly until the core area is almost equal to that of a circle.

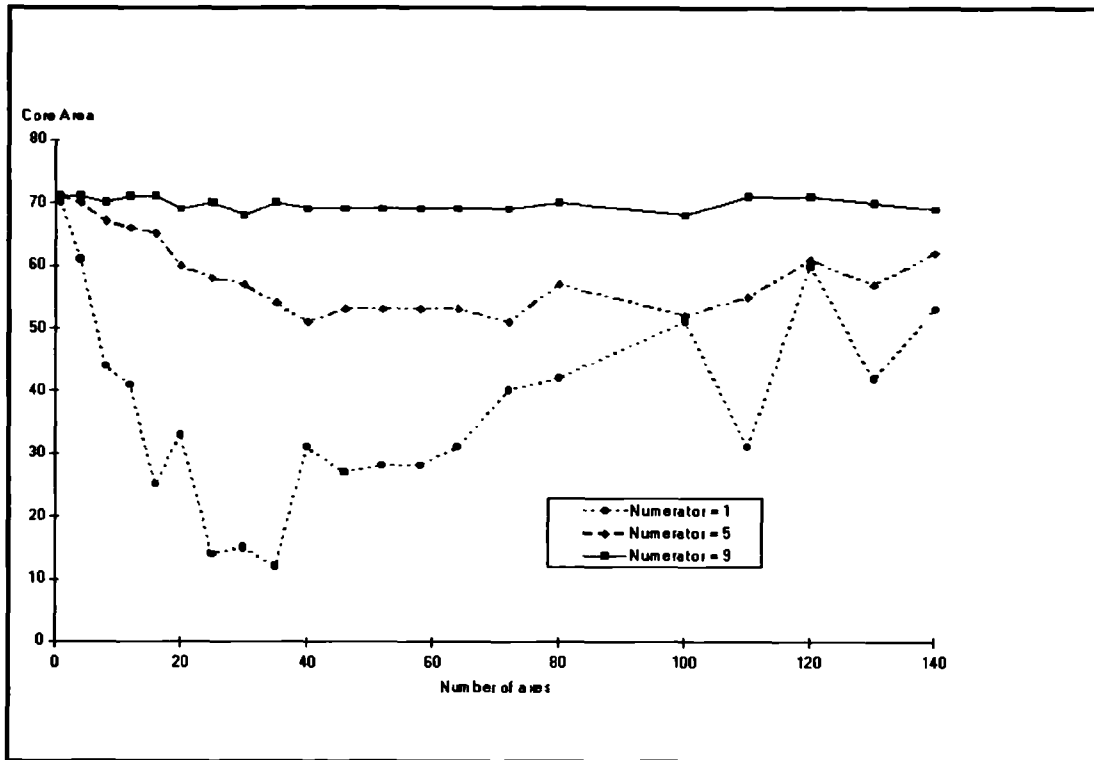


Figure 7.3 Variation of core area with number of axes for three numerator values

A number of conclusions follow. First, the effect of the number of axes parameter on core area depends on the numerator parameter. Second, the relationship between core area and number of axes is not linear. Third, at high values of the number of axes all shapes are approximately circular. The jagged form of the lines on the graphs, especially for $N = 1$, is an effect of the discrete nature of the raster.

7.2.6 Experiment 3 : Effect of size

This experiment investigates how the CA ratio varies with size when the relative edge effect is constant. For a perfect geometric circle the CA ratio is constant for a constant relative edge effect irrespective of the size of the circle. However, in a raster the core area ratio can be expected to vary for two reasons. First, because the actual shapes are approximate and second because of the way the edge effect is modelled. If the edge effect is an integer then the edge effect is also approximate and the relative edge effect is not actually constant. In all experiments the numerator parameter is 10. From the tests in 7.2.4 this value is known to be sub-optimal, i.e. it does not give the best circle,

but is chosen so that variations in N do not mask out the effects being investigated.

In this experiment the edge effect is defined as $1/4$ of the radius of a circular patch. For circles of different sizes, the core area and core area ratio are computed and plotted against total area. The results are plotted in Figure 7.4. The radius of each patch is an integer, and because the edge effect is an integer, the edge effect in the model is sometimes an under or over estimate of the actual edge effect. Successive groups of four points have equal edge effects in the model so that the value is only accurate for 1 in 4 patches and is inaccurate for the other three.

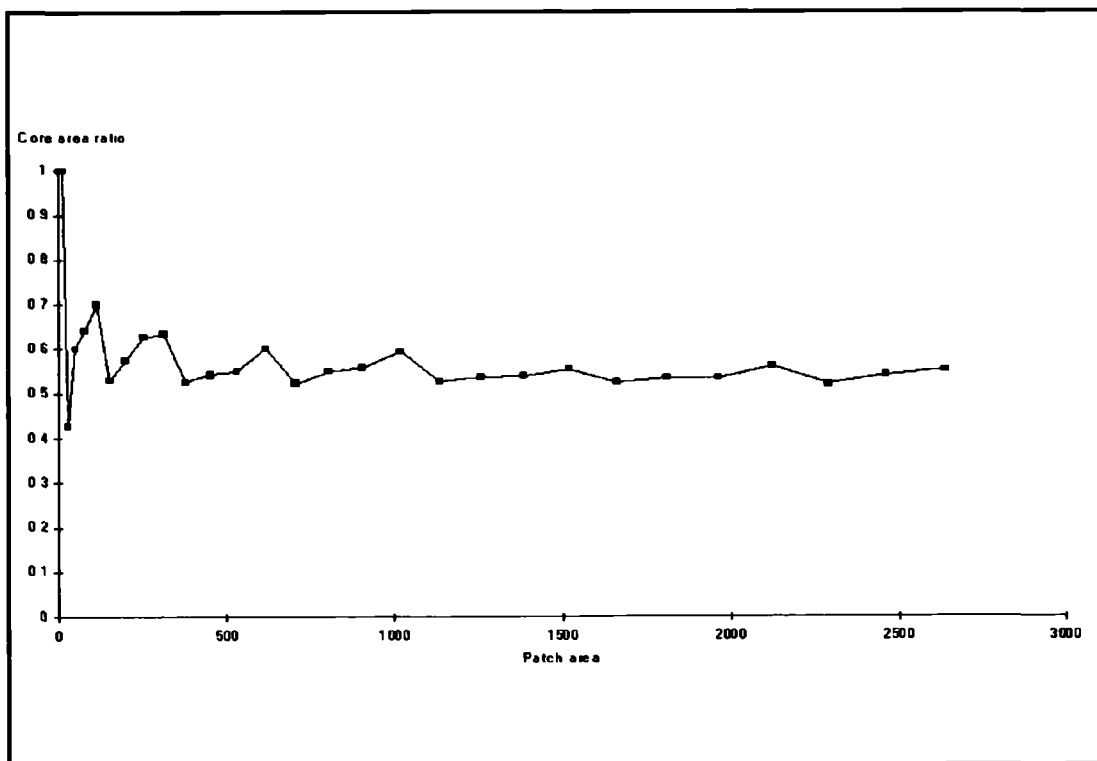


Figure 7.4 Variation of core area with size for constant proportional edge effect

The graph of core area ratio against total area (Figure 7.4) shows an attenuated oscillation. There are three features to note: the pattern of points, the spread of core area ratios and the general trend. The first two data points are special cases; the edge effect is zero, and therefore the CA ratio is 1. Starting with the third data point there is

a repeating pattern of three increases in CA ratio followed by a decrease. The decrease occurs when the edge effect increases by 1. The following three points have equal edge effects. The same pattern is repeated for every set of four points but the spread of values decreases as patch size increases. The trend is towards a core area ratio which is slightly greater than the value for a geometric circle (0.5625).

The lowest core area ratio for each edge effect occurs when the edge effect has been rounded up (i.e. over estimated). The higher values occur when the edge effect is accurate or has been under estimated. As patch size increases the spread of values gets smaller indicating that for large patches the integer approximation of edge effect is less important than for small patches. The trend is towards a value which is greater than the geometrical value (0.5625) and this result agrees with the results in 7.2.4, that the assumed parameters do not accurately represent a circle as defined by core area ratio.

7.2.7 Experiment 4 : Effect of shape control parameters

This experiment investigates the interplay between the numerator and number of axes parameters.

This test is similar to the test in 7.2.5 but the numerator and number of axes parameters are varied to produce different shapes which are almost circular. With three values for the number of axes, A , (8, 10 and 12) and three for the numerator, N , (8, 9 and 10) there are nine combinations and nine different shapes. The patch sizes and edge effects are as for test 7.2.5. Figures 7.5, 7.6 and 7.7 show the variation in core area ratio with patch size for three values of the numerator when the values of the axes parameter are 8, 10 and 12 respectively. The line representing a true circle has a core area ratio of 0.5625. In each figure the line for $N = 10$ (which is assumed to give a circle) is actually a poorer approximation to a circle than the lines for $N = 8$ and $N = 9$. Apart from when the patch area is small the lines for $N = 9$ and $N = 10$ are consistently above the true circle line whereas the lines for $N = 8$ are most often below the line. Significantly, for different

patch areas different lines are closest to a true circle. There is a complex interplay between the way shapes are represented and the way core area is measured on a raster. The fact that $N = 10$ does not give the best approximation of a circle is demonstrated in test 7.2.4. Figures 7.5, 7.6 and 7.7 show that the optimal parameters also vary with size. Most of the points do not fall on the line of a true circle which means that there are few true circles and many points fall above the line which means that some shapes have greater values than is geometrically possible. Neither the method of estimating edge effect nor the representation of circles is perfectly accurate.

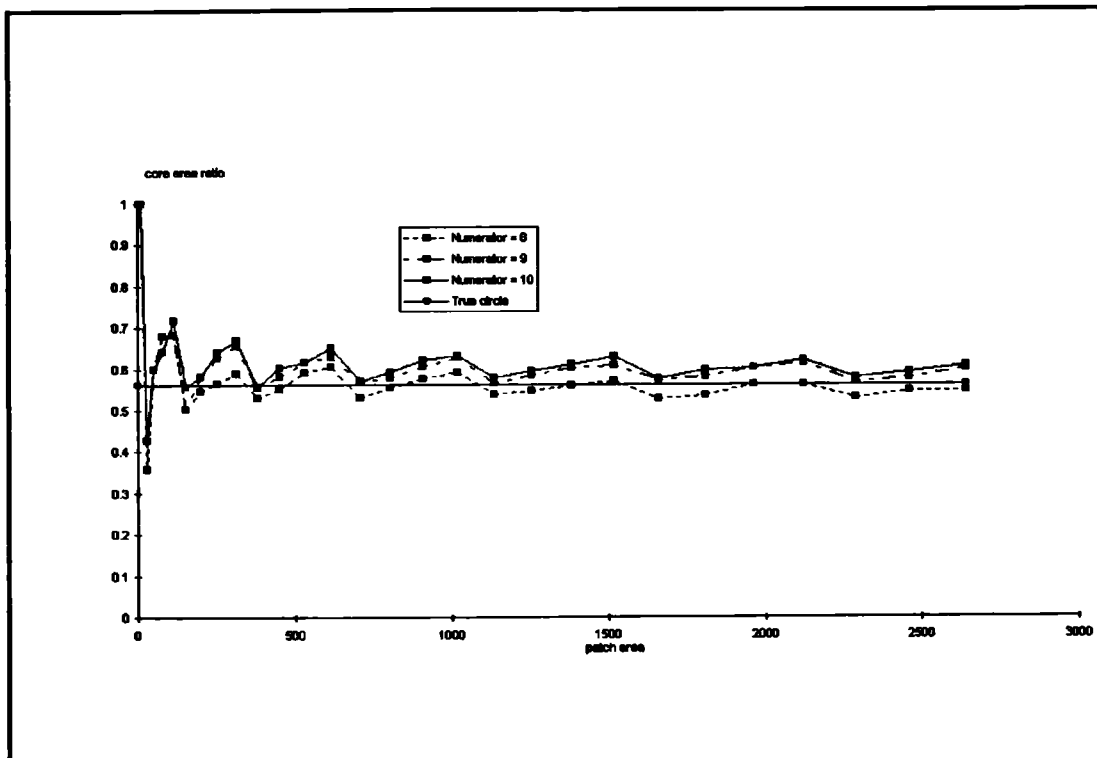


Figure 7.5 Variation of core area with size for number of axes = 8

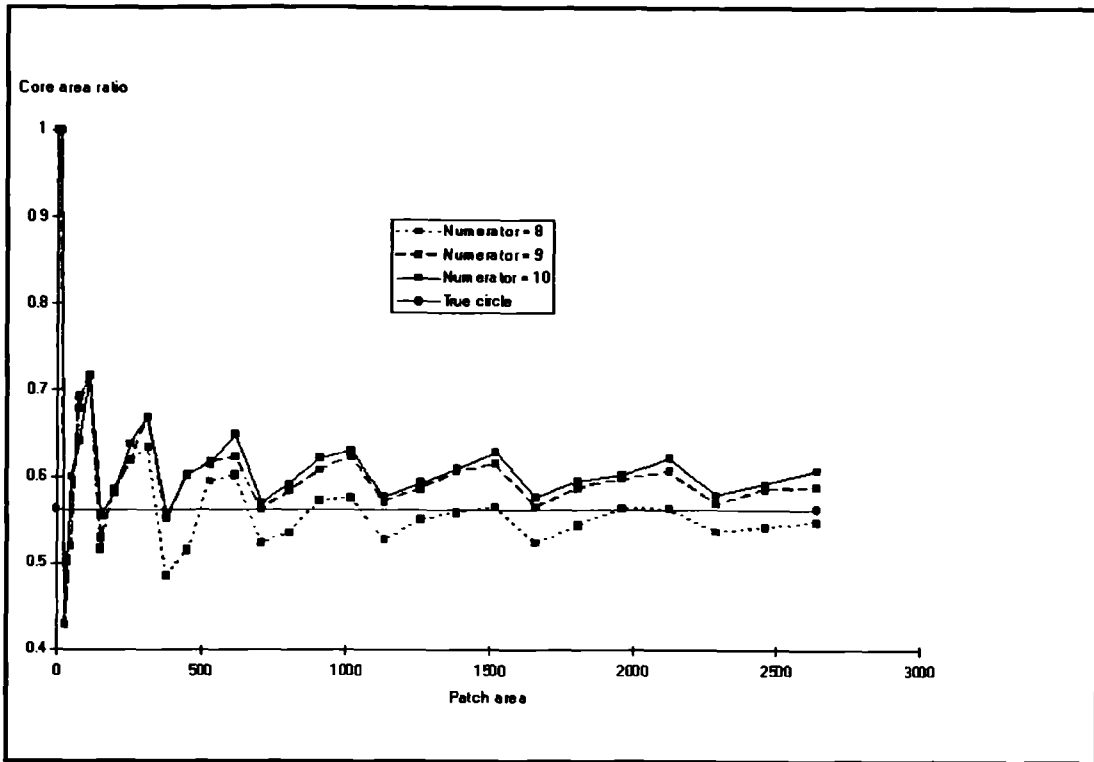


Figure 7.6 Variation of core area with size for number of axes = 10

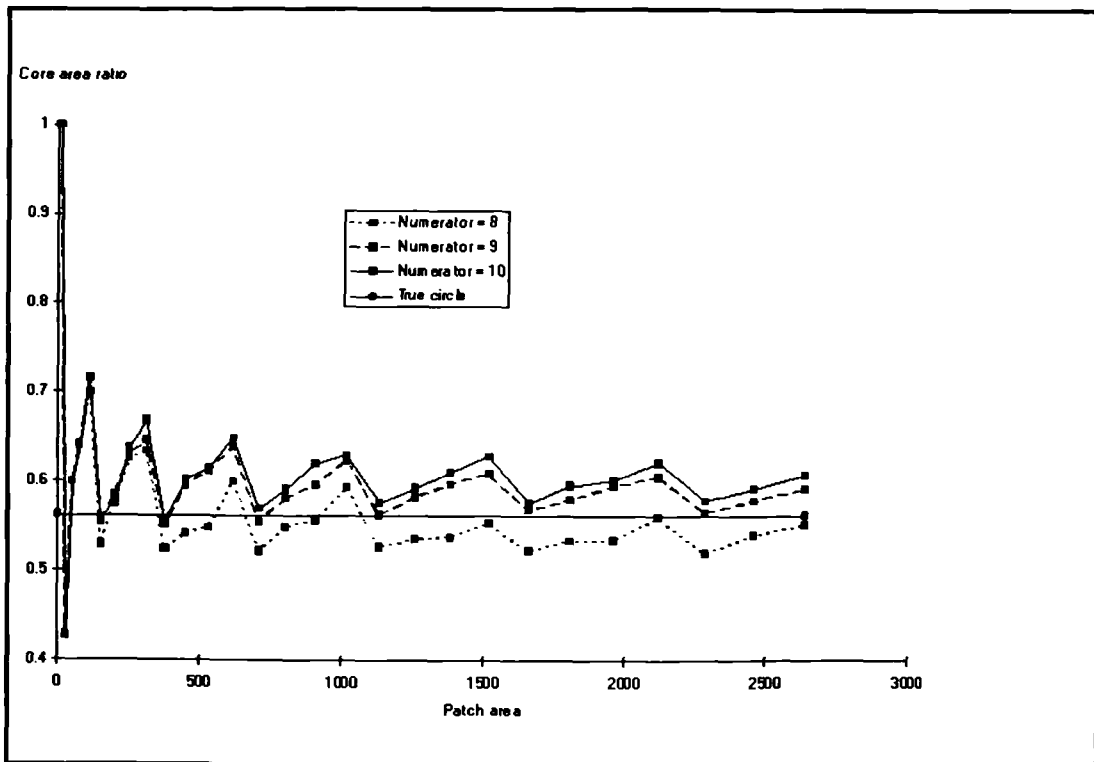


Figure 7.7 Variation of core area with size for number of axes = 12

7.2.8 Conclusions

The conclusion from these results is that the relationship between the PSG parameters, the resolution of the raster and the method of measuring shape is complex. Each element has an effect but it is difficult to separate out the effects of different components.

The main observations are:

- The parameters which might be assumed to grow the best circle do not, in fact, do so.
- The PSG parameter values which grow the best circle differ with resolution and method of measuring shape.
- As the number of axes parameter increases, patches tend towards a circular shape.
- The CA and PI ratios are not equivalent measures of shape.
- Using an integer value for edge effect has a more detrimental effect at small patch sizes than large ones.
- Beyond a certain patch size, of approximately 100 cells, good approximations of circles are possible.

In relation to the four objectives set out at the head of this section the conclusions are:

- Shape is meaningful when patches are bigger than about 100 cells.
- The accuracy of shape representation is variable and larger shapes are not always more accurate than smaller ones.
- The effects of the shape control parameters are complex.
- Different measurement techniques are not equivalent.

The results of these experiments have important implications in the context of optimal patch design in raster GIS. They indicate the limitations of any attempt to incorporate configuration into spatial planning. They also demonstrate the complexity of the problem.

As demonstrated in experiment 3, shape is not meaningful for very small patches and the CA scores suggest that the critical size is about 100 cells. This has important implications

for the resolution of the raster data because of the relationship between patch size measured in cells, actual geographic patch size and the resolution of the dataset. When patch size is a criterion smaller patches will require finer resolution datasets than larger patches. In real applications data must be acquired at sufficient resolution to represent patches meaningfully in the raster maps. In the test problems patch sizes of 100 cells or more were specified when shape was a criterion. The second conclusion, that shape accuracy is variable, means that there is uncertainty in the spatial measurements which does not behave in a simple way e.g. larger patches are not always more accurate than smaller ones. Intuition says that the orientation of the raster will also affect measurements. These uncertainties are in addition to uncertainties in base data criteria and objective functions. This is an important area for further research into the sensitivity and robustness of GAPD solutions. The complexity in the response of shape properties to shape parameters also has important implications. Because actual spatial properties cannot be predicted from PSG parameter values, a user would have difficulty choosing optimal values in a supervised system. An unsupervised system which finds optimal parameter sets automatically can overcome this problem, as has been demonstrated in 7.2.4. The effects of changing PSG parameters are non-linear, which implies a complex decision space. For PRG, which also takes aspatial cell suitability into account, the complexity is increased. High complexity of the search space calls for an intelligent, adaptive search mechanism. The complex interactions between parameters, resolution, measurement method and the added complication of spatial variability in the raster suitability map, suggest that it will be difficult to devise problem-specific heuristics. Genetic algorithms, which are general purpose and adaptive, are potentially able to overcome these difficulties. As well as being difficult to choose PRG shape parameters *a priori* it is also difficult to choose locations because suitability maps can be misleading to human operators. GAPD also addresses this problem directly and goes further by removing the need for suitability maps altogether because it can operate on the raw data. One of the test problems was subsequently designed to use raw data. The final conclusion, that different measurement techniques are not equivalent, means that the way configuration is measured must be chosen carefully. Measurements of properties that affect utility in well-defined ways, such as the core area model, should be used in

preference to vague shape indices such as PI ratio.

7.3 Effectiveness

The ability of the prototype to find good solutions was established by testing against 3 contrasting objectives:

- maximise core area,
- maximise edge area,
- maximise both core and edge area.

These three objectives were chosen because they demand completely different configurations and because the optimal values can be computed. The evaluation function for the third objective is the maximum of the minimum of core area and edge area. The optimal shape will, therefore, have equal core and edge area.

Compactness, measured using the core area model, is a useful way to distinguish shapes. There is a complete spectrum of possible shapes from the most compact to the least compact. The core area model gives exact values for the core area of the most compact shapes, the least compact shapes and for an intermediate shape whose core area is half the total area. The most compact shape is a circle. For a given edge effect and patch size the core area of a circle can be calculated. This value is a theoretical upper bound on the core area of any shape. At the other end of the spectrum the least compact shape will have a core area of zero; it will be all edge. In the middle of the spectrum will be shapes which have equal core and edge area and, therefore, have a core area equal to half the total area. For a shape of size 100 with an edge effect of 1, the theoretical core area values are: 0, 50 and 67.69. As shown in section 2 of this chapter it is possible for shapes in a raster to have core areas larger than the theoretical maximum. A reasonable estimate of the upper bound is 70.

On a uniform suitability raster in which all cells have the value 10, the expected utility values are 700 for optimal core area, 1000 for optimal edge area and 500 for optimal core and edge. The use of a uniform suitability surface eliminates the effects of intrinsic

cell scores on patch shapes and means that patch shapes will be exactly as specified in the PDC.

GAPD was run three times for each objective function. The results are shown in Table 7.1 and the patch shapes are illustrated in Figure 7.8. It is clear from Figure 7.8 that GAPD is responding to the different objectives. Although GAPD did not find an optimal solution each time, it came very close (usually within one cell) for each problem.

These trials show that the PDC (via PRG) can represent a full range of shapes and that the driver can generate them.

| Population = 50, Generations = 100 | | | | | | | | | | |
|--|-----|-----|------|----|-----|-----|----|-----|-----|--------|
| Maximise core area | | | | | | | | | | |
| Utility | Row | Col | Size | R1 | A1 | O1 | R2 | A2 | O2 | Weight |
| 700 | 14 | 23 | 99 | 7 | 0.7 | 18 | 8 | 4.3 | 147 | 8 |
| 700 | 34 | 10 | 98 | 2 | 1.9 | 106 | 10 | 5.8 | 104 | 1 |
| 720 | 68 | 20 | 100 | 6 | 1.8 | 130 | 9 | 8.0 | 79 | 6 |
| Maximise edge area | | | | | | | | | | |
| 940 | 44 | 24 | 99 | 0 | 0.7 | 86 | 0 | 2.8 | 159 | 8 |
| 990 | 63 | 23 | 100 | 0 | 1.5 | 91 | 0 | 1.3 | 179 | 8 |
| 990 | 20 | 65 | 100 | 0 | 1.7 | 2 | 0 | 1.9 | 40 | 5 |
| Maximise (Minimum(Core area, Edge Area) | | | | | | | | | | |
| 500 | 69 | 32 | 100 | 2 | 3.2 | 104 | 1 | 8.3 | 45 | 5 |
| 490 | 24 | 21 | 99 | 2 | 2.6 | 10 | 4 | 7.3 | 33 | 9 |
| 500 | 33 | 59 | 100 | 7 | 7.6 | 100 | 1 | 4.9 | 151 | 9 |

Table 7.1 GAPD solutions to optimise core area, edge area and both core and edge area on a uniform raster



Figure 7.8 Shapes optimising core (top row), edge (bottom row) and both core and edge area (middle row).

7.4 Efficiency

7.4.1 Introduction

The efficiency of the prototype was evaluated by comparing it with an exhaustive search in a number of trials. Each trial used the same problem but a different suitability map. It was not feasible to use an exhaustive search in the whole search space so the search space was reduced by specifying a fixed patch shape. GAPD was compared directly with exhaustive search in the reduced search space and then GAPD was run in the full search space. In the reduced search space GAPD was usually as effective as exhaustive search and more efficient. In the full search space GAPD was able to find better solutions with no more effort than used in the reduced search space.

In each trial, the objective was to optimise core area using the core area model (Chapter

4). The utility function was cumulative core suitability with an edge effect of 1. Cumulative core suitability is the sum of the intrinsic suitability values of each cell in the patch core, i.e. all cells more than 1 cell from the patch perimeter.

7.4.2 Test maps

The suitability maps were designed to simulate different landscape patterns. Each map consists of a number of natural patches with different suitability values and a matrix of intermediate suitability values.

The test maps were generated using PSG to grow compact shapes. The seeds were arranged in a regular lattice and the shape control parameters were randomly chosen but were restricted to those that would produce a compact shape. The size of each shape and the spacing between them is different in each map and the patches are arranged to cover most of the map. Each patch has a randomly assigned type that is interpreted as the intrinsic cell suitability. The possible values are {10,20,30,40,50,60,70,80,90} Cells in the spaces between patches were randomly assigned values from the set {30,40,50}.

Although the seeds are arranged in a regular lattice the patches were grown in a random order. Where patches overlapped the later ones covered the earlier ones. This means that the patch size varies within a map as well as between maps. The patches grown first are smaller because they are overlapped by later patches. Other patches are bigger because two contiguous patches of the same type count as one patch.

The maps were generated in two series with ten maps in each series. The maps in series one are numbered 1.1 to 1.10 and those in series two are numbered 2.1 to 2.10. Maps in series 1 were generated as described above. They have the original 9 suitability classes and patches are homogenous. The second series was generated by modifying the maps in series 1. A random number in the range {0:9} was added to each cell so that the patches in the series 2 maps are heterogeneous. There are 89 suitability values (10 to 99). Although patches are heterogeneous they are still recognisable as patches because

the suitability values within a patch are in a different range from those in neighbouring patches. On average cells are more similar to other cells within the patch than to those outside the patch. Maps 1.1 and 2.1 have a patch size of 1 so they were not actually generated using PSG. They are random maps in which each cell value is independent from neighbouring cells. However, there are recognisable patches in these two maps because of random clustering of cells with similar values.

The PDC size parameter varied between 1, in maps 1.1 and 2.1, and 640, in maps 1.10 and 2.10, but the actual patch sizes were different. The size of the largest, high-suitability patch was estimated using the IR algorithm with an arbitrary threshold of 90. The values are shown in Table 7.2. The patches are nearly always larger than specified in the PDC. Both series have the same change in pattern from map 1 through to map 10. High suitability patches are small, irregular and scattered on the low numbered maps and large on the high numbered maps. Figures 7.9 to 7.18 show the maps in series 2.

| Map | Estimated patch Size | Map | Estimated patch Size |
|------|----------------------|------|----------------------|
| 1.1 | 18 | 2.1 | 12 |
| 1.2 | 36 | 2.2 | 36 |
| 1.3 | 118 | 2.3 | 118 |
| 1.4 | 185 | 2.4 | 185 |
| 1.5 | 282 | 2.5 | 282 |
| 1.6 | 164 | 2.6 | 164 |
| 1.7 | 414 | 2.7 | 414 |
| 1.8 | 315 | 2.8 | 315 |
| 1.9 | 306 | 2.9 | 306 |
| 1.10 | 1084 | 2.10 | 1084 |

Table 7.2 Estimated patch sizes for maps in series 1 and 2

7.4.3 Method

The efficiency of the genetic search driver was tested by comparison with an exhaustive search. Both searches operate on PDCs and use PRG to convert the codes to raster maps. The size of the search space can be estimated as the product of the domain of each of the PDC parameters. The search space can be reduced by fixing the PDC parameters to that of a circle ($R=1, A=1, O=0$), so that the only variables are the seed row and column position. A circle is the most compact shape and has the largest proportion of core area so it is reasonable to assume that roughly circular patches will have high utility. Because the size, shape control and shape bias parameters are fixed, the size of the search space is reduced to the number of cells in the raster. Thus, the exhaustive search examines every alternative in a reduced search space by keeping shape control parameters constant and only changing the location parameters. It tries the same shape and bias parameters for every possible seed position.

The exhaustive search was tried with two values for shape bias. First, the shape bias parameter was set to 1, so that all patches were circles. The other value used was 0.5 which gives equal influence to spatial and aspatial suitability.

There were 80 rows and 80 columns in each map. Seeds were restricted to being more than 5 cells from the map boundary because the map boundary constrains patch growth. Thus, there were 4900 ($70*70$) possible seeds and, therefore, 4900 iterations were required to try every seed. Although exhaustive search always finds the optimum in a given space the global optimum may not be contained within that space.

GAPD was compared to exhaustive search by running in the same reduced search space for both map series. GAPD was then run with the whole search space. The population size and number of generations were set to 70 so that GAPD was allowed the same number of evaluations of the utility function as the exhaustive search.

7.4.4 Results

GAPD can never beat the exhaustive search if it runs in the same search space because exhaustive search always finds the optimum, but GAPD equalled exhaustive search 15 times out of 20 in the series one trials and 9 times out of 20 in the series two trials. Operating in the whole search space, GAPD was better than the exhaustive search 8 times for series 1 and 9 times for series 2. In each case GAPD was inferior to exhaustive search only once. When GAPD is restricted to the reduced search space, it does worse in series two than in series one, but when it is unrestricted, it does better in series two. The results are listed in Table 7.3 with separate columns for the fixed and variable shape bias cases.

All the GAPD and exhaustive search solutions are plotted in Figures 7.19 and 7.20. For low-numbered maps the utility increases sharply as the natural patch size increases. For intermediate maps the increase is slower and for high-numbered maps the utility levels off. There is little change in utility with natural region size once the natural regions are larger than the allocated patch. A shape bias of 1, which results in circles, works best for maps with small natural regions. On maps with larger regions, a shape bias of 0.5 gives better results. The GAPD solution is the best on most maps and is never the worst.

| Map | Exhaustive search | | Genetic Algorithm | | |
|------|-------------------|------|-------------------|-------------|------|
| | Shape bias | | Shape bias | | |
| | 0.5 | 1 | 0.5 | 1 | Free |
| 1.1 | 1400 | 2440 | 1370 | 2440 | 2490 |
| 1.2 | 4370 | 4860 | 4370 | 4800 | 4780 |
| 1.3 | 5670 | 5770 | 5670 | 5650 | 5830 |
| 1.4 | 6120 | 6210 | 6120 | 6110 | 6390 |
| 1.5 | 6300 | 6210 | 6300 | 6210 | 6300 |
| 1.6 | 6390 | 6210 | 6390 | 6210 | 6480 |
| 1.7 | 6270 | 6210 | 6210 | 6210 | 6480 |
| 1.8 | 6390 | 6210 | 6390 | 6210 | 6390 |
| 1.9 | 6390 | 6210 | 6390 | 6210 | 6480 |
| 1.10 | 6390 | 6210 | 6390 | 6210 | 6480 |
| Map | Exhaustive | | Genetic algorithm | | |
| | Shape bias | | Shape bias | | |
| | 0.5 | 1 | 0.5 | 1 | Free |
| 2.1 | 2596 | 4229 | 2244 | 4022 | 4410 |
| 2.2 | 4662 | 5157 | 4662 | 4853 | 5178 |
| 2.3 | 5931 | 6087 | 5931 | 6087 | 6258 |
| 2.4 | 6424 | 6545 | 6420 | 6419 | 6632 |
| 2.5 | 6650 | 6569 | 6465 | 6514 | 6508 |
| 2.6 | 6658 | 6575 | 6565 | 6575 | 6742 |
| 2.7 | 6645 | 6558 | 6645 | 6558 | 6833 |
| 2.8 | 6731 | 6570 | 6731 | 6538 | 6830 |
| 2.9 | 6712 | 6546 | 6535 | 6546 | 6814 |
| 2.10 | 6670 | 6579 | 6670 | 6577 | 6866 |

Table 7.3 Comparison of GAPD and exhaustive search in the GAPD efficiency tests

An upper bound for utility can be estimated by considering the largest possible core area in a patch of 100 cells. The theoretical value is 68 but in practice it can be as high as 72 (see section 7.3). The upper bound is, therefore, 6480 (72×90) for maps in series 1 and around 7128 (72×99) for maps in series 2. GAPD found optimal solutions for maps 1.6 1.7 1.9 and 1.10. In series 2 the actual optimal solution is probably less than the upper bound because the patches are heterogeneous. Many of the GAPD solutions are close to the upper bound and may be optimal. The upper bounds are likely to be greater than the actual optimum on the low-numbered maps because there are no large regions of high-scoring cells.

Table 7.4 shows the solutions and patch images for map 2.1. The results show that 0.5 is not a good value for shape bias. The patch is not compact and it has a small core area. The 0.5 shape bias patch has the best average cell suitability for the whole patch but has a low utility score because of its small core. GAPD found the best patch by finding the best patch configuration.

8. EVALUATION TESTS

8.1 Objectives and methodology

8.1.1 Objectives

The purpose of this research is to develop a working computer system, GAPD, to design optimal patch configurations. A goal is to develop a general technique which will work for a range of problems. There are, therefore, three main objectives to the evaluation tests:

- to assess how effective GAPD is in solving problems;
- to see how efficiently GAPD finds solutions; and
- to see how GAPD can be adapted to different situations.

Quantifying how well GAPD meets these objectives is difficult. The effectiveness of GAPD was assessed by comparing solutions with those found using other heuristics, by calculating theoretical upper bounds or by contriving a problem with a known solution. The effectiveness of GAPD can also be assessed by visual inspection of the patch maps to verify that GAPD respond to changes in spatial criteria by generating different configurations. Efficiency was judged by considering the size of the search space and the number of evaluations of the utility functions needed to converge on a solution.

The adaptability of GAPD was investigated by running five evaluation tests that cover a range of situations. Useability and applicability of GAPD were not rigorously investigated but these aspects were considered during testing and are commented on in the test descriptions.

8.1.2 Test design

Tests differ in the number of patches in the solution, the number and type of input data layers, the criteria and the number of objectives. The simplest applications have a single

patch, a single suitability layer and a single objective, while the most complex problems have multiple patches, multiple objectives and use multiple data layers. Classification is according to the problem constraints, not the form of the solution. A multi-patch problem has no constraint that the solution be a single patch but the optimal solution may turn out to be a single patch. Intermediate problems in which the number of patches is fixed, and greater than one, are referred to as fixed multi-patch as distinct from variable multi-patch.

The evaluation tests show how GAPD can use suitability maps or raw data layers, and multiple or single input layers. Finally, the problems illustrate a range of evaluation functions using composition and configuration. Table 8.1 summarises how the different tests cover a range of situations. The tests use a small number of fundamental evaluation routines combined in different ways.

The first evaluation, test 1, is the simplest. It uses the same data and utility function as were used in the original development of the PRG. Results of earlier work on PRG (Brookes, 1997a) were used as a benchmark for evaluating the quality of the GAPD solutions. The objective is to maximise core suitability of a single patch. Test 2 emphasises configuration; the objective is to design multiple patches with constrained locations. Test 3 is a multiple-objective problem which combines elements of test 1 and test 2: maximise core suitability and minimise isolation. Maximising core suitability pushes the solution towards a single compact patch, while minimising isolation pushes the solution towards patches which are not compact.

Test 4 uses a utility function based on a different measure of patch composition. The objective is to locate patches that have a core area with a given minimum composition. This problem differs from the first three in using attribute data layers not a suitability map. Variations on this problem were run with the solutions constrained to be a fixed number of patches or allowed to be a variable number of patches. In some cases the patch sizes were fixed and in others they were unconstrained.

| Test | Number of Patches | | | Objectives | | | Data | | |
|------|-------------------|-----------|----------|------------|----------|-------------|--------------|-----------------|----------|
| | One | Fixed > 1 | Variable | Single | Multiple | Conflicting | Suitability | | Raw data |
| | | | | | | | Single layer | Multiple layers | |
| 1 | X | | | X | | | X | | |
| 2 | | | X | X | | | | X | |
| 3 | X | | | | X | | X | X | |
| 4 | X | X | X | X | | | | | X |
| 5 | | X | | | | X | | X | |

Table 8.1 Summary of the five evaluation tests

The last test, test 5, is a variation on a real problem described by Eastman *et al.* (1993). The original problem was used to illustrate the use of GIS to solve planning problems with multiple conflicting objectives. In the original problem the solution was evaluated purely by composition without reference to the configuration of patches. Hypothetical spatial criteria were added and test 5 shows how GAPD can be applied to a multi-objective planning problem when the spatial configuration of patches is important.

The original PRG was designed to operate with a single suitability map. Suitability maps are not appropriate to every problem. Test 4 shows how GAPD can be applied by amending the PRG to operate on multiple data layers directly.

8.1.3 Test methods

Each test uses a different implementation of GAPD. As explained in Chapter 6, GAPD was not developed as a single software system. One objective in running several different tests is to assess how easy it is to implement GAPD in different situations. The implementations differ mainly in the PRG and the evaluation components. Common to all implementations are the genetic search driver and operators. Values for the operator probabilities, the genetic algorithm parameters and the domains are input via a control file, allowing variations in the driver without altering any program code. Each application has different objectives and criteria so a utility function must be coded for each individual problem. Alternative PRG components were needed because the cell scoring rules varied for different problems. The same shape scoring rules were used throughout. Several variations of the genetic code, that is the PDC, were used. This was partly so that GAPD could be developed incrementally and partly for pragmatic reasons. Some of the tests were run on a micro computer with limited memory, which meant that PDCs had to be as short as possible. The use of different PDCs does not necessitate changes to the genetic search driver because the program is coded in such a way that it can accommodate genetic codes of different lengths and domains.

A general problem in raster processing that also applies in patch design is caused by

boundary effects. The input raster data layers have a finite extent with hard boundaries. The growth of patches near the raster boundaries is constrained since patches cannot grow across the boundary. To avoid this problem the 5 rows and columns nearest the raster boundary, the **boundary zone**, are designated a restricted zone and no seeds can be located there. Patches can, however, grow into this zone so that patch shape is no longer constrained. The width of the boundary zone was set by considering the largest circle that could grow unrestricted if the seed were on the edge of the boundary zone. Most of the tests use a patch size of 100 cells. The largest circle centred on the edge of the boundary zone would have a radius 6 cells and would be bigger than 100 cells. A patch of size 100 can be assumed to grow unrestricted if the boundary zone is 5 cells wide. Theoretically the boundary zone should be adjusted depending on the anticipated size of patches. This was not done but none of the GAPD patches in testing were affected by boundary effects.

8.1.4 Test verification

GAPD reads and writes IDRISI (Eastman, 1993) format raster files and the results were verified using the IDRISI GIS. GAPD reports the utility of the best solution and also writes the PDC string and an IDRISI image and document file. The general procedure for checking answers is to use IDRISI to measure the composition and configuration of patches on the images. The appropriate scoring system is applied to the measurements to convert them to scores and then to utility values. The result should be the same utility score as was reported by GAPD. The choice of evaluation functions to test GAPD was, in part, influenced by the availability of functions in IDRISI which could be used to check the answers. Details of the verification procedures are included in the test descriptions.

8.2 Evaluation 1 : Designing a single compact patch

8.2.1 Purpose

The main purpose of this test is to see how GAPD performs in a simple patch design problem involving one patch and simple spatial criteria. The problem is simple so that GAPD can be compared to two other methods, one autonomous and the other supervised, which are likely to find good solutions. This is also the first test of GAPD with real data.

The problem has a single objective and a single input suitability map. The objective function - maximise core suitability - is the same as that used in the efficiency tests of the GAPD prototype described in Chapter 7 but, in this case, the suitability map is derived from a real spatial dataset. The suitability map has more complex spatial variations than the maps used previously: one significant difference is it includes barriers. The problem was used originally to evaluate the stand-alone PRG program (Brookes, 1997a).

The results from GAPD are compared to those from the IR heuristic and the stand-alone PRG. The IR heuristic optimises the composition of the whole patch but does not take account of configuration. Because the objective is to maximise core suitability, edge effects mean that more compact patches will have higher utility than less compact ones. GAPD optimises both composition and configuration and, therefore, should do better than IR. The comparison with the stand-alone PRG program is more significant. With a suitability map, an operator should be able to judge the best locations for patches by looking at the map and to judge what shape of patch will be the most compact. Therefore, an operator should be able to give appropriate parameters to the PRG program and generate promising patches. If GAPD finds better solutions than stand-alone PRG then that indicates the value of using an unsupervised computer search algorithm rather than relying on a human operator.

The quality of the solutions can also be evaluated by calculating an upper bound to the

solution analytically. The solution was constrained to be a single patch because the edge effect should mean that a single patch is superior to several small patches. This constraint facilitates calculating an upper bound.

8.2.2 Problem description

8.2.2.1 Objective function

The objective is to plan the location of a patch which maximises the suitability of core habitat. The evaluation function is cumulative core suitability. The method for calculating cumulative core suitability was given in Chapter 6. Three different scenarios are proposed: in the first (a) the total area is 150 and the edge effect is 1, in the second (b) the area is 200 and the edge effect 2 and in the third (c) the area is 300 and the edge effect 3.

8.2.2.2 Data

A suitability map was generated using the landuse and relief data from the sample datasets supplied with the IDRISI GIS (Eastman, 1993). Roads were given a suitability of zero and they effectively partition the suitability map into disconnected zones. Cell suitability scores were based on landcover, slope and distance from features such as urban areas and water bodies. The various factor maps were combined into a single suitability layer shown in Figure 8.1. This layer has 85 rows and 72 columns. Because of boundary effects, seeds were not permitted in the 5 rows and columns bounding the map. The number of possible seeds for a single patch was therefore 5360 (i.e. $80 * 67$). An estimate of the number of possible patches for each seed can be made by considering how the PRG algorithm works. Each cell has four neighbours. To avoid counting combinations twice, assume a row by row processing order so that there are now only two new neighbours for each neighbour added. Therefore, for a patch of size N , there are 2^{N-1} configurations. This figure is actually an underestimate. The argument is explained fully in Appendix IV. For a patch of size 150 there are more than 2^{150} ($> 10^{45}$)

possible configurations.

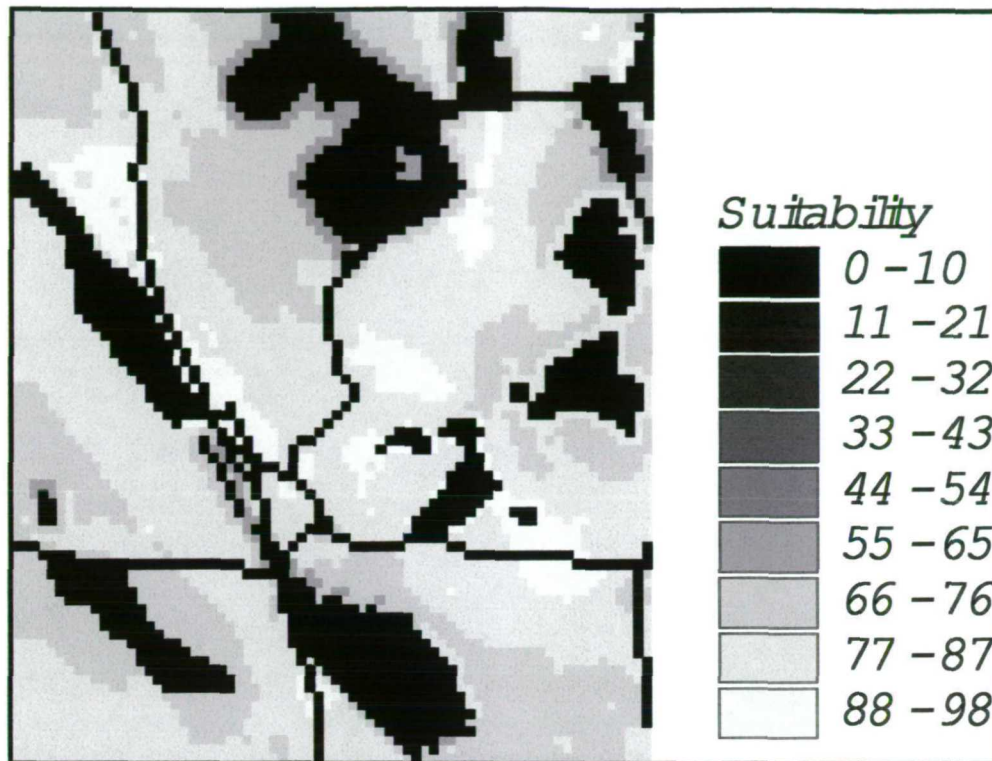


Figure 8.1 Suitability map for evaluation test 1

The PRG algorithm does not add diagonally-linked cells and, therefore, patches cannot grow across barriers without incorporating barrier cells. This factor inhibits the algorithm from crossing barriers because barrier cells have zero cell suitability. However, a very high weighting for spatial suitability would allow a patch to cross a barrier.

8.2.3 Method

GAPD

A single patch version of GAPD was used to solve the problem. The solution is constrained to be a single patch by having only one PDC in the tail of the genetic code (Table 8.1a). GAPD was run once for each allocation level.

Evaluation routines

The objective function was evaluated within GAPD as follows. First grow a patch under the control of the PDC in the genetic code on the input suitability map. Second, shrink the patch by the amount specified by the edge effect (1, 2 or 3 cells) and produce a new raster with the core patch. Overlay this layer on the suitability map and for each cell in the core extract the suitability value from the suitability map. The total of all suitability values is the cumulative core suitability which is the utility. This value is passed back to the genetic algorithm as the fitness score.

| Head | Tail | | | | | | | | |
|-------|----------|-----|-----------------|---|---|-----------------|---|---|------------|
| total | location | | shape control 1 | | | shape control 2 | | | shape bias |
| size | row | col | R | A | O | R | A | O | |

Table 8.1a Genetic code for evaluation test 1

The control variables for GAPD are shown in table 8.2. The population size was 100 and the maximum number of generations was 200 allowing for a total of 20,000 alternatives to be evaluated. This number is larger than the number of possible seeds but much smaller than the total search space.

| Genetic parameter | Value |
|-----------------------|-------|
| Population | 100 |
| Generations | 200 |
| Max tries | 20 |
| Crossover probability | 0.3 |
| Mutation probability | 0.1 |
| Average probability | 0.2 |
| Sum probability | 0.2 |
| Creep probability | 0.1 |
| Null probability | 0.2 |

Table 8.2 Genetic algorithm control parameters in evaluation test 1

IR

The IR heuristic, which was described in Chapter 4, was implemented in IDRISI as follows. First, rank the cells in the suitability map using the RANK command. Each cell now has a unique value. Then choose an arbitrary number of cells, *N*, to include in the solution. Make a layer TEMP which awards a 1 to the *N* highest ranked cells and zero to all others using the RECLASS command. Using the GROUP command, group the cells in TEMP into contiguous patches to make TMPGRP. Assign the area of each patch to its component cells, using the AREA command, to make TMPAREA. Every cell will be given a value, even those originally excluded because they have low rank. These cells are masked out by using the TEMP layer as a masking layer: the OVERLAY command is used to multiply the TMPAREA and TEMP layers together to create MAXAREA. In MAXAREA only the *N* highest ranked cells have non-zero values and, therefore, clumps of these cells are candidate patches. The area of the largest patch is found by looking at the IDRISI document file for MAXAREA. A binary search procedure can be used to find a value for *N* that gives a patch of the right size.

The algorithm does not guarantee to find patches of exactly the right size because the addition of one cell may join two existing patches. If the IR heuristic finds several patches larger than the target, the smallest such patch is taken as the solution. Another problem with the IR procedure is that patches may contain holes and that is not always acceptable. The iterations used to find solutions at the 3 allocation levels are shown in table 8.3.

| IR iterations | | IR iterations | |
|--------------------|--------------------|--------------------|--------------------|
| Number of cells, N | Largest patch size | Number of cells, N | Largest patch size |
| 500 | 112 | 1100 | 151 |
| 1000 | 120 | 1050 | 147 |
| 2000 | 353 | 1075 | 151 |
| 1500 | 352 | 1063 | 149 |
| 1250 | 214 | 1069 | 150--Finish |
| 1200 | 187 | 1300 | 303 |
| 1225 | 201 | 1275 | 228 |
| 1220 | 196 | 1287 | 236 |
| 122 | 199 | 1294 | 236 |
| 1224 | 200--Finish | 1297 | 300--Finish |

Table 8.3 IR iterations to find a patch of the right size in evaluation test 1

Stand-alone PRG

In the original experiments (Brookes 1997a), the PRG seeds were chosen by one of two methods (a) using the centroids of the regions found by the IR method or (b) by an operator viewing a display of the suitability map and identifying the coordinates of likely seeds using the mouse. The PDC parameters were set at those for a circle and the shape bias was fixed at 0.5. In method (b) three different seeds were tried.

8.2.4 Results

GAPD found the best solutions in every scenario. The results are summarised in table 8.4, which records only the best PRG results. The different solutions are illustrated in figures 8.2 through 8.10. The difference between the utility of the different solutions is a function of the differences in core area and average suitability. For the largest patch, scenario (c), the GAPD patch has a much larger core than the PRG patch but a lower average suitability. For the 200 cell patch, scenario (b), the GAPD patch has a slightly

larger core and slightly poorer suitability and the utility scores of the three methods are the closest. In scenario (a) the GAPD patch has a slightly greater core and also greater average suitability. The IR patches have poor utility, despite having high average cell values, because the core areas are small.

| Scenario | Algorithm | CoreArea | | Average cell value | Minimum cell value | Utility |
|-------------------|-----------|----------|----|--------------------|--------------------|---------|
| | | cells | % | | | |
| a. Patch area 150 | IR | 93 | 62 | 88.4 | 82 | 8223 |
| | PRG | 110 | 73 | 82.4 | 78 | 9108 |
| | GAPD | 113 | 75 | 84.5 | 78 | 9546 |
| b. Patch area 200 | IR | 61 | 31 | 84.9 | 82 | 5050 |
| | PRG | 112 | 56 | 81.5 | 74 | 9124 |
| | GAPD | 118 | 59 | 80.6 | 74 | 9448 |
| c. Patch area 300 | IR | 77 | 26 | 83.78 | 82 | 7904 |
| | PRG | 138 | 46 | 80.65 | 74 | 11129 |
| | GAPD | 161 | 54 | 77.85 | 74 | 12534 |

8.4 Solutions for the 3 methods in evaluation test 1

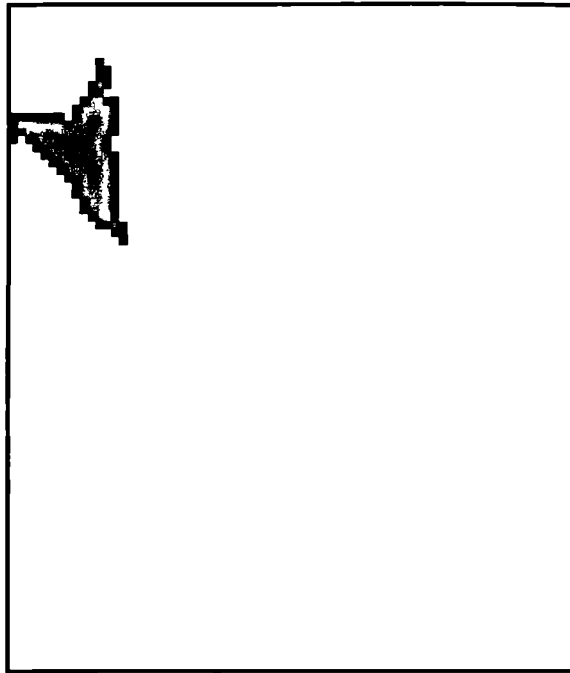


Figure 8.2 Patch found by IR for the 150 allocation in test 1

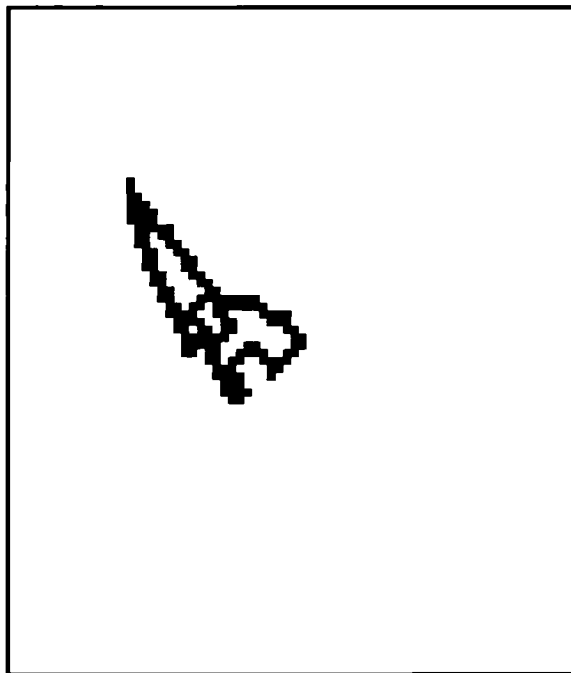


Figure 8.3 Patch found by IR for the 200 allocation in test 1

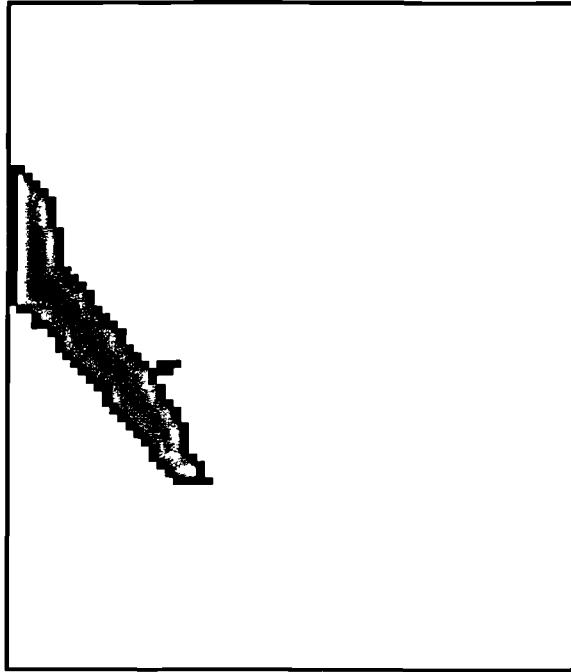


Figure 8.4 Patch found by IR for the 300 allocation in test 1

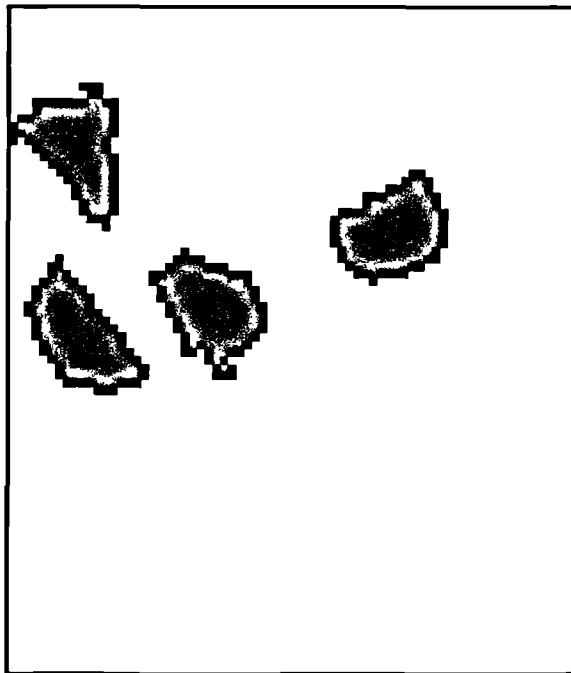


Figure 8.5 Four patches found using PRG for the 150 allocation in test 1

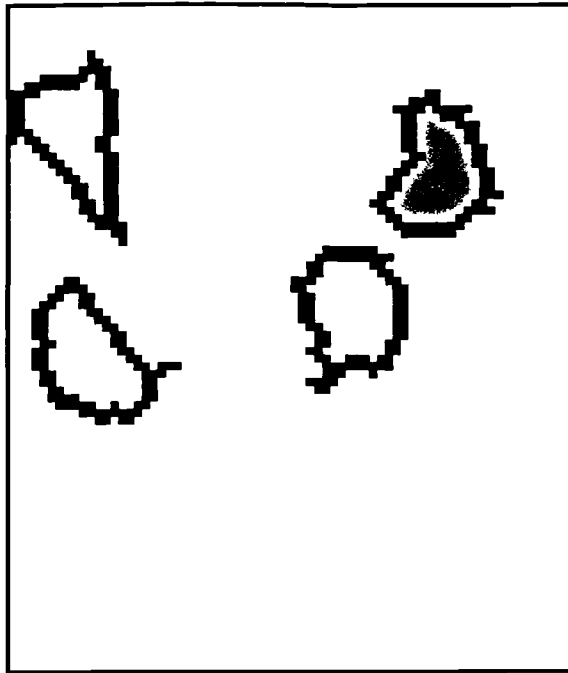


Figure 8.6 Four patches found using PRG for the 200 allocation in test 1

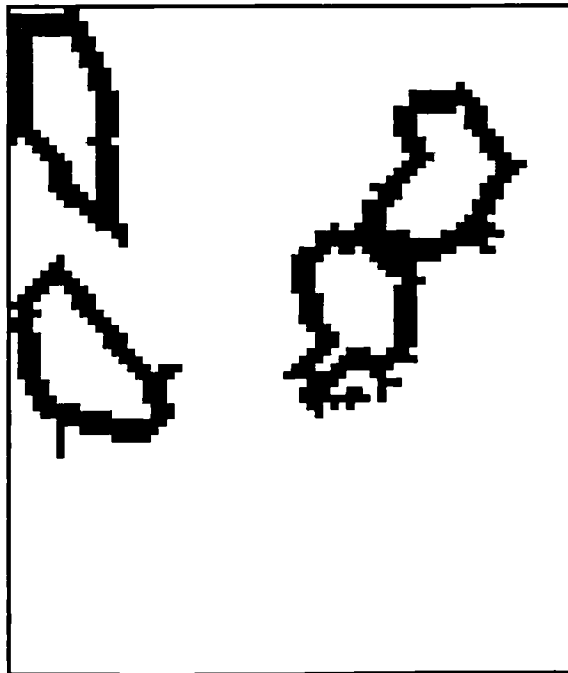


Figure 8.7 Four patches found using PRG for the 300 allocation in test 1. The two right hand patches have merged.

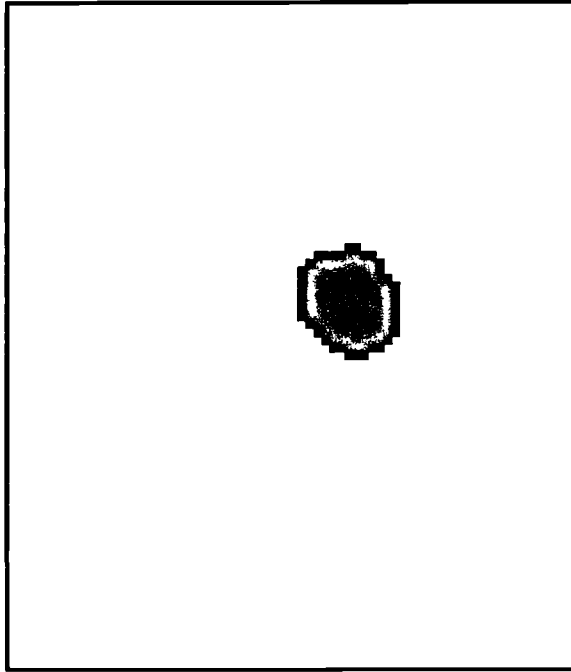


Figure 8.8 Patch found by GAPD for the 150 allocation in test 1

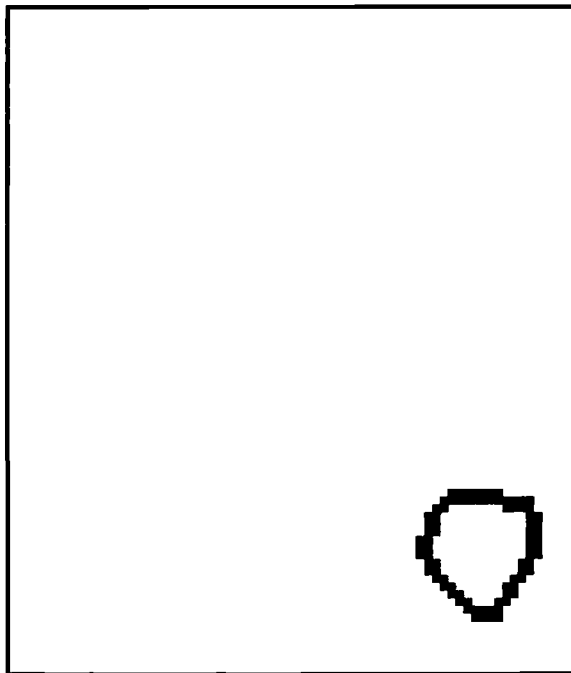


Figure 8.9 Patch found by GAPD for the 200 allocation in test 1

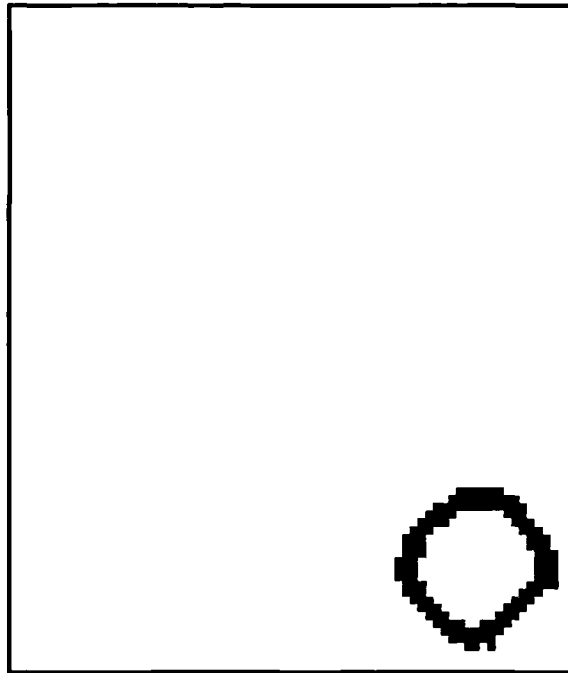


Figure 8.10 Patch found by GAPD for the 300 allocation in test 1

In each run, GAPD stopped because of the MAX_TRIES condition, which means that the solution had converged. The average and maximum utility values of the initial and final populations are shown in table 8.5. The first generation solution (maximum utility) was always poorer than the PRG solution and improved quickly in the early generations overtaking the PRG solutions within 12, 19 and 7 generations. In scenarios (a) and (b) the average solution never exceeded the PRG solution but in scenario (c) the average solution overtook the PRG solution. The relative difference between the average solution and best solution in the final population was greatest for scenario (a) and least for scenario (c). This result may indicate that the solution space for scenario (c) was the least complex. Intuitively this seems reasonable since (c) is the most constrained problem, first because the patch area (300) is larger compared to the space between the barriers, and, second, because the edge effect ($=3$) is larger. In the initial population the average utility was greatest in scenario (a) which has the smallest patches. This is probably because the edge effects are stronger in (b) and (c), with the result that random patches probably have very small core areas.

| Scenario | Generations | Maximum utility | Average utility |
|----------|-------------|-----------------|-----------------|
| a. | 1 | 9070 | 5640 |
| | 12 | 9284 | 7246 |
| | 89 | 9546 | 8582 |
| b. | 1 | 7826 | 3980 |
| | 19 | 9198 | 7130 |
| | 54 | 9448 | 7761 |
| c. | 1 | 9542 | 4285 |
| | 7 | 11216 | 8887 |
| | 71 | 12534 | 11307 |

Table 8.5 GAPD progress in evaluation test 1 showing initial population values, final population values and the generation when maximum utility became better than the PRG solution.

The shape bias parameter values were 0.6, 0.7 and 0.7. These values favour shape score over cell score but not strongly.

8.2.5 Discussion

The IR heuristic was not designed for problems involving configuration and, therefore, GAPD ought to be superior. The fact that GAPD was better than stand-alone PRG is particularly significant because this was a relatively simple problem and an operator might have been expected to find good solutions by inspection. In fact, the stand-alone PRG solutions were good and could have been improved by trying different PDCs. However, these results do indicate the superiority of an unsupervised method. There are a number of advantages of unsupervised methods:

- suitability maps can be misleading because the interpretation of colouring and shading is problematic;
- not all problems can be addressed using suitability maps as will be shown in test 4 ;
- more complex spatial criteria would make choosing PDCs more difficult;
- in multi-patch and multi-objective problems the difficulties would be immense.

The geographical location of the different solutions (Figures 8.2 through 8.10) shows that in scenario (a) the GAPD solution is close to one of the PRG solutions but that for (b) and (c) the GAPD solutions are in a different part of the map. Here, using IR as a guide to selecting seeds for the PRG is misleading, as also is visually assessing the map.

The natural spatial pattern in the suitability map affects the solutions. The PRG patches for (b) are less compact than those for (a) and (c) so there is no simple relationship between compactness and size.

The stand-alone PRG program used a fixed weighting between shape and locational suitability which does not actually optimise the trade-off between these criteria. GAPD is able to adjust the weighting to find better solutions. The spatial criteria in this example are very simple but finding an optimal solution is still difficult and justifies using the genetic algorithm.

The fact that the site of the best patch moves as the allocation level is increased indicates sensitivity to the objective function. If there is uncertainty over the exact area to be allocated, GAPD could be used to identify how significant the uncertainty is. For example, suppose it was originally decided to allocate 150 cells and then later the decision was made to expand the site to 300. If the best 150 cell patch had been allocated it would be impossible to expand it to the best 300 patch. However, if the original plan was for 200, the site could later be expanded because the 200 and 300 patches overlap.

This kind of consideration is important because landuse is dynamic and requirements are always liable to change and also because decisions have to be made on uncertain information which may also change and need to be incorporated. SDSSs facilitate this kind of exploration. The objective function and criteria should be formulated in different ways and the problem solved many times. This exploratory approach requires that the optimisation technique is robust and efficient as well as capable of finding good solutions.

8.3 Evaluation 2 : Single Objective Multi-Patch

8.3.1 Purpose

This example shows the application of GAPD to a multiple-patch, single-objective problem. The spatial criteria are less restrictive than in evaluation test 1 because there is no constraint on the number of patches. Consequently, there are more viable configurations and the decision space is more complex. GAPD operated on two input layers, a suitability layer and a distance layer.

The problem simulates a situation where the configuration of patch networks is significant. Spatial pattern can be interpreted and measured in numerous different ways. Here, the evaluation function uses a straightforward measurement with an obvious physical interpretation: edge-to-edge distance between patches. Inter-patch distance can be measured as a simple geometric distance or on a cost or friction surface. Inter-patch distance is important for species which travel between habitats but have a limited range.

8.3.2 Problem description

8.3.2.1 Objective function

The objective is to design a network of patches which optimises cumulative suitability subject to spatial constraints. The cumulative suitability of a patch is the sum of the suitability values of all cells in the patch. The total size of all patches is constrained.

Four scenarios were devised with different constraints. Two allocation levels (total size constraints), 100 and 200 cells, and two spatial constraints, were used. In two scenarios, patches must have a minimum size of 30 cells. That is, configuration is relevant at the cell level. In the other two, a further spatial constraint was imposed: patches had to be no more than a specified distance, 10 cells, from the nearest of four pre-existing sites (a configuration criterion at the patch network level). There are no criteria relating to patch

shape. The scenario naming convention is:

(a1) - no distance constraint, 100 cell allocation;

(a2) - no distance constraint, 200 cell allocation;

(b1) - 10 cell distance constraint, 100 cell allocation;

(b2) - 10 cell distance constraint, 200 cell allocation.

In a more realistic scenario, the distance constraint would have applied to the spatial relations among all patches old and new. The reason that this was not done was purely pragmatic. It would have been necessary to generate a distance layer for every patch in every alternative solution. Generating distance layers is computationally expensive and, with the computer resources available, the time needed to run the evaluations would have been prohibitive (several days per scenario).

8.3.2.2 Data

The two input data layers were based on the suitability layer used in test 1. Four patches grown by PRG were overlaid on this map to represent the pre-existing sites. Four distance layers were generated, one for each pre-existing site. These layers were overlaid and the minimum value from each layer was taken to make a composite distance layer, MINDIST, in which each cell value is the distance to the nearest pre-existing patch. The pre-existing patches have zero suitability. The new suitability map is shown in figure 8.11 and the MINDIST layer is shown in figure 8.12. The central part of the image is the furthest from the existing patches.

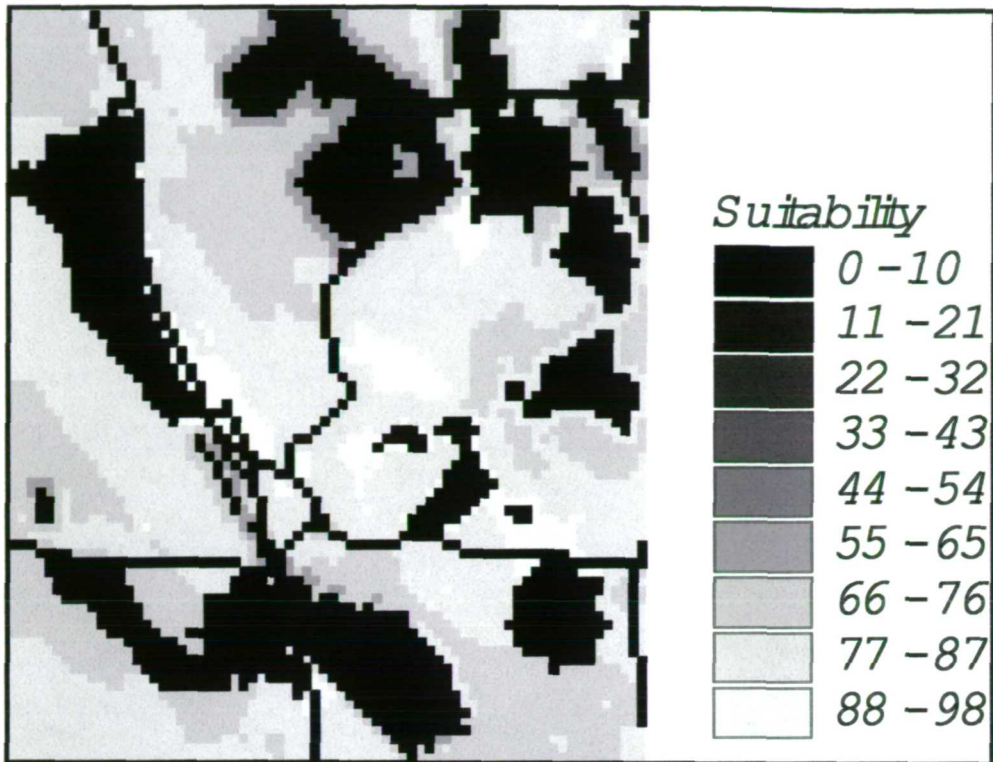


Figure 8.11 Suitability map for evaluation tests 2 and 3

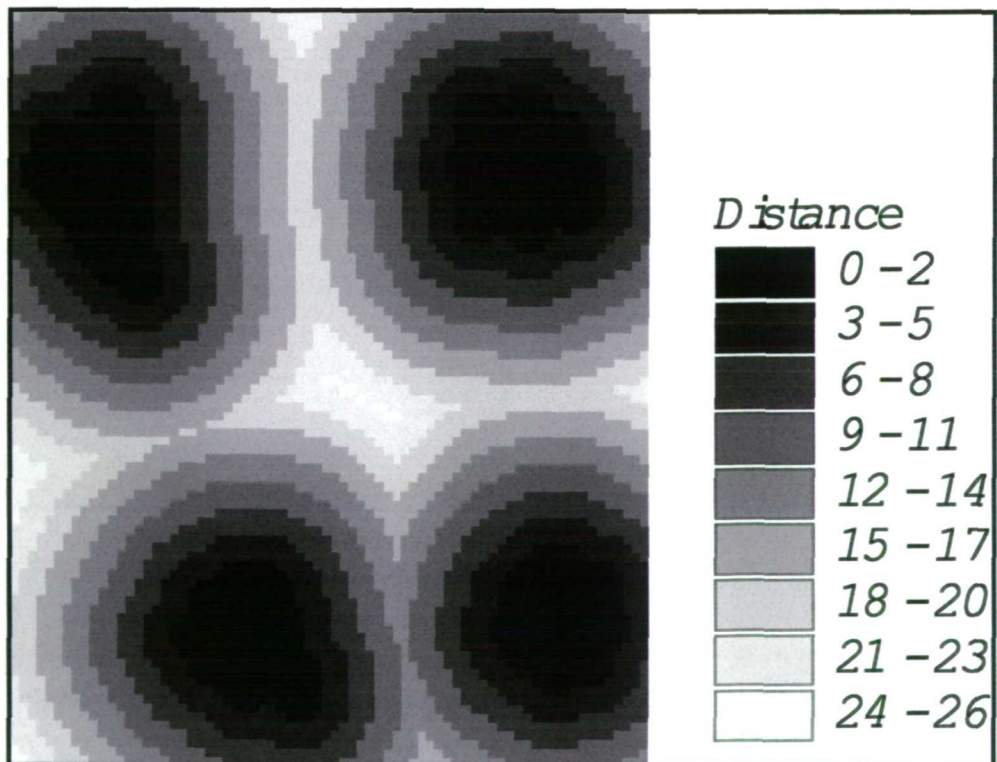


Figure 8.12 Distance layer for evaluation tests 2 and 3

8.3.3 Method

GAPD was implemented with a simplified genetic code. The shape control parameters were coded in the header and therefore applied to all patches. As shape is not a criterion, it is reasonable to expect that the optimal value for the shape bias parameter is zero. Consequently, shape control parameters should be irrelevant. Shortening the genetic code in this way reduces run-time and is unlikely to compromise the quality of the solution. Each PDC in the tail consisted of the seed row and column and the relative size parameters. The PRG algorithm was not adjusted for this example and its operation was as described in Chapter 6. The distance layer was not relevant to the PRG component and was only used by the evaluation functions.

Since each patch must consist of at least 30 cells there can only be at most 6 patches in the 200 allocation. A fixed-length genetic code with 6 patch control parameter sets could have been used. In fact, 10 patches were allowed so as to make the problem more difficult for GAPD. The genetic code is shown in table 8.5a.

| Head | | | | | | | Tail (9 instances) | | | |
|-------|-----------------|---|---|-----------------|---|---|---------------------|--|----------|-----|
| total | shape control 1 | | | shape control 2 | | | shape bias | | location | |
| size | R | A | O | R | A | O | | | row | col |

Table 8.5a Genetic code for evaluation test 2

Evaluation routines

The objective function was evaluated within GAPD as follows. First grow patches under the control of the PDC in the genetic code on the input suitability map. Then assign a unique identifier to each distinct patch using the count_patch function. With this patch layer calculate the area of each patch using extract_count and assign that value to all cells in the patch. Next, eliminate small patches by replacing all values less than 30 by zero. Finally, overlay this layer on the suitability map and for each patch (ie. none zero)cell extract the suitability value from the suitability map and sum all values to give total suitability which equals utility. In scenarios (b1) and (b2), patches which were more

than 10 cells from the nearest pre-existing site were also given zero scores. This was done by taking the second patch map, in which each patch has a unique identifier, and overlaying it on the distance layer. The `extract_min` function returns, for each patch, the minimum of all values extracted from the distance layer. Where this value is greater than ten all cells in the patch are assigned a zero, thus eliminating the patch.

GAPD was run once for each scenario and the control parameters were constant for each run. Population size and number of generations were both set to 100. The genetic operator probabilities were the same as those used in evaluation test 1. The domain of the patch relative size parameter was (int,0,2,1). A zero relative size means a patch does not grow and this allows for solutions with fewer than 10 patches. The chosen relative size domain means that patch sizes will have a bi-modal distribution with smaller patches being about half the size of larger patches. The domain was set arbitrarily but in the event GAPD performed well with this restriction and so no other values were tried. For scenarios (b1) and (b2), the distance constraint was deliberately chosen so that the solutions in scenarios (a1) and (a2) were not valid.

Upper bounds for solutions in scenarios (a1) and (a2) were estimated using a version of the IR heuristic described in test 1. The IR algorithm was modified by changing the stopping condition. All patches that do not satisfy the patch size constraint are eliminated and the size of the remaining patches are added together. When this total equals the allocation level the solution has been reached. The patch size constraint is implemented in IDRISI as follows. Group cells into patches, assign the area of each patch to the cells in the patch using `GROUP` and `AREA` and mask out background cells using `OVERLAY` as before. Then reclass all cells less than 30 (the minimum size) to be zero, and all other cells to be 1. The total area of valid patches is the number of 1 cells and can be found using `AREA` or `HISTO` or by checking the document file. IR does not always find a solution that uses the exact cell allocation.

Results were verified in IDRISI by overlaying the GAPD output maps on the suitability map to extract the total suitability score. The constraints were checked by overlaying

the patches on the MINDIST layer.

8.3.4 Results

Six solutions were generated, IR-a1, IR-a2, G-a1, G-a2, G-b1 and G-b2. The naming convention is IR, for iterative relaxations, or G, for GAPD, followed by the scenario name. Note that IR was only run for scenarios (a1) and (a2). IR could have been run with the distance constrained scenarios by using the distance layer to masking out cells on the suitability map which are more than 10 cells from the existing sites. However, this is only possible because the distance criteria are static. IR would not be suitable in the general case when the distance criteria are dynamic - that is when distance between new patches is constrained.

Of the two IR, solutions only IR-a1 used the exact allocation level. For the 200 allocation no iteration found a solution with exactly 200 cells in valid patches and the closest solution used 223 cells. The iterations of IR are listed in Table 8.6 which shows the threshold, the total allocation and the sizes of qualifying patches. The utility of IR-a2 was adjusted to the equivalent for an exact 200 cell allocation by using the average cell value. The original figure is in brackets in Table 8.7.

The patches are illustrated in Figures 8.13 through 8.18 and the results are tabulated in Table 8.7. There are similarities and differences in the patch patterns. The IR-a2 patches encompass the IR-a1 patches, but the same does not apply for GAPD. Some of the cells in the G-a1 solution are not in the G-a2 solution. The IR method builds on existing solutions as the allocation level is increased, but GAPD can switch to completely new patterns.

| Threshold rank | Total number of qualifying cells | Sizes of qualifying patches |
|----------------|----------------------------------|-----------------------------|
| 100 | 42 | 42 |
| 300 | 146 | 46 47 53 |
| 200 | 47 | 47 |
| 250 | 134 | 34 47 53 |
| 220 | 87 | 47 40 |
| 230 | 97 | 47 50 |
| 235 | 100-----STOP | 47 53 |
| 400 | 229 | 50 33 48 53 45 |
| 350 | 178 | 46 47 53 32 |
| 375 | 190 | 46 47 53 44 |
| 385 | 223 | 31 46 48 53 45 |
| 380 | 192 | 46 48 53 45 |
| 382 | 192 | 46 48 53 45 |
| 383 | 192 | 46 48 53 45 |
| 384 | 223-----STOP | 31 46 48 53 45 |

Table 8.6 IR iterations for the 100 and 200 allocations in evaluation test 2

| | Utility | Distance to furthest site | Minimum cell value | Total allocation |
|-------|--------------|---------------------------|--------------------|------------------|
| IR-a1 | 9042 | 17 | 88 | 100 |
| IR-a2 | 17852(19906) | 20 | 86 | 223 |
| G-a1 | 9118 | 13 | 84 | 100 |
| G-a2 | 17832 | 13 | 82 | 200 |
| G-b1 | 8858 | 3 | 78 | 100 |
| G-b2 | 17586 | 6 | 80 | 200 |

Table 8.7 The six solutions in evaluation test 2

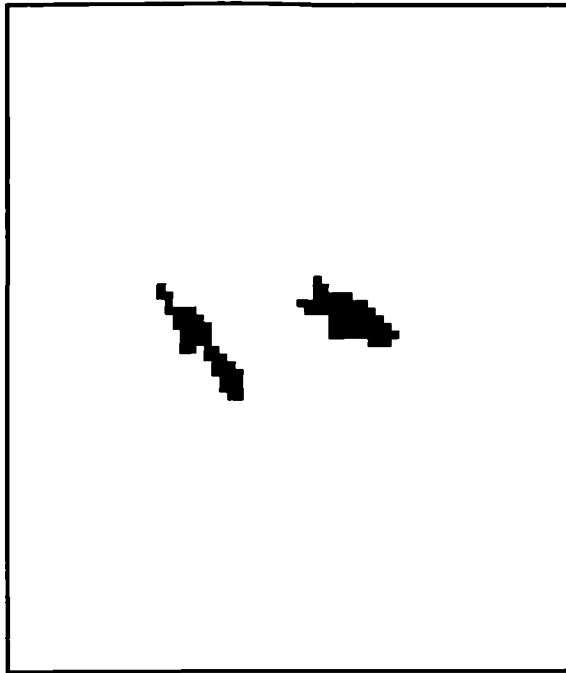


Figure 8.13 IR-a1: patches found by IR for the 100 allocation in evaluation test 2

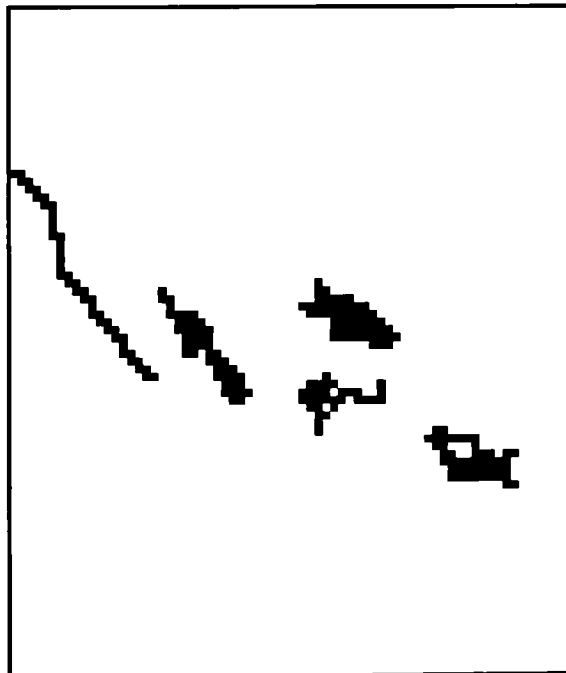


Figure 8.14 IR-a2: patches found by IR for the 200 allocation in evaluation test 2

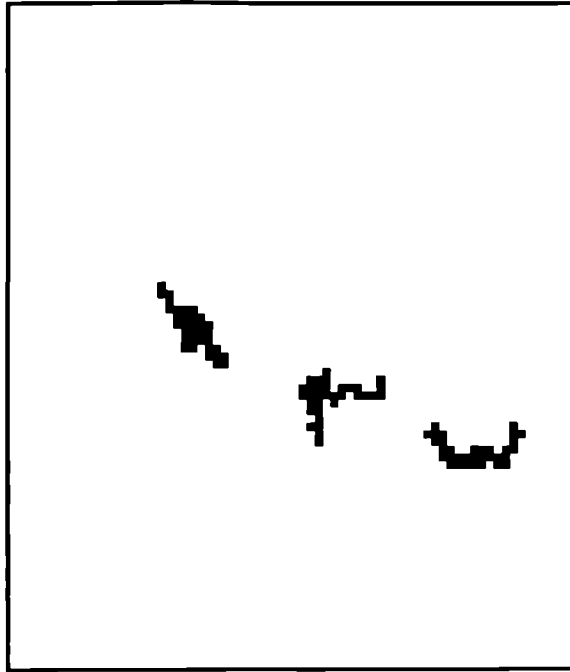


Figure 8.15 G-a1: patches found by GAPD for the 100 allocation in evaluation test 2, with no distance constraint

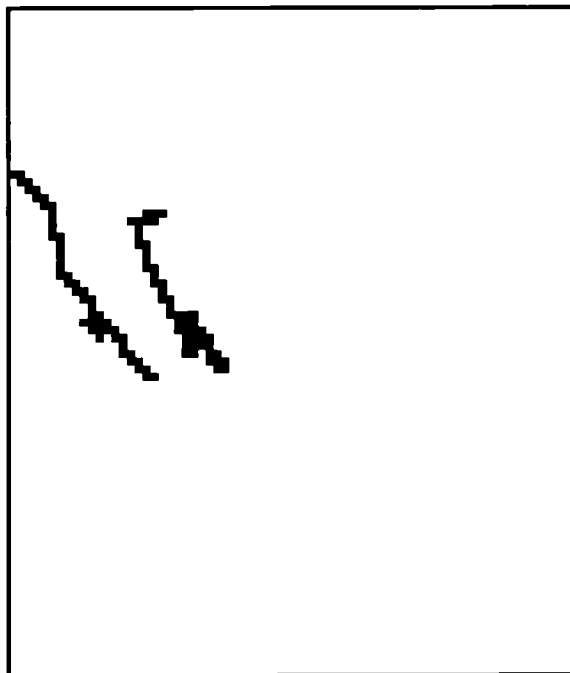


Figure 8.16 G-b1: patches found by GAPD for the 100 allocation in evaluation test 2, with a distance constraint



Figure 8.17 G-a2: patches found by GAPD for the 200 allocation in evaluation test 2, with no distance constraint

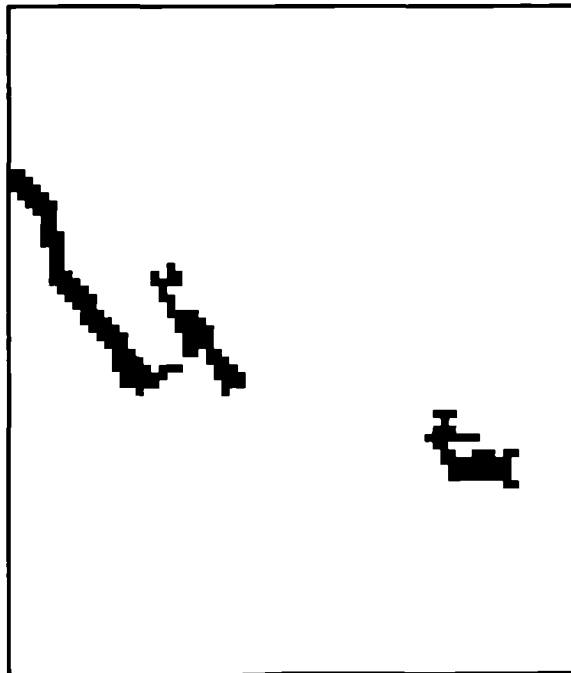


Figure 8.18 G-b2: patches found by GAPD for the 200 allocation in evaluation test 2, with a distance constraint

The GAPD solutions G-b1 and G-b2 have lost the central patches (in G-a1 and G-a2) which did not satisfy the distance constraint. In both cases GAPD has generated a completely new patch. One of the patches in G-a1 disappears in G-b1 and reappears in G-b2.

In scenarios without the distance constraint the GAPD solution is better than the IR solution at the 100 level and almost as good at the 200 level. As expected, the scenario (b-) GAPD solutions have lower utility than their scenario (a-) counterparts. The distance constraint was imposed by a simple modification to the evaluation routine.

The difference between the GAPD and IR algorithms is reflected in the minimum cell values shown in table 8.7. The GAPD solutions included lower valued cells than the IR solutions and yet, at the 100 level, the GAPD solution was better than the IR. The fact that in scenarios (b1) and (b2) the GAPD patches included poorer cells and have poorer utility than in scenarios (a1) and (a2) is an indication that GAPD responds to the change in problem constraints.

In each GAPD result the shape bias parameter was 0 giving 100% bias to cell suitability against shape suitability. The shape parameters were, therefore, irrelevant and GAPD succeeded by optimising the seed locations and the relative sizes of each patch.

8.3.5 Discussion

This illustration demonstrates that GAPD finds good solutions compared to a promising heuristic. IR is a promising heuristic when there is no distance constraint, because patch configuration is not a strong factor, so it is a good benchmark test. Intuitively, one would expect IR to find good solutions. However, the algorithm does not guarantee to find optimal solutions as can be seen by considering a one-dimensional example. In the string 9991999077757770 the best string of length 7 is 9991999, which scores 55, but IR would find 7775777, which scores 47.

GAPD is able to compromise on cell suitability because patch composition, which is a dynamic criterion, is evaluated at an appropriate level. Low-scoring cells can actually improve the patch utility by connecting higher-scoring cells together. The IR heuristic is unable to exploit this possibility.

GAPD is shown to be flexible because it can be easily adapted to include additional spatial constraints. Only a simple change to the utility function was needed to add the distance constraint. As configuration becomes a more important factor, IR becomes less suitable but GAPD is equally applicable to both situations.

The same kind of problem exploration as described in evaluation test 1 could be done here. The way the patch configurations respond to different criteria may be very important if the criteria are uncertain. In such circumstances, the flexibility of GAPD is an important quality.

8.4 Evaluation 3 : Multi-objective single-patch problem

8.4.1 Purpose

This example illustrates the application of GAPD to a multi-objective single-patch problem. The hypothetical problem is to design a single patch which satisfies two objectives. The first objective is to maximise cumulative core suitability and the second is to minimise the isolation from four existing patches. This test combines elements of evaluation tests 1 and 2.

Patch composition and configuration both contribute to the first objective but for the second objective only patch configuration is relevant. The objectives are complementary since one site will satisfy both objectives, unlike evaluation test 5 (section 8.6) where the objectives are conflicting because each cell can only be allocated to one out of a number of alternative uses. However, there will be a trade-off between the two objectives because a compact shape will maximise core suitability whereas a long thin shape will

minimise isolation. The two objectives will pull the solution in opposite directions because the best shape for the first objective will be a compact shape whereas a more linear shape with extensions towards each existing site would satisfy the second objective. The actual optimal shape will depend on the suitability map and the relative weights of the two objectives.

This example also illustrates the use of GAPD with multiple input layers because it uses distance and suitability layers.

8.4.2 Problem description

8.4.2.1 Objective function

Multi-objective problems present the same kind of difficulties as multi-criteria problems. The key issue is the weighting of objectives in relation to each other. One approach is to rank the objectives and solve for the most important objective first. An alternative approach is to combine the different objectives into a single objective function and there are a number of mathematical procedures for doing this (Tamiz, 1996).

A multi-objective spatial planning problem is described by Eastman *et al.* in a workbook on GIS and decision-making produced by the IDRISI project for the United Nations Institute for Training and Research (UNITAR) (Eastman *et al.*, 1993). They use a weighted combination method analogous to the weighted summation method used in MCDM. Eastman *et al.* describe the method's implementation in the IDRISI GIS. The method described in the UNITAR workbook for complementary objectives is to prepare suitability maps for each objective in turn and then combine all the maps into a single suitability map for all objectives.

In this illustration, the objectives are complementary because the solution is a single patch. However, the UNITAR workbook method is not appropriate. With respect to the isolation objective, individual cells have no intrinsic suitability. Isolation is a dynamic

criterion so utility can only be evaluated for the patch as a whole. Consequently, it makes no sense to try and combine suitability maps to assign a single suitability value to each cell. This is not a problem for GAPD because it handles dynamic criteria at the appropriate level by explicitly calculating isolation for each patch configuration. The principle on which GAPD works is that each alternative (in this case a single patch) is evaluated and its fitness used to drive the evolution of a solution. The fitness of each alternative is a weighted combination of the utility evaluated with respect to each objective independently. Although the suitability of an individual cell can not be derived from the individual layers, the utility of a patch can be evaluated easily by extracting values from data layers.

The isolation objective is to minimise the isolation of the new patch with respect to four existing sites. As with other spatial attributes, isolation can have different interpretations. To be used as an objective function, isolation must be defined more strictly. In this case, the isolation of a patch is the maximum distance to any other patch and the distance between two patches is defined as the minimum edge-to-edge distance. The isolation of a network of patches is the maximum isolation of each individual patch. The objective is to maximise the reduction in the isolation of the network resulting from the introduction of a new patch.

The attribute measurement for the core area objective is cumulative core suitability, the same as in evaluation test 1. The attribute measurement for the distance objective is the maximum of the shortest edge-to-edge distance from the new patch to each of the existing sites. The unified utility function is a weighted sum of the individual utility functions. In order to unify the two objectives, individual utility scores have to be normalised in some way. This is problematical and a method was adopted which, although not rigorous, was considered reasonable and could be used in a real problem.

Normalisation

The evaluated scores for each single objective are normalised by scaling the attributes to a common scale from 0, meaning worst, to 1000, meaning best. In the UNITAR

workbook, all suitability maps are normalised by scaling all contributing factor maps to a common scale. This is done using the IDRISI STRETCH operation which applies a linear conversion to all attribute values such that the highest attribute value is given the maximum utility score, the lowest attribute value is given the minimum utility score and intermediate values are interpolated.

The problem in this case is that the best and worst solutions are not known. Upper and lower bounds can, however, be estimated using one of two approaches. The first way, the **analytical method**, is to calculate theoretical values for the best and worst scores. For core area the worst score is zero (i.e. when the patch is all edge) and the best score would be a circle with each cell in the core having maximum cell suitability. The number of cells in the core will vary with the size of the patch but can easily be found by growing a circle (see Chapter 7). For isolation, the best score is when distance is zero to all patches and the worst score can be taken to be the maximum cell value on each distance layer. These values give the attribute values that correspond to the zero and maximum utility scores. The utility for actual attribute values are found by interpolation. An alternative method, the **estimation method**, is to estimate maximum and minimum utility values for each objective separately. Using GAPD, the worst utility scores can be estimated by running a single generation. Effectively this is simply a number of random solutions and the worst utility of the random population can be taken as the worst possible solution. The utility of the best possible solution is estimated by running GAPD to convergence. Then, for each objective, the utility is normalised by interpolation.

The estimation method may be preferable for several reasons. The theoretical values may not be attainable in practice and the search spaces of each objective have different characteristics. One objective may be inherently easier to meet and it then becomes problematic to assign weights to each objective. Finally, theoretical values may be difficult to determine.

In the tests the analytical method was used. Given an objective for which the attribute value V_{max} gives maximum utility and the attribute value V_{min} gives minimum utility, the

utility, U , of an attribute value, V , is given by equation 8.1 below.

$$U = \frac{(V - V_{\min})}{(V_{\max} - V_{\min})} * 1000 \quad \text{Equation 8.1}$$

For N objectives, the combined utility of alternative a , U_a , is given in equation 8.2 below where U_{ai} is the utility of alternative a against objective i and W_i is the weight of objective i .

$$U_a = \frac{\sum_{i=1}^N U_{ai} * W_i}{\sum_{i=1}^N W_i} \quad \text{Equation 8.2}$$

8.4.2.2 Data

The data layers were the same as those in evaluation test 2. The distance layer, MINDIST, was a composite of four layers and the suitability map is from evaluation test 1 with 4 patches superimposed to represent the pre-existing patches. The suitability map is shown in figure 8.11 and the distance map is shown in figure 8.12. In each map the highest values are shown in lighter shading.

8.4.3 Method

The number of patches in the genetic code was fixed at 1. The header of the genetic code contains the size. The seed, shape control and shape bias are in the tail. Because the tail is only one patch code, the relative weight is redundant. This structure is shown in figure 8.7a. GAPD was amended to read normalisation and weighting information from the control file. Normalisation parameters are the maximum and minimum values

for each attribute for each objective. Weights are integers between 0 and 10.

| | | | | | | | | | |
|-------|----------|-----|-----------------|---|---|-----------------|---|---|------------|
| Head | Tail | | | | | | | | |
| total | location | | shape control 1 | | | shape control 2 | | | shape bias |
| size | row | col | R | A | O | R | A | O | |

Table 8.7a Genetic code for evaluation test 3

Evaluation routines

The objective function was evaluated within GAPD as follows. First grow a patch under the control of the PDC in the genetic code on the input suitability map. Second, shrink the patch by 1 cell to produce a new raster with the core patch. Overlay this layer on the suitability map and for each cell in the core extract the suitability value from the suitability map using the `extract_all` function. The total of all suitability values is the cumulative core suitability which is the utility for the patch core objective. The utility score for the isolation objective is calculated next. Overlay the original patch on the MINDIST layer and extract the minimum value using `extract_min`. This value is the isolation of the patch. Utility is a measure of the improvement in isolation so this value is subtracted from the original isolation (55). Both utilities are normalised on a scale of 0 to 1000 using equation 8.1 with the minimum and maximum single objective values supplied in the control file. The final utility is calculated using equation 8.2 with the relative objective weights read from the control file.

In each run, the population and number of generations were set to 200 giving a maximum of 40000 generations. The program ran on a DEC Alpha machine in about 2 hours. The genetic operator probabilities were the same as the previous evaluation tests. The PRG shape control parameters were unrestricted, the size parameter was set to 150 and the edge effect was 1 cell.

GAPD produces a single utility value for the best patch found. This value was checked using IDRISI. The IDRISI `EXTRACT` function extracts values for a feature layer

overlaid on an attribute layer. The attribute values were used to calculate the individual and combined utility scores.

To illustrate the effect of the compromise between the two objectives on patch shape, several scenarios were run with different relative weights for the two objectives. The weights used were, 10-0, 7-3, 5-5, 3-7, 1-9 and 0-10 where the first figure is the weight for the core area objective and the second figure is for the isolation objective. In the following discussion, each scenario is identified by the weights used. The 1-9 scenario was added because the original 5 scenarios did not show the transition in patch shape as the bias changed from core to isolation. Intuitively, one would expect more compact shapes when core area is more important and less compact shapes as distance becomes more important. In a real situation it is unlikely that the relative weights could be assigned exactly and, therefore, some sort of exploration of the problem would be advantageous using a decision support system framework.

8.4.4 Results

The analytical method was used to normalise the individual objective scores. The two objectives were considered independently. For the core area objective, the upper bound was set at 10000, based on a target area of 150 cells having a core of around 100 cells, each having suitability not greater than 100. The largest distance between any two of the four existing patches is 55 cells so the maximum improvement is 55. This would occur if the new patch had a zero distance from all existing sites. For both objectives the lower bound is zero. Normalising in this way is an attempt to make the two objective scores comparable. It is not a rigorous method and will not guarantee equal weight to each objective. Assigning weights is problematical in itself. Therefore, the method should be used in a qualitative exploratory way by solving the multi-objective problem several times, with different weights, to investigate how the solution varies. In this illustration the weights were varied to see how the shape of the patch changes.

The results are shown in Tables 8.8, 8.9 and 8.10. Table 8.8 shows the PRG parameters

for each solution. Table 8.9 shows the attribute values and utility values of the patches. The results are presented in two rows for each test. The first row shows the attribute measurements for the whole patch and the second row shows the attribute measurements for the core. The total cell score for the core divided by 10000 gives the utility with respect to core area. The utility with respect to isolation is found by taking the largest distance to the four existing patches, subtracting it from 55 to get the improvement, and then dividing by 55 and multiplying by 1000 to normalise the score. For example, in row 1 the largest distance is 24 giving an improvement of 31 which converts to $(31/55)$ times 1000, which makes a score of 563. The final column gives the combined score for both objectives. Table 8.10 lists the individual utility scores for each test together with an illustration of the patch shape. This table makes it easy to see how the patch changes in response to the change in weights.

| Objective Weights | Parameters | | | | | | | | | |
|-------------------|------------|---------------|-----|-----|----|-----|-----|------|----|-----------|
| | Shape Bias | Shape control | | | | | | Seed | | Rel. Size |
| 10-0 | 0.3 | 9 | 2.4 | 100 | 10 | 2.7 | 3 | 39 | 44 | 1 |
| 7-3 | 0.5 | 7 | 1.1 | 156 | 10 | 2.5 | 44 | 38 | 42 | 1 |
| 5-5 | 0.5 | 6 | 1.2 | 135 | 8 | 0.9 | 67 | 39 | 41 | 1 |
| 3-7 | 0.4 | 9 | 0.8 | 135 | 8 | 5.6 | 105 | 38 | 41 | 1 |
| 1-9 | 1.0 | 2 | 4.9 | 19 | 1 | 6.5 | 45 | 44 | 40 | 1 |
| 0-10 | 0.9 | 0 | 4.2 | 93 | 0 | 5.6 | 128 | 42 | 38 | 1 |

Table 8.8 PRG parameters (PDCs) for the GAPD solutions in evaluation test 3

| Weight | Score for | Sum of cells | Dist. 1 | Dist. 2 | Dist. 3 | Dist. 4 | Isolation Change | Single Utilities | Total Utility |
|--------|-----------|--------------|---------|---------|---------|---------|------------------|------------------|---------------|
| 10-0 | Isolation | - | 23 | 24 | 11 | 21 | 31 | 563 | 961 |
| | Core | 9614 | - | - | - | - | - | 961 | |
| 7-3 | Isolation | - | 23 | 23 | 11 | 22 | 32 | 581 | 843 |
| | Core | 9556 | - | - | - | - | - | 956 | |
| 5-5 | Isolation | - | 22 | 22 | 13 | 22 | 33 | 600 | 769 |
| | Core | 9378 | - | - | - | - | - | 938 | |
| 3-7 | Isolation | - | 22 | 22 | 11 | 22 | 33 | 600 | 699 |
| | Core | 9290 | - | - | - | - | - | 929 | |
| 1-9 | Isolation | - | 16 | 15 | 15 | 15 | 39 | 709 | 683 |
| | Core | 4498 | - | - | - | - | - | 450 | |
| 0-10 | Isolation | - | 12 | 6 | 9 | 14 | 41 | 745 | 745 |
| | Core | 2304 | - | - | - | - | - | 230 | |

Table 8.9 Patch attribute and utility values in evaluation test 3

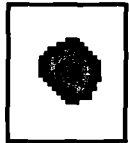

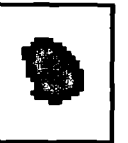

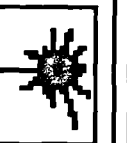

| | 10-0 | 7-3 | 5-5 | 3-7 | 1-9 | 0-10 |
|-------------------|---|---|---|--|---|---|
| Isolation Utility | 563 | 581 | 600 | 600 | 709 | 745 |
| Core Utility | 961 | 956 | 938 | 929 | 450 | 230 |
| Patch |  |  |  |  |  |  |

Table 8.10 Change in patch characteristics with changing objective weights in evaluation test 3 (The first five images are exaggerated 1.5 times relative to the last)

As table 8.8 shows, all the seeds are very similar so the location of the best patch does not change much however the objectives are weighted. This can be seen in Figures 8.19, 8.20 and 8.21 which show the locations of patches 10-0, 5-5 and 0-10 respectively. The fact that all the locations are similar is interesting because it means that, even in a single objective problem to optimise core area, the patch location would not change. Therefore, the addition of a second objective does not compromise the location. As regards location, there is no conflict between the objectives.

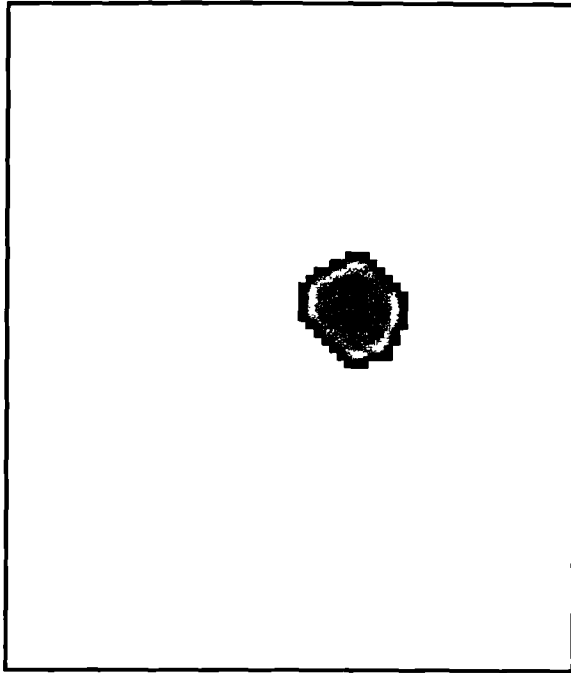


Figure 8.19 Optimal patch for the core objective (weight 10-0) in evaluation test 3

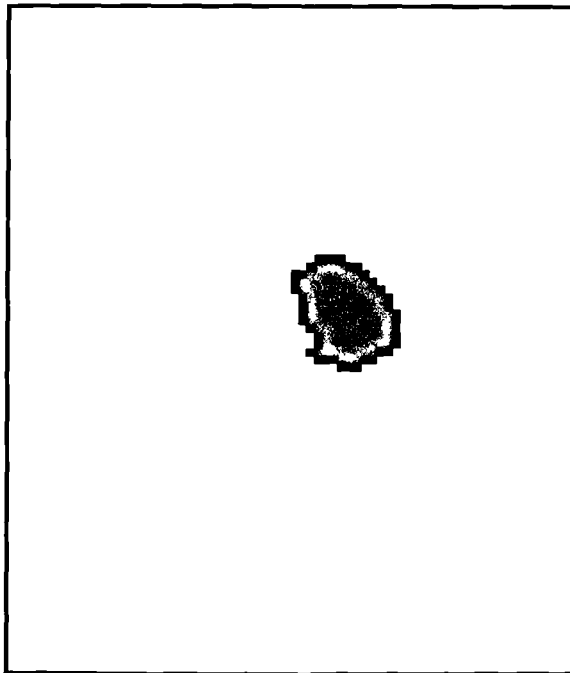


Figure 8.20 Optimal patch for two equally weighted objectives (weight 5-5) in evaluation test 3

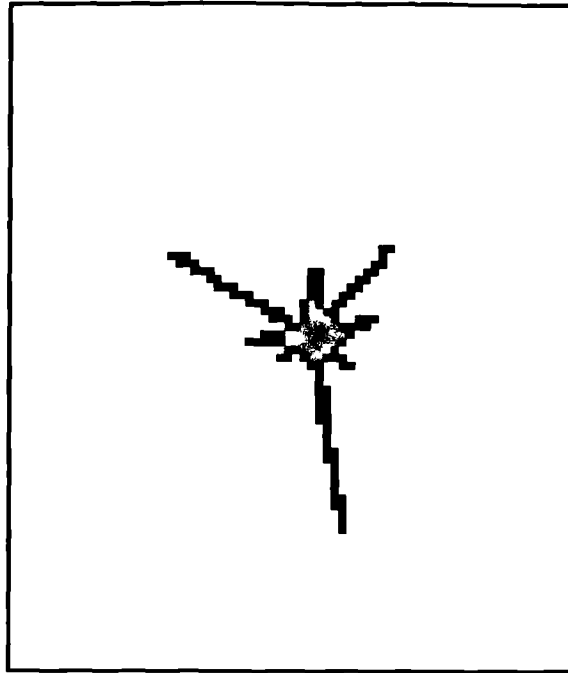


Figure 8.21 Optimal patch for the isolation objective (weight 0-10) in evaluation test 3

Conflict between the objectives does affect patch shape, as can be seen in Table 8.10. The patch shape changes as the relative objective weights are changed. However, the change in shape is small until the objective weights are heavily in favour of the isolation objective suggesting that there is a natural bias towards compact shapes. Only when the weights are strongly in favour of the isolation objective does the shape deviate from a compact shape. This may be a result of the way utility is defined in this particular case. The change in utility by making a compact shape spread out towards an existing patch is relatively small, and may be zero, because it will only affect the distance to one patch. Thus, a large weight is needed to make the trade-off of distance against loss of core suitability worthwhile.

An examination of the shape bias PDC parameter, column 2 in Table 8.8, reveals that there is a step up in value from around 0.4 for the first 4 solutions to around 0.9 for the last 2 solutions. High values of shape bias mean that the PRG algorithm weights the contribution of a cell to the patch shape more highly than its intrinsic score. What this means is that when isolation is the more important objective, patch shape becomes more

important than patch composition. This is expected but what is more revealing is the low values for shape weight when core suitability is relatively important. Here the value for the shape bias is influenced by the spatial pattern in the underlying suitability map. Low values for shape bias mean that high scoring cells are tightly clustered and naturally form compact shapes. This means that, with the given datasets, optimising for core is an easier problem than optimising for isolation because a wider range of PRG parameters will grow good shapes. Considerations of this sort illustrate the difficulty in establishing suitable normalising functions. If the objectives are given equal weights GAPD will converge on a solution biased towards the easier objective.

As the weights are adjusted to change the bias from 10-0 through 5-5 to 0-10 the utilities with respect to the single objectives change. Note that when one of the relative weights is zero, there is effectively only a single objective. The response of utility to the changing weights is clearly shown in Table 8.10. The utility with respect to core area decreases and that with respect to distances increases. This is to be expected and is reflected in the shapes of the different patches. When core area is the single objective patches are most compact and when isolation is the only objective the patches are least compact.

8.4.5 Discussion

GAPD appears to be better at optimising for core than isolation. The isolation only patch (0-10) does not look optimal as it has more axes than it needs and consequently they are shorter. This may be because, as explained above, the isolation objective is harder than the core objective in this particular case. An interesting experiment would be to develop a variation of the PRG component that would use the distance layers as well as the suitability layer. The shape scoring algorithm could be adjusted so that cells with low distances values would be preferred to those with high distances. Different shape-growing heuristics may help to guide the search. Such modifications have to be balanced against the ideal that GAPD should not rely on domain-specific heuristics.

The results from this evaluation test show firstly that GAPD can generate different patch

configurations in response to different objectives and that it is sensitive to the relative weighting of objectives. The test also illustrates how multiple objectives can be introduced by changing the evaluation component without affecting the GAPD driver component or the PRG region-growing component. In these respects, GAPD can be judged a success.

The flexibility of GAPD is important because it can be used to explore a problem, in this case by varying the relative weights of the two objectives to see how the solution changes.

The ability to operate with attribute maps instead of suitability maps is a very significant feature of GAPD. As this and evaluation test 2 show, suitability maps are not always meaningful. Isolation is a patch attribute, not a cell attribute, and can only be measured at the patch level.

8.5 Evaluation 4 : Multi-patch problem

8.5.1 Purpose

For this evaluation, the multi-patch problem is to design patch configurations when the optimal number of patches is unknown. GAPD's evaluation function uses goal programming techniques instead of the MCDM methods used in the other evaluation tests. There are no static criteria and hence no suitability map and the PRG component is adjusted to operate on raw data layers.

A number of variations on a single problem were run, partly to show the development of the full-function GAPD and partly to compare it with the prototype. GAPD was run in fixed-patch mode as an incremental development between single and variable-patch versions.

The objective is to allocate patches with a specified composition and a constraint on the

total amount of land to be allocated. Each patch must incorporate different landcover types and the utility of a patch is evaluated by comparing its composition with an ideal. When a patch matches the ideal it has optimal utility and making the patch bigger will not improve the solution. In some cases several small patches will be better than one large patch. The patches could represent animal territories which must incorporate different attributes as, for example, certain birds such as the Dartford Warbler (Catchpole and Phillips, 1992) require different types of vegetation for nesting, foraging and so on. Alternatively, the territories could represent catchments for facilities or sales territories that must incorporate diverse socio-economic categories.

8.5.2 Problem description

8.5.2.1 Objective function

The problem is not to find the maximum number of ideal territories but to optimise the number and quality of all territories. There will be a trade-off between solutions with fewer better territories against those with more but poorer territories.

The utility of a patch network is a function of the utility of each individual patch. In turn, patch utility is a measure of how well composition matches the composition of an "ideal" patch. Configuration is made a stronger factor by stipulating that the core area must satisfy the composition criteria. This simulates a situation where the edge of the patch functions as a buffer and is not exploited for its resources.

Patch composition is the amount of each different landcover type in the patch. The ideal composition is specified as a vector whose elements are the required area, in cells, of each landcover type.

A number of scenarios were run: in some the object is to find a fixed number of patches and in others it is to find the best arrangement of patches when the number of patches and their individual configurations and compositions are unknown. The second situation

could represent an attempt to find the maximum population that a landscape could support given the territorial needs of each breeding group. A theoretical upper limit for the number of territories can be calculated by looking at the landscape composition and dividing the total area of each landcover type by the requirements of each territory. However, the configuration of the landscape may mean that this figure can not be reached.

Goal programming methods were adopted to define a utility function. In goal programming each alternative is evaluated using some measure of distance from the goal as the basis of the utility function (Tamiz, 1996). Goal programming is applicable because each patch must satisfy a goal. The mathematics are similar to ideal point analysis with the objective being to minimise distance from the goal or ideal point. In this case, there is a maximum distance in each dimension so the utility function can be turned around to measure the difference between the maximum distance from the ideal point and the actual distance from the ideal point. For the patch composition goal, the worst score against a criterion is zero cells. The distance of a point from the ideal is the sum of squares of the distances in each dimension. The distances in each dimension are normalised to a uniform scale by using relative distance. Further, because the worst solution is known (i.e. when actual area is zero) distance from the worst solution can be used and the objective function becomes a maximisation instead of a minimisation. Thus, if there are N requirements, the utility, U , of a solution with areas A_i is given by equation 8.3 where R_i is the requirement for landcover i and A_i is the area of landcover i . If A_i is greater than R_i , the distance is defined to be one (from the worst) or zero (from the ideal) in that dimension.

$$Max \quad U = \sum_{i=1}^N \left(\frac{A_i}{R_i} \right) \quad \text{Equation 8.3}$$

When the solution is constrained to be a single patch the above method evaluates the solution. Extension of the method to multiple patches is simple if the number of patches is fixed. Each patch can be treated as a dimension in a goal programming evaluation and

the scores for each patch can be added together to get a single score. The utility is found by summing over all patches and all requirements. Utility, U , is given by equation 8.4 where A_{ip} is the area of type i in patch p and R_{ip} is the requirement of type i in patch p . If the requirements are the same for all patches R_i , can be substituted for R_{ip} .

$$\text{Max } U = \sum_{i=1}^N \sum_{p=1}^P \left(\frac{A_{ip}}{R_{ip}} \right) \quad \text{Equation 8.4}$$

Extension to multiple patches is difficult if the number of patches is unknown. In that case, the number of dimensions is also unknown. In fact, the number of dimensions depends on the number of patches and will vary between different alternatives. Consequently, differences from the ideal are not comparable. The problem can be avoided if, as in this hypothetical scenario, the objective has been expressed as a maximisation. The utility of an alternative can then be taken to be the sum of the utilities of each individual patch. This is mathematically equivalent to the goal programming or ideal point methods.

However, there is still a real problem with the variable multi-patch case and that is the nature of the trade-off between number of patches and patch quality. Using the distance metrics idea, the quality of a patch is a linear function of its composition. If edge effects are ignored, a large patch could be divided into two halves without affecting the overall utility. This is not always realistic and, in general, the relationship will not be linear. A non-linear function can be interpreted as meaning that, up to a point, it is better to improve patch quality; but as quality gets nearer the ideal, then, at some point, the payoff gets less and it is better to have more patches. In this example problem, because patch configuration is significant, dividing a large patch into smaller patches may reduce total utility because smaller patches have a smaller proportion of core area.

8.5.2.2 Data

The base map was taken from the IDRISI sample datasets (Eastman, 1993). Three

woodland landcover types on the WORCWEST landuse map were reclassified to types 1, 2, 3 and the rest of the image to type 4 (the PRG component requires that the relevant categories be numbered consecutively starting at 1). The GAPD control file specifies the number of categories and the requirements in each category. In this case, there are three categories and the program interprets any landcover category greater than 3 or less than 1 as making no contribution to utility. Therefore, category 4 is automatically assigned a zero requirement. The requirement vector for the ideal territory is {6,23,30,0}.

8.5.3 Method

For this application, the PRG operates on a base map of attributes rather than a suitability map. Consequently, a new cell scoring algorithm is employed. The intrinsic suitability of a cell cannot be taken from a base map, nor can it be calculated from the cell attributes alone. Patch composition is a dynamic criterion so cell suitability changes as a patch grows and is different for each patch to which the cell potentially belongs. For a given patch, the suitability of a candidate cell is defined to be 1 if the requirement for that category is unsatisfied, and 0 if it is satisfied. All classes which are not required are lumped together and have a requirement of zero, so each cell in those categories automatically has zero suitability. Shape suitability is calculated in the usual way and the PRG shape control parameters are unchanged. The trade-off between shape and cell suitability is controlled by the shape bias parameter. The resolved cell suitability is a weighted average of cell and shape suitability.

For testing in multi-patch mode, the GAPD driver and PRG used a multi-patch genetic code. The genetic code header includes the total size and the shape control parameters. The tail contains the seed, shape bias and relative size parameters for each patch. The code is illustrated in table 8.10a. This was done to keep run-time as short as possible. Ideally, each patch should have its own shape parameters but that would have made for a very long genetic code and would have increased run-time beyond acceptable limits (several days). The longest run time was about 48 hours on a DEC Alpha workstation. Because all the patches must be compact, it is reasonable to give each patch the same

shape and the restriction is not likely to compromise the solution too much. In variable multi-patch mode the genetic code was a fixed length with 10 PDCs. The prototype GAPD was used for single-patch mode.

| Head | | | | | | | Tail (2 instances) | | | |
|---------------|-----------------|---|---|-----------------|---|---|---------------------|-----|------------|------------------|
| total size | shape control 1 | | | shape control 2 | | | location | | shape bias | relative size |
| | R | A | O | R | A | O | row | col | | |

Table 8.10a Genetic code for evaluation test 4

When the number of patches is fixed, the number of PDCs equals the number of patches and the minimum value of the relative size parameter must be greater than zero. For variable multi-patch problems, that is when the number of patches is not known in advance, the minimum value for relative size is zero. A zero relative size means the patch will have zero size; even though there is a code for it, the patch does not grow. Another important difference between fixed and variable multi-patch GAPD is that in the former, each PRG code generates a single distinct patch, but in the latter, contiguous patches merge into a single patch. Merging patches allows more complex shapes to be generated. There are no other differences between the fixed and variable varieties of multi-patch GAPD.

In each run all utility scores were normalised to a scale of 0 to 1000. The number of generations was 300 and the population size was 300 in each run. The genetic operator probabilities were the same as in other tests (table 8.2). The PDC shape control parameters were unrestricted. The size constraint was varied in each run as explained below.

Evaluation routines

The evaluation in GAPD worked as follows. Grow the patches on the raw data map using PRG under the control of the genetic code. Shrink each patch by 1 cell to generate a core layer. Overlay the core layer on the classified map and for each of the required landcover types extract the number of cells falling with each patch. Maintain a table of

values for each patch. Calculate the ratio of actual cells to required cells in each category, using a value of 1 when the requirement is exceeded. Convert each fraction to a requirement score on a scale of 0 to 1000. Take the average of the requirement scores to generate the patch score. Total the scores for all patches to get the final utility of the solution.

All the results were checked using the SHRINK utility to shrink patches to the core and using the IDRISI EXTRACT function to count the number of cells in each landcover category in each patch. The cell counts were compared to the requirements vector, converted to distance metrics and then to utility using equation 8.4.

8.5.4 Results

Preliminaries

More significant changes were made to the PRG component for this evaluation test and so an experiment was made to check that GAPD would function correctly with the revised PRG. The preliminary experiment was done by altering the data in such a way that there was a solution which satisfied all demands. A circular patch with the right composition (i.e. it is an "ideal" territory) was overlaid on a sub-section of the original data image. The circular patch was generated using PRG, first to grow a single patch of 100 cells of type 3, then another patch of 30 cells of type 2 and finally two patches of 6 cells each of type 1. The smaller circles were overlaid on top of the larger circles. Two patches of type 1 were used to make the solution harder to find. If there were only one patch of type 1 then any seed within the patch would grow to an optimal solution. Two patches allows for the possibility of a trial patch including only one of the small circular patches.

Ten trials were run using the "planted" solution map. In each trial, the population was set to 50. The results are presented in Table 8.11 and show that single patch GAPD found the optimal solution each time but the number of generations taken to find the optimal solution varied from 4 to 49. GAPD continued running after reaching the

optimum because the average utility was still improving. The best solution in the initial population and the number of generations are shown in Table 8.11. The initial population is a random sample of all possible answers. In one run, the initial population got a score of 989. Each initial population can be considered as a series of generations without adaptation and the numbers clearly show the difference between the adaptive search, where each generation is an improvement, and a random search where each generation is independent of the previous ones.

The method of planting a solution can be used to verify that GAPD is capable of finding a solution but does not prove that it will always find an optimal solution. The planted solution may be easy to find and, thus, may greatly distort the solution space. Variables such as the size of the population and the probability of the genetic operators may affect performance. The size of the initial population may be too small causing premature convergence to a sub-optimal solution. If this is the case, then it is reasonable to expect the solution to improve if GAPD is run again with a larger population. Therefore, some idea of the quality of the solution could be gained by running GAPD several times with increasing population sizes. When the solution stops improving, it is reasonable to accept that solution as the best possible.

| Population = 50, Map with a "planted" solution | | |
|--|----------------|-----------------------|
| Initial solution | Final solution | Number of generations |
| 800 | 1000 | 8 |
| 914 | 1000 | 14 |
| 989 | 1000 | 4 |
| 843 | 1000 | 6 |
| 682 | 1000 | 19 |
| 805 | 1000 | 16 |
| 672 | 1000 | 49 |
| 617 | 1000 | 32 |
| 708 | 1000 | 9 |
| 789 | 1000 | 36 |

Table 8.11 Preliminary trial of GAPD with modified PRG for evaluation test 4

Evaluation results

Three versions of GAPD, single-patch, fixed multi-patch and variable multi-patch, were compared in six scenarios, making a total of 18 trials. The cell allocation levels were 100, 200, 300, 400, 500 and 600 cells. In single-patch and fixed multi-patch mode the number of patches were 1 through 6 respectively. The prototype GAPD was run iteratively, generating a single patch of 100 cells in each iteration. After the first iteration, the first patch was overlaid on the base map to mask out that area and prevent subsequent iterations from using the same cells. The fixed and variable multi-patch GAPD allocated all cells in a single step.

Table 8.12 lists the features of the three methods and highlights the significant differences. Single-patch GAPD, run iteratively, generates patches of equal size; fixed multi-patch GAPD generates a predetermined number of patches of varying size; and variable multi-patch GAPD generates an indeterminate number of patches of varying size. Single-patch GAPD is the least flexible and allows no trade-off between the quality

of individual patches to maximise overall utility. By generating all patches simultaneously, the multi-patch versions of GAPD do allow such a trade-off. In addition, variable multi-patch GAPD allows a trade-off between the quality of individual patches and the number of patches. Variable multi-patch GAPD can generate more patches of lower quality to maximise overall quality.

| Versions of GAPD | Method | Features |
|----------------------|-------------|--|
| Single-patch | Iterative | Fixed Number of patches All patches the same size No trade off in patch quality between patches No trade off between patch quality and number of patches |
| Fixed multi-patch | Single step | Fixed number of patches Patches different sizes Possible trade off in patch quality between patches No trade off between patch quality and number of patches |
| Variable multi-patch | Single step | Variable number of patches Patches different sizes Possible trade off in patch quality between patches Possible trade off between patch quality and number of patches |

Table 8.12 Comparison of different versions of GAPD in evaluation test 4

The results from all trials are summarised in Table 8.13. For the three lowest allocation levels, variable multi-patch GAPD found better solutions than single-patch but the situation was reversed for the three higher levels. At all levels, variable multi-patch generated more patches than the other two methods and this may be an indication that the algorithm is biased towards more rather than fewer patches. One might expect single-patch GAPD to find poorer solutions in successive iterations. The fact that this did not happen shows that GAPD does not always find global optima. However, there is a tendency for later iterations to find poorer patches and that is an indication that GAPD is finding good solutions.

| Method | | Allocation level | | | | | |
|---------------------------|------------|------------------|-----|-----|-----|-----|-----|
| | | 100 | 200 | 300 | 400 | 500 | 600 |
| Single-patch GAPD | utility | 822 | 813 | 823 | 803 | 783 | 765 |
| | no.patches | 1 | 2 | 3 | 4 | 4 | 6 |
| Fixed multi-patch GAPD | utility | 833 | 819 | 800 | 768 | 736 | 732 |
| | no.patches | 1 | 2 | 3 | 4 | 5 | 6 |
| Variable multi-patch GAPD | utility | 840 | 873 | 869 | 794 | 767 | 729 |
| | no.patches | 2 | 3 | 5 | 5 | 6 | 8 |

Table 8.13 Comparison of results for different allocation levels in evaluation test 4

The results for allocation level 600 are presented in detail in table 8.14. The best solution was found using six iterations of single-patch GAPD. The six patches had a combined score of 4577 compared to 4435 and 3333 for the other methods. The full function GAPD found better solutions when run in variable multi-patch mode than when the number of patches was fixed. The base map is shown in figure 8.22 and the outlines of the patches found by the different methods are shown in figures 8.23, 8.24 and 8.25. The figures show that the patch locations are similar in each case.

| Solution found using variable multi-patch GAPD | | | | | | | | | |
|--|------------|-----------|---------------------------------------|----|----|-------------------------------|------|-----|-------------|
| Patch | Patch Area | Core Area | Composition of core by landcover type | | | Score for each landcover type | | | Total Score |
| | | | 1 | 2 | 3 | 1 | 2 | 3 | |
| 1 | 96 | 63 | 5 | 21 | 16 | 417 | 913 | 533 | 621 |
| 2 | 70 | 46 | 0 | 22 | 20 | 0 | 956 | 667 | 541 |
| 3 | 71 | 45 | 2 | 22 | 17 | 167 | 956 | 567 | 563 |
| 4 | 76 | 50 | 3 | 20 | 10 | 250 | 869 | 333 | 484 |
| 5 | 71 | 45 | 1 | 18 | 24 | 83 | 782 | 800 | 555 |
| 6 | 73 | 45 | 1 | 23 | 11 | 83 | 1000 | 366 | 483 |

| 7 | 71 | 47 | 1 | 23 | 21 | 83 | 1000 | 700 | 594 |
|---|------------|-----------|---------------------------------------|----|----|-------------------------------|------|------|-------------|
| 8 | 72 | 46 | 5 | 16 | 20 | 416 | 695 | 667 | 592 |
| Map | 600 | 387 | | | | | | | 4435 |
| Solution found using fixed multi-patch GAPD | | | | | | | | | |
| Patch | Patch Area | Core Area | Composition of core by landcover type | | | Score for each landcover type | | | Total Score |
| | | | 1 | 2 | 3 | 1 | 2 | 3 | |
| 1 | 107 | 75 | 8 | 25 | 11 | 666 | 1000 | 366 | 677 |
| 2 | 101 | 69 | 6 | 25 | 22 | 500 | 1000 | 733 | 744 |
| 3 | 109 | 74 | 9 | 22 | 22 | 750 | 956 | 733 | 813 |
| 4 | 95 | 65 | 5 | 20 | 31 | 416 | 869 | 1000 | 761 |
| 5 | 89 | 60 | 3 | 33 | 22 | 250 | 1000 | 733 | 661 |
| 6 | 99 | 64 | 4 | 23 | 26 | 333 | 1000 | 866 | 733 |
| Map | 600 | 407 | | | | | | | 4392 |
| Solution found using single-patch GAPD | | | | | | | | | |
| Patch | Area | Core | Composition of core by landcover type | | | Score for each landcover type | | | Total Score |
| | | | 1 | 2 | 3 | 1 | 2 | 3 | |
| 1 | 100 | 69 | 6 | 23 | 29 | 500 | 1000 | 956 | 822 |
| 2 | 100 | 72 | 6 | 21 | 30 | 500 | 913 | 1000 | 804 |
| 3 | 100 | 68 | 10 | 23 | 21 | 833 | 1000 | 700 | 844 |
| 4 | 100 | 70 | 4 | 22 | 27 | 333 | 956 | 900 | 729 |
| 5 | 100 | 70 | 6 | 21 | 21 | 500 | 913 | 700 | 704 |
| 6 | 100 | 71 | 1 | 23 | 28 | 83 | 1000 | 933 | 672 |
| Map | 600 | 420 | | | | | | | 4577 |

Table 8.14 Comparison of different versions of GAPD for the 600 allocation level in evaluation test 4

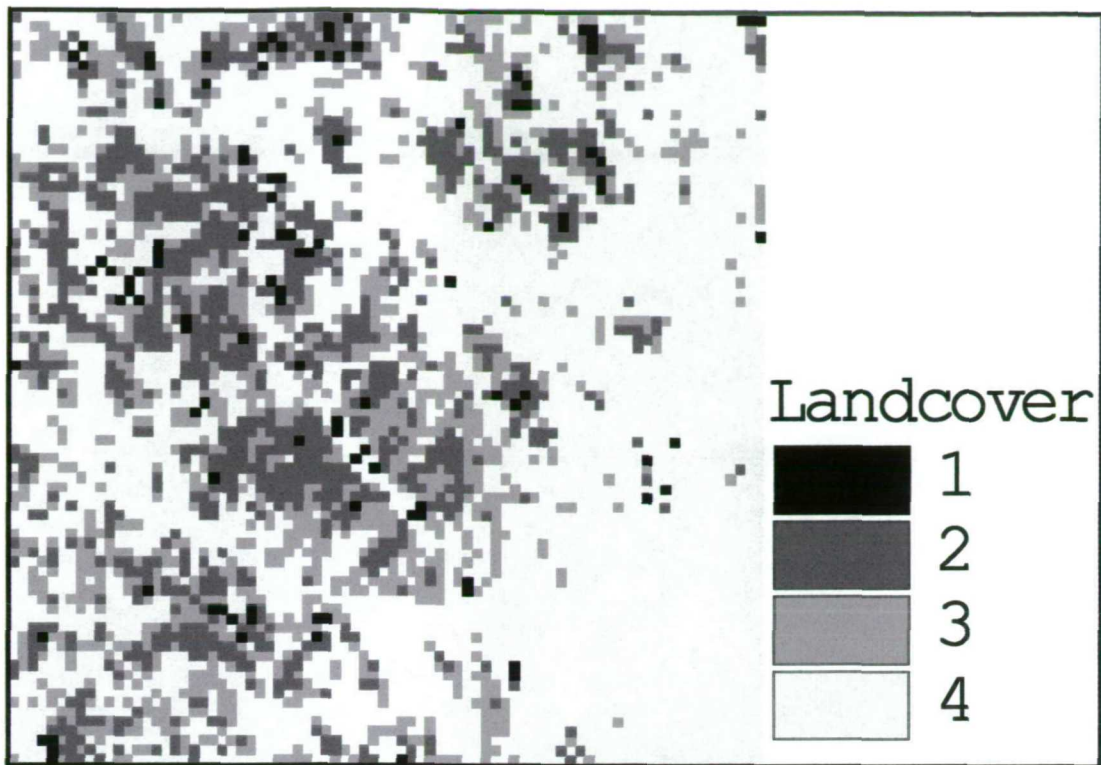


Figure 8.22 Data layer showing landuse categories in evaluation test 4

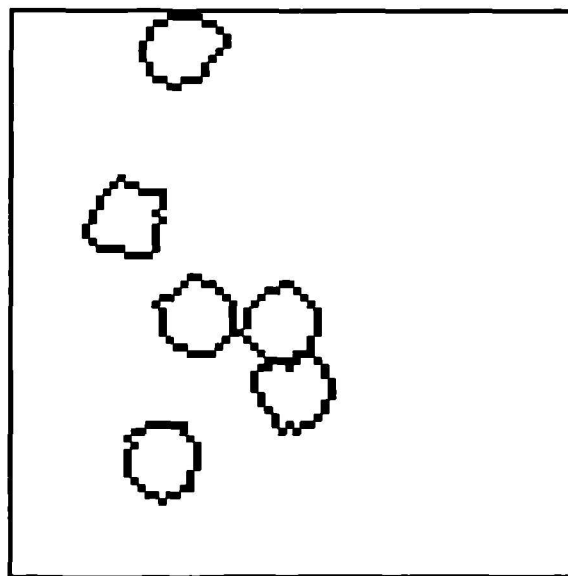


Figure 8.23 Six patches found by iterative application of single patch GAPD in evaluation test 4

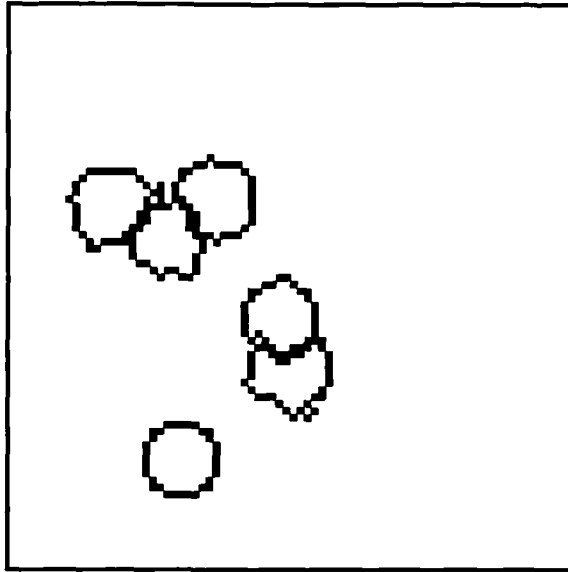


Figure 8.24 Six patches found by fixed multi-patch GAPD in evaluation test 4

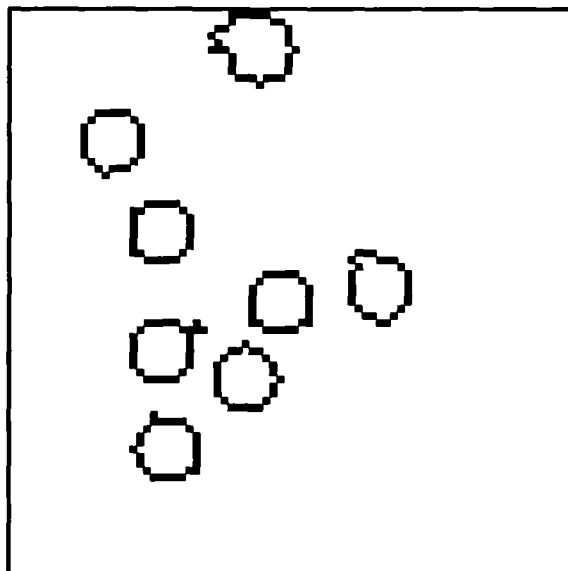


Figure 8.25 Eight patches found by variable multi-patch GAPD in evaluation test 4

8.5.5 Discussion

It is impossible to draw definite conclusions from these few trials because there are many unknowns. A possible explanation why single-patch GAPD sometimes found better solutions than fixed and variable multi-patch GAPD is that the search space is much

smaller and less complex. It is important to realise that the search space is defined by the genetic coding and not by the real world. Even though all three methods were applied to the same problem, they did not use the same representation and, therefore, did not have the same search space. The added flexibility of fixed and variable multi-patch GAPD did not compensate for the added complexity of the search space. This result highlights the importance of considering the appropriate application of GAPD in complex problems. In the hypothetical simulations only one objective was considered. Other questions might be what is the optimal size of a territory or how many territories are there? The sensitivity of the solution to uncertainty in the definition of what constitutes an optimal territory is also important.

It would be interesting to experiment with complex utility functions. In the utility functions used above, there is a perfect trade-off between each criterion and each patch. Therefore, one good patch can compensate for a bad patch just as a good score for one criterion can compensate for a bad score for a different criterion. This is a necessary condition for the iterative, single-patch approach to be valid. The iterative approach assumes that, by finding the best patch at each iteration, the high quality of the early patches will compensate for the low quality of subsequent patches. If this assumption is not valid then the iterative application of GAPD in single-patch mode is not the best method. Variable multi-patch GAPD is better because it can handle a non-linear utility function. Because variable multi-patch GAPD finds patches simultaneously, it can, for example, find a solution of two moderate patches which is better than a solution of one good and one bad patch. The flexibility of variable multi-patch GAPD makes it a more promising tool than single-patch GAPD for this type of exploratory investigation.

Another advantage of multi-patch modes over single-patch mode is that the patches can be of different sizes within limits set in the control file. This allows solutions in which the improvement in utility by making one patch larger more than compensates for the decrease in utility by making another patch smaller. The minimum and maximum values for the patch relative size parameter control the range of patch sizes. The ratio of the maximum and minimum values sets the ratio of the largest to the smallest patch area and

the absolute difference controls the number of possible intermediate sizes. For example, the domains {int,1,2,1} and {int,100,200,1} both have the same ratio of 2:1, but in the first case, there are only 2 different sizes while, in the second, there are more than 100. If the minimum value is zero, then for calculating ratios, the effective value is 1. The zero value allows the number of patches, as well as the size of patches, to vary.

8.6 Evaluation 5 : Multi-objective multi-patch problem

8.6.1 Purpose

This evaluation demonstrates the application of GAPD to a multi-objective, multi-patch planning problem with conflicting objectives: to allocate two zones to two different uses. The objectives conflict because each patch can be allocated to only one use. A feature of this test is that there are two suitability maps.

The demonstration problem is a variation on a case study described in the United Nations Institute for Training and Research (UNITAR) workbook on decision-making, "Explorations in GIS Technology " by Eastman *et al.* of the IDRISI project (Eastman *et al.*, 1993). The demonstration problem uses the same data layers and suitability maps as the case study but with a modified planning objective that includes criteria relating to patch configuration.

The data layers in this example are much bigger than those in the previous four evaluation tests; they are large enough for this test to assess the efficiency of the genetic algorithm. Because of the size of the datasets, GAPD could not run on a PC but was run on a DEC Alpha workstation.

8.6.2 Problem description

8.6.2.1 Objective function

In the UNITAR workbook, Eastman *et al.* illustrate multi-objective decision-making using case studies. As an example of multi-objective planning with conflicting objectives, they describe a planning project in Kathmandu, Nepal, to control the expansion of the carpet industry. The original problem is to develop a planning map which allocates 1500 hectares (16,666 cells) to the carpet industry and 6000 hectares (66,666 cells) to agriculture. The two planning objectives conflict because each unit of land is allocated to one use only.

The workbook describes the factors and criteria used in making the decision and the methodology used. The first step is to generate two suitability maps, one for the carpet industry and one for agriculture, using multi-criteria evaluation. Sufficient cells are then independently assigned to each landuse to satisfy both objectives: allocate the best 16,666 cells on the carpet suitability map to the carpet industry and the best 66,666 cells on the agriculture suitability map to agriculture. Because the objectives conflict some cells will have been allocated to both uses, which is not allowed, and these cells are re-allocated to one or other use. This leaves a shortfall that is made up by allocating more cells, leading to more conflicts, and so on, until both objectives are satisfied simultaneously.

The workbook case study is an aspatial optimisation problem because there are no criteria relating to the configuration of land units in the final planning map. Once a suitability value has been assigned to each cell, the allocation of cells to a landuse can be determined without reference to their spatial location. The method works because all criteria are static. In general, it is likely that some sort of spatial criteria would be relevant in this sort of planning exercise. Configuration is made a criterion by introducing some hypothetical spatial criteria. The new planning objective is to locate two planning zones, one for the carpet industry and one for agriculture; each zone (of

500 cells) should be compact to minimise the impact on the natural environment. As in evaluation test 1, compactness is enforced by using cumulative core suitability as the objective function for each patch. The two patches must be distinct, that is they cannot overlap, but otherwise there is no constraint on their proximity to each other, nor is proximity a factor. The utility function is cumulative core suitability with an edge effect of 1 cell.

Proximity could be included by making a simple change to the utility function. A routine to measure the distance between pairs of patches as they are generated, as was suggested in evaluation test 3, would be easy to program. However, such distance calculations are very computer intensive and would not have been feasible with the resources available.

Each objective is given equal weight and the combined utility is the weighted sum of the individual utilities, as in evaluation test 3. The individual utility scores are normalised by solving for each objective individually, using single-patch GAPD (i.e. using the estimation method described in 8.4.2.1). The two single-objective solutions are used to assess the degree of conflict in the objectives.

8.6.2.2 Data

Two suitability maps, AGSUIT and CARPSUIT, were created following the steps described in the workbook and using the supplied geographic datasets. The maps are illustrated in Figures 8.26 and 8.27 respectively and have 490 rows and 761 columns. High suitability is indicated by lighter shading and the large dark area is the urban area of Kathmandu. The constraints and factors are similar for both suitability maps and, therefore, the suitability maps themselves are similar. This means that cells which are highly suitable for one use are usually highly suitable for the other. Consequently, there is conflict between the two objectives because a cell can only be allocated to either the carpet industry or agriculture.

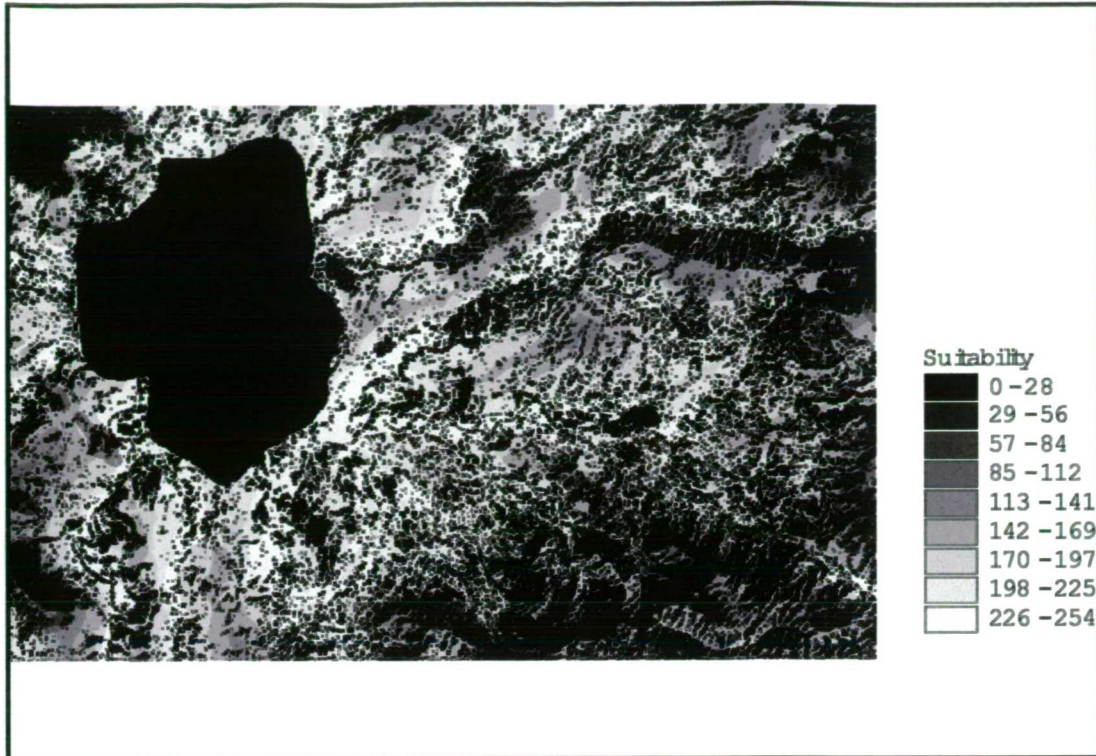


Figure 8.26 Suitability map for agriculture in evaluation test 5

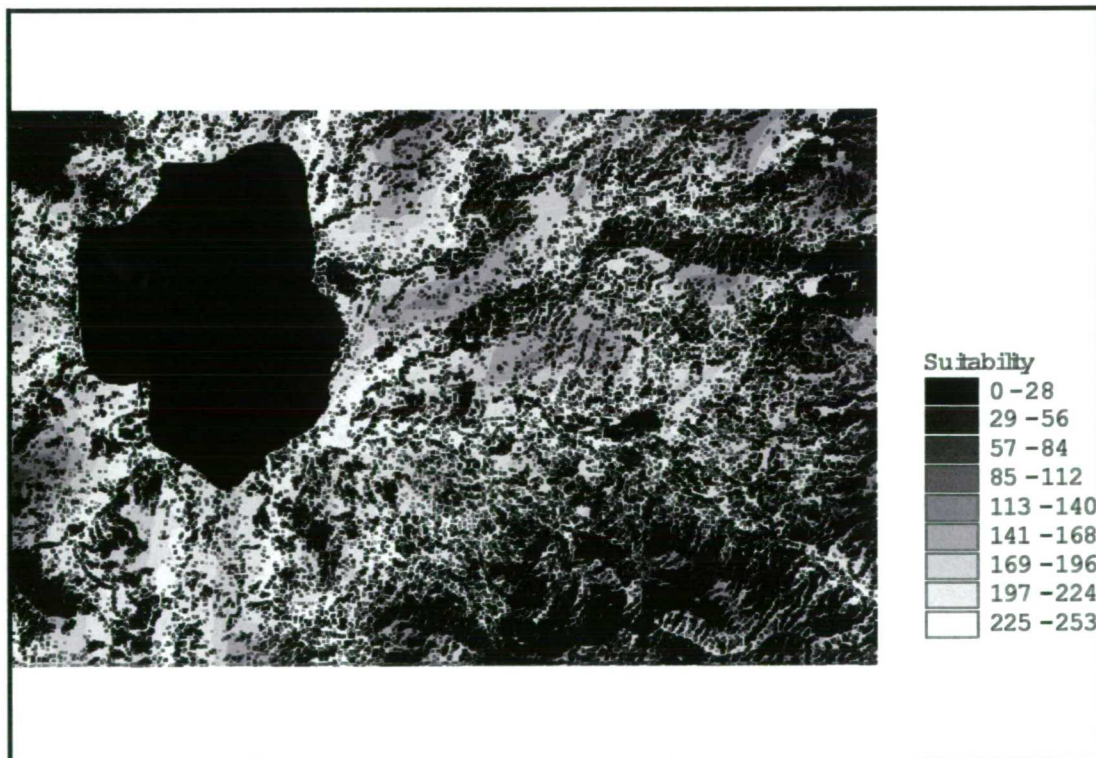


Figure 8.27 Suitability map for the carpet industry in evaluation test 5

The degree of conflict between the objectives can be assessed by solving the single-objective problems individually and comparing the results. The solution to the aspatial problem is found by ranking cells and then selecting sufficient of the highest-ranked cells to make up the total area to be allocated. When both objectives are solved individually, the conflict between the two solutions is found by cross-tabulating the two planning maps.

The procedure (described in the workbook) is as follows. Each suitability map is ranked to create two rank images CARPRANK and AGRANK. The conflict for some chosen number of cells N is found by selecting the first N cells in each rank image and creating two new binary images, AGBEST and CARPBEST, in which 1 represents a high ranking cell and 0 is a low ranking cell. These two images are cross-tabulated to give an output image CONFLICT which has four categories of cells. Cells valued 4 are 1's in both images, cells valued 1 are 0's in both images and cells valued 2 or 3 are 1 in one image and 0 in the other. Cells valued 4 are, therefore, high ranking cells for both objectives and therefore, the number of 4's is an indication of the conflict. The degree of conflict for four different allocation levels is shown in Table 8.15. For a small allocation of only 100 cells to each use, there is no conflict; but for 500 cells and above, there is significant conflict.

| | | | | |
|------------------|-----|-----|------|------|
| Allocation level | 100 | 500 | 1000 | 1500 |
| Conflict | 0 | 77 | 293 | 338 |

Table 8.15 Conflict between aspatial objectives at different allocation levels in evaluation test 5

8.6.3 Method

Three GAPD components, the genetic driver, the evaluation function and the PRG function were altered. To find two patches, a fixed-length genetic code with two PDCs was used. Unlike evaluation tests 2 and 4, where the shape control parameters were in the header, here the shape control parameters are in the tail so that each patch has its own shape. This is feasible because there are only two patches, hence the genetic code

is still relatively short. Almost the whole PDC is duplicated for each patch and the tail consists of location, relative size and shape with only the total size in the header. The genetic code is shown schematically in Table 8.16. The shape control parameters are the normal PDC parameters.

| Head | Tail (2 instances) | | | | | | | | | |
|------------|--------------------|-----|-----------------|---|---|-----------------|---|---|------------|---------------|
| total size | location | | shape control 1 | | | shape control 2 | | | shape bias | relative size |
| | row | col | R | A | O | R | A | O | | |

Table 8.16 Genetic code for the multi-objective GAPD in evaluation test 5

The longer the genetic code, the larger the search space, so there are performance implications in having parameters in the tail or the header. Ideally, all parameters except total size should be in the tail, as is the case here, but as the number of patches increases so does run-time. The run-time with this genetic code was around 36 hours. The advantage of the longer genetic code is that each patch has its own shape and, therefore, the quality of solutions should be better. With the longer genetic code it is possible to have different spatial criteria for each patch, for example, one patch can be compact and another can be elongated.

Another difference from the previous evaluation tests is that each patch is a different type. There are two patches, one of each type, so the type can be inferred from the position of the PDC in the genetic string. In general, the type would have to be coded explicitly to allow for the possibility of different numbers of patches of each type. Here the convention is that the first set of patch codes are for the carpet patch and the second set for the agriculture patch.

The GAPD driver component was altered to manipulate the new genetic code. GAPD was altered to read two suitability maps and the cell-scoring rules in the PRG were altered so that cell suitability was taken from the appropriate layer depending on the patch type. Similarly, the evaluation routine calculated the cumulative core suitability of a patch by extracting cell values from the appropriate suitability layer.

Evaluation routines

Within GAPD the utility functions were evaluated as follows. Grow the carpet patch using the carpet suitability map and the first set of patch control parameters. Shrink this patch by 1 cell. Overlay the core onto the carpet suitability map and extract the sum of all suitability values using the `extract_sum` function. Normalise the score using equation 8.1 and the minimum and maximum carpet patch values from the control file. This computes the utility for the carpet patch. Repeat the process for the agriculture patch using the second patch code and appropriate values. The multi-objective utility is the average of the two individual patch utilities as in equation 2 with equal weights for both objectives.

The population and number of generations were both set to 300. The genetic operators and their probabilities were the same as in each of the other evaluation tests. The total allocation level was specified as 1000, with equal relative size for both patches, and the edge effect was set to 1.

Equal patch size is achieved by setting the maximum and minimum values in the relative size parameter domain to a single high value. This test showed up a bug in the program code. In general, any patch can have zero relative size and so it is possible for all patches in a genetic string to have zero relative size. To avoid a zero divide when this happens, the program adds 1 to the sum of the patch relative sizes. This flaw means that the program cannot guarantee patches of exactly equal size. This problem was mitigated by setting relative size to a single high value, so the significance of the extra one is reduced and both patches will be very close in size. The problem could have been avoided altogether by coding a simple `IF..THEN` statement that only adds the one if the relative size sums to zero - but this simple expedient was overlooked at the time.

The solutions were checked using IDRISI. Each patch generated by GAPD was contracted using the `SHRINK` program. The IDRISI `EXTRACT` function was used to get values for the patch and core compositions.

8.6.4 Results

Four methods were applied to the single and multi-objective problems. In the following discussion the generated patches are referred to by names reflecting the method used and the objective to be met. The methods were, **Iterative Relaxation (IR)**, **Single Objective GAPD (SOG)**, **Multi-objective GAPD (MOG)** and **Hierarchical GAPD (HIG)**. The patch names are IR-AG, IR-CARP, etc. for the agriculture and carpet industry objectives respectively. SOG and HIG were implemented using single-patch GAPD and MOG was implemented using fixed multi-patch GAPD.

The attributes of all patches are shown in Table 8.17. Tables 8.18 and 8.19 show the utilities and graphical images of each patch. Maps of the patches are shown in Figures 8.32 through 8.37. The four methods were used:

- to find benchmark figures to assess multi-objective GAPD,
- to find upper bounds for normalising the utility functions in the multi-objective case,
- to assess the degree of conflict in the objectives, and
- to explore the spatial properties of the suitability maps.

| Method & Objective | | Values | | | | |
|--------------------|-------|--------|--------------|-----------------|-----------------|--------------------|
| | | Area | Sum of cells | Max. cell Value | Min. cell Value | Average cell value |
| IR Agriculture | Patch | 500 | 126289 | 254 | 251 | 253 |
| | Core | 227 | 57445 | 254 | 251 | 253 |
| IR Carpet | Patch | 500 | 121904 | 253 | 240 | 244 |
| | Core | 347 | 84981 | 253 | 240 | 244 |
| SOG Agriculture | Patch | 500 | 118754 | 254 | 0 | 238 |
| | Core | 432 | 104633 | 254 | 0 | 242 |
| SOG Carpet | Patch | 500 | 113947 | 253 | 0 | 228 |
| | Core | 430 | 100457 | 253 | 0 | 234 |
| MOG Agriculture | Patch | 497 | 112833 | 254 | 0 | 227 |
| | Core | 429 | 98820 | 254 | 0 | 230 |
| MOG Carpet | Patch | 503 | 110868 | 252 | 0 | 220 |
| | Core | 439 | 97619 | 251 | 0 | 222 |
| HIG Agriculture | Patch | 500 | 118754 | 254 | 0 | 238 |
| | Core | 432 | 104633 | 254 | 0 | 242 |
| HIG Carpet | Patch | 500 | 111032 | 244 | 0 | 222 |
| | Core | 430 | 97680 | 244 | 0 | 227 |

Table 8.17 Patch attributes for all solutions in evaluation test 5

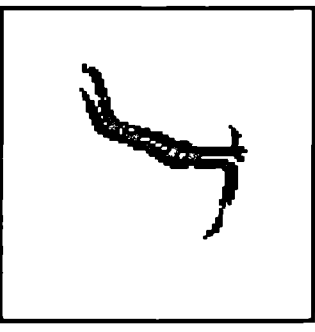
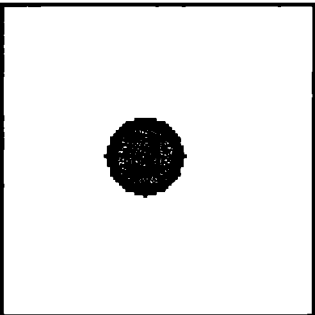
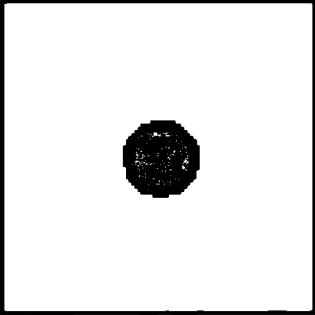
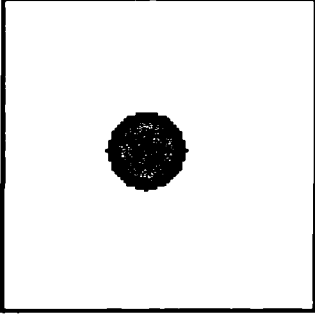
| Agriculture | | | | |
|-------------|---|---|--|---|
| Objective | IR | SOG | MOG | HIG |
| Method | 57445 | 104633 | 98820 | 104633 |
| Utility | | | | |
| Patch |  |  |  |  |

Table 8.18 Agriculture patches in evaluation test 5

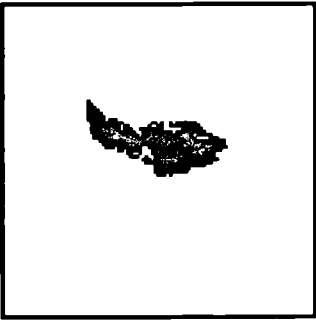
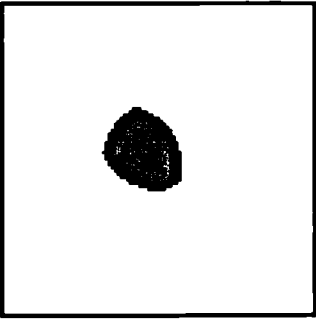
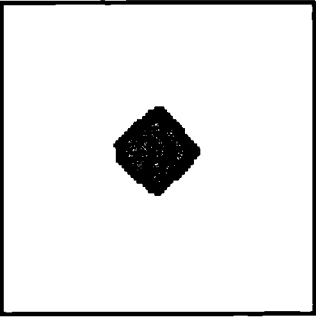
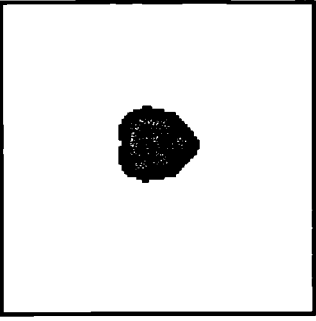
| Objective | Carpet Industry | | | |
|-----------|---|---|--|---|
| Method | IR | SOG | MOG | HIG |
| Utility | 84981 | 100457 | 97619 | 97680 |
| Patch |  |  |  |  |

Table 8.19 Carpet industry patches in evaluation test 5

The IR algorithm described in evaluation test 1 is spatial only in so far as patch size is a constraint, otherwise there are no spatial factors. The IR algorithm was not expected to find the optimal patch configuration but it does reveal something about the spatial pattern in the suitability layers. The threshold level to find a patch for agriculture (Table 8.20) was much lower at 3146 than that for the carpet industry (Table 8.21) at 4302. This indicates more spatial association between cells in the agriculture suitability map.

| Single patch for agriculture | |
|------------------------------|------------|
| Number of cells | Patch size |
| 3500 | 604 |
| 3200 | 501 |
| 3000 | 481 |
| 3151 | 501 |
| 3101 | 492 |
| 3141 | 497 |
| 3146 | 500 |

Table 8.20 IR iterations to find the agriculture patch (IR-AG) in evaluation test 5

| Single patch for carpet | | Single patch for carpet | |
|-------------------------|---------------|-------------------------|---------------|
| Number of cells | Largest patch | Number of cells | Largest patch |
| 3001 | 337 | 4351 | 510 |
| 4001 | 456 | 4311 | 505 |
| 4501 | 518 | 4306 | 503 |
| 4401 | 518 | 4303 | 501 |
| 4201 | 472 | 4302 | 500 |
| 4301 | 499 | | |

Table 8.21 IR iterations to find the carpet patch (IR-CARP) in evaluation test 5

Figures 8.28 and 8.29 show the locations of the two IR patches in relation to Kathmandu (defined by the ring road) and major roads. Both patches are elongated, narrow and located very close to Kathmandu. An examination of the suitability maps shows why elongated shapes are expected. Suitability for both agriculture and the carpet industry is closely related to distance from roads and to slope. Roads are associated with lower slope values and follow valleys, hence the most suitable cells also follow the valleys. Contiguous patches of high-ranking cells are likely to be linear rather than compact. Although the IR thresholds indicate that high-ranking cells in AGSUIT have greater contiguity than those in CARPSUIT, the patch IR-AG was much less compact than IR-CARP with a core area of 227 against 347. Thus, even though the cells in IR-AG have a higher average suitability, the core suitability, which is how utility is measured, is less than that of IR-CARP.

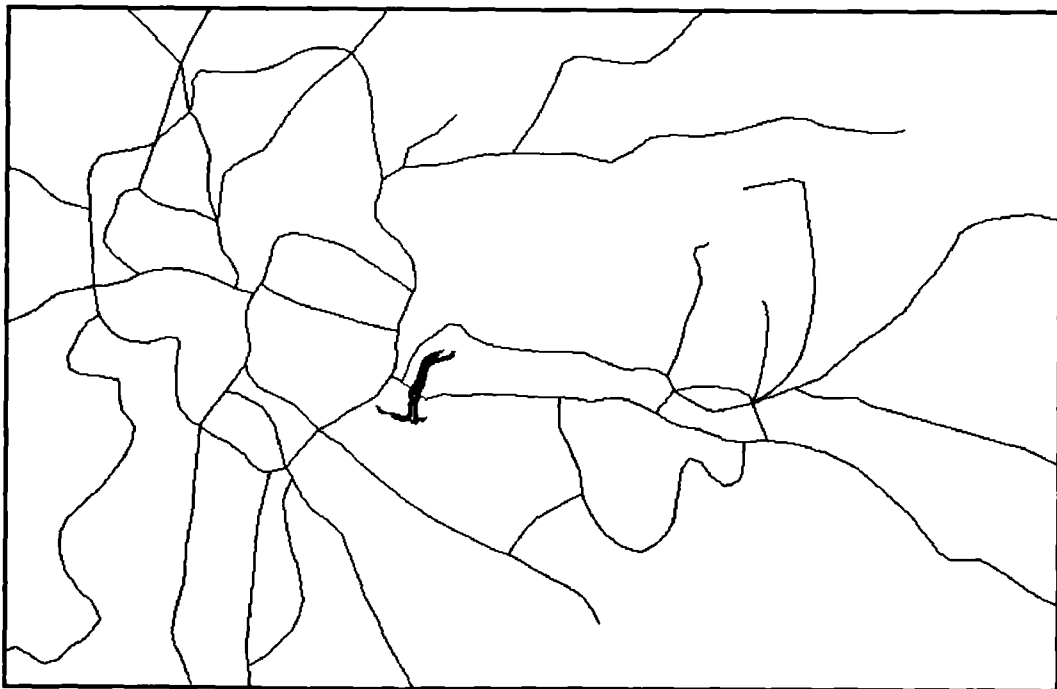


Figure 8.28 Location of the agriculture patch found by IR in evaluation test 5

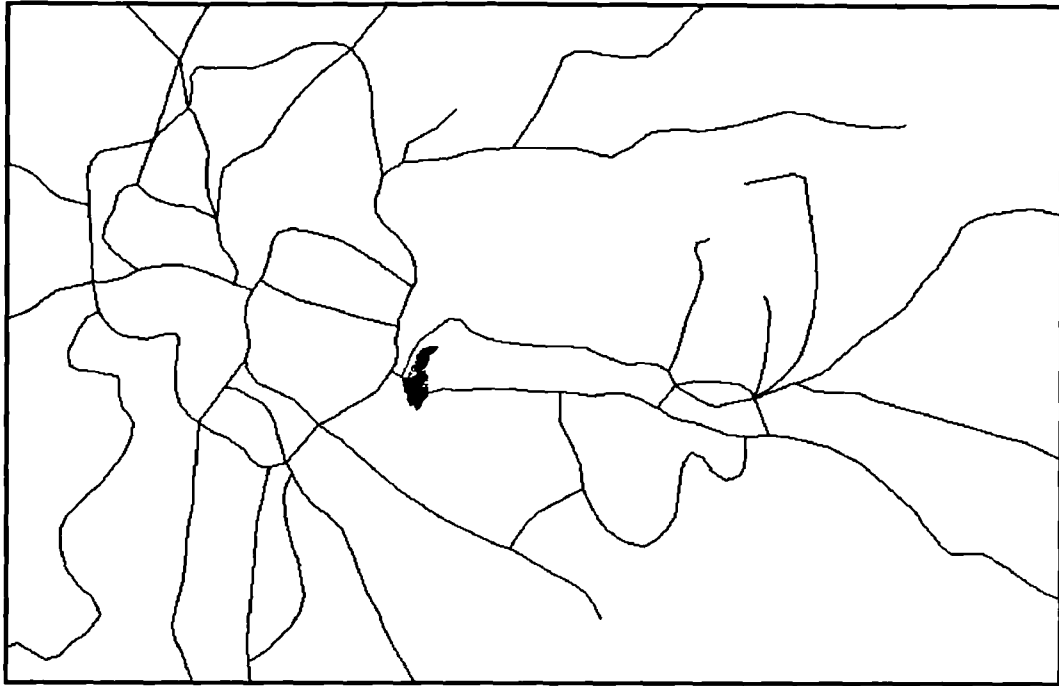


Figure 8.29 Location of the carpet industry patch found by IR in evaluation test 5

Single-objective GAPD was used to find upper bounds for the patch utility values by solving for each objective independently. The utility values of the patches SOG-AG and SOG-CARP were taken to be upper bounds of the multi-objective problem and were then used to normalise patch scores in the multi-objective GAPD. The results are shown in Table 8.17, along with values for the average, minimum and maximum cell values in the patch core. The corresponding values for the whole patch are also included.

In addition, the patches SOG-AG and SOG-CARP have a degree of conflict. The cross-tabulation of the two patches reveals that 383 of the core cells are shared between both patches. In other words, the solution to the carpet problem uses 383 of the best cells for agriculture and vice versa, which represents a very high degree of conflict because there is a 90% overlap of the two patches by area. The row and column positions of the two patch seeds are very close, as can be seen from Table 8.22, and this also indicates a large degree of conflict between the two objectives. This conflict means that any solution to the multi-objective problem involves a compromise between the two objectives and that

the utility of the two patches will be less than in the single objective case. Both patches are located close to the IR patches (compare Figures 8.30 and 8.31 with Figures 8.28 and 8.29 respectively).

| | Size | Shape bias | Shape control | | | | | | Row | Col |
|-------------|------|------------|---------------|-----|-----|----|-----|-----|-----|-----|
| | | | 9 | 6.0 | 85 | 10 | 4.5 | 104 | | |
| Agriculture | 500 | 1.0 | 9 | 6.0 | 85 | 10 | 4.5 | 104 | 286 | 291 |
| Carpet | 500 | 1.0 | 6 | 0.7 | 100 | 6 | 1.3 | 76 | 285 | 293 |

Table 8.22 PRG parameters (PDCs) for the single objective GAPD solutions in evaluation test 5



Figure 8.30 Location of the agriculture patch found by single objective GAPD in evaluation test 5



Figure 8.31 Location of the carpet industry patch found by single objective GAPD in evaluation test 5

The multi-objective problem was first solved using a hierarchical method. Arbitrarily, agriculture was chosen as the primary objective and carpet industry as the secondary objective. Patch SOG-AG was, therefore, taken as the solution to the agriculture objective and was overlaid on the CARPSUIT layer with zero utility to prevent the carpet patch from overlapping the agriculture patch. Single-patch GAPD was then used to find the best carpet patch, HIG-CARP, on the revised suitability map. Patch HIG-AG is described in Tables 8.17, 8.18 and 8.19 for convenience but is actually identical to patch SOG-AG. Patch HIG-CARP is located close to the IR (Figure 8.29) and SOG (Figure 8.31) agriculture patches. The locations of HIG-AG and HIG-CARP are shown in Figure 8.32.

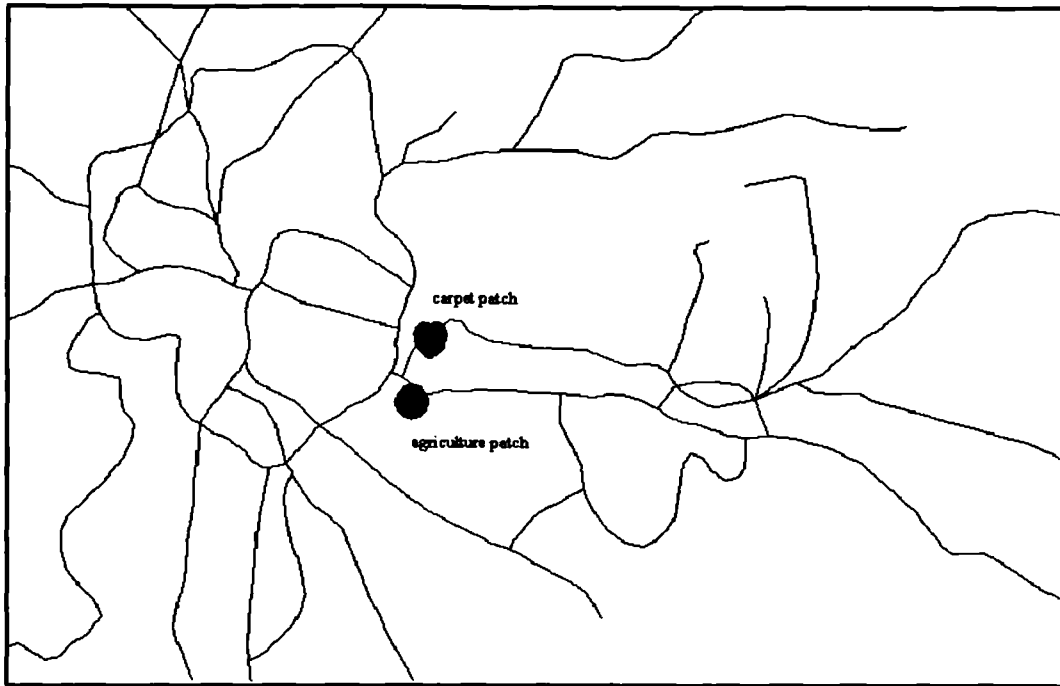


Figure 8.32 Location of the agriculture and carpet industry patches found by the hierarchical GAPD method in evaluation test 5

The single objective utilities found using SOG were used to normalise the scores in the multi-objective method, MOG. The upper bound values for each patch were set to 100500 for the carpet patch and 105000 for the agriculture patch. These values are round numbers slightly greater than the utilities found by GAPD for the single-objective problems (Table 8.17) and are used to ensure equal weight to each objective. The utilities of each patch are measured as absolute values but are then normalised to a scale of 0 to 1000 before they are combined into a single utility score. The values from the single objective problems define the 1000 mark. Equal weight was given to each objective.

The locations of the two patches found by the multi-objective method are shown in Figure 8.33. The agriculture patch MOG-AG is very close to the patches found in all other solutions but the carpet patch, MOG-CARP, is in a totally different location (see Figures 8.28 through 8.32) although it is still near to Kathmandu. The values for the two patches are listed in Table 8.17 along with all the other patches. The utility of each MOG patch against the appropriate objective is shown in Table 8.23. The utility against

the other objective is shown in brackets.

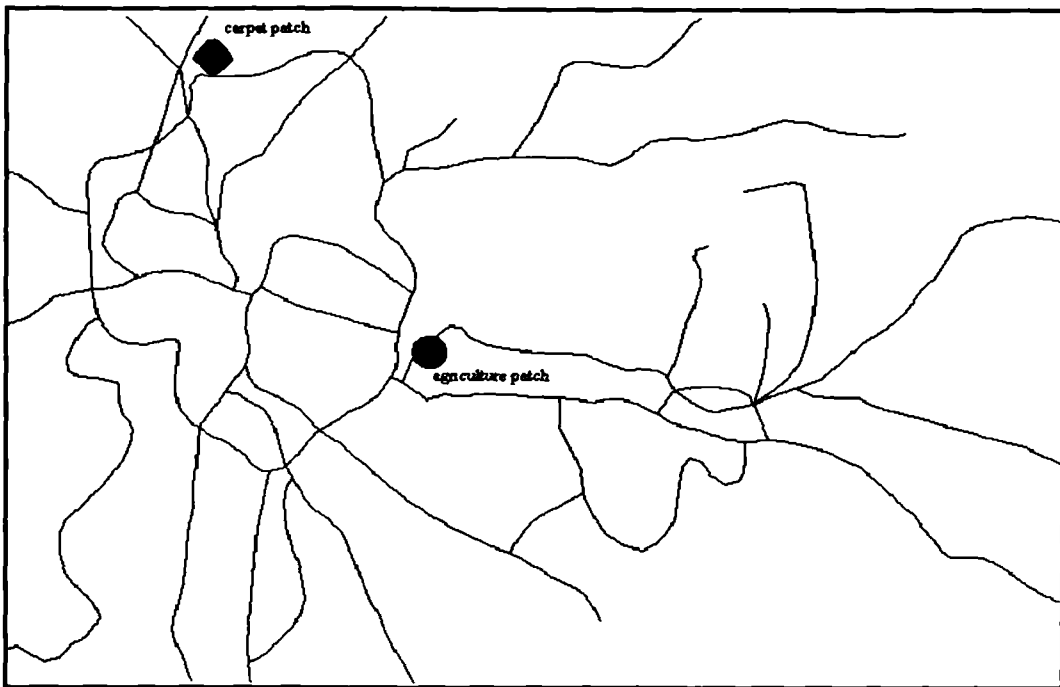


Figure 8.33 Location of the agriculture and carpet industry patches found by multi-objective GAPD in evaluation test 5

| | MOG-CARP | MOG-AG | Combined patches |
|-----------------------|----------|---------|------------------|
| Carpet industry | 97619 | [95089] | - |
| Agriculture | [101203] | 98820 | - |
| Normalised Utility | 971 | 941 | 956 |
| [Reversed allocation] | [963] | [946] | [955] |

Table 8.23 Utility of the multi-objective GAPD solutions in evaluation test 5

The normalised score for the combined objectives is 956. If the patches are reversed, so that MOG-AG is used for the carpet industry and MOG-CARP is used for agriculture, the utility is 955. The fact that the values are so similar indicates the close correspondence between the two suitability maps and also is an indication that the

solution is probably sensitive to the weights assigned to the different factors when creating the original suitability maps.

Interestingly, patch MOG-CARP is better than MOG-AG for both agriculture and the carpet industry which again shows the conflict between the objectives. Both patches are compact but have different shapes: MOG-AG is roughly circular and MOG-CARP is diamond shaped but MOG-CARP is slightly more compact (i.e. has a larger core area). MOG-AG has higher cell suitability values and higher absolute utility but MOG-CARP has higher normalised utility.

Table 8.24 shows the PDC parameters for the multi-objective GAPD solution. The actual sizes of the patches are 497 cells for MOG-AG and 503 cells for MOG-CARP. The reason they are not exactly the same is because of the program bug described above. Neither patch scores as well as the corresponding patch for a single objective (see sum of cells column in Table 8.17). The fact that MOG-CARP is located well away from the sites of SOG-CARP and SOG-AG means that MOG-AG could have occupied the same location as SOG-AG without compromising MOG-CARP. That it did not means that the solution is sub-optimal, since if it were identical to SOG-AG it would have a higher utility score but the score for MOG-CARP would not change, so the combined score would be higher. All the GAPD patches are compact, but whereas the agriculture patches are all approximately circular, the carpet patches have different shapes.

| | Size | Shape bias | Shape control | | | | | | Row | Col | Rel. Size |
|------------------|------|------------|---------------|-----|-----|---|-----|----|-----|-----|-----------|
| Header | 1000 | | | | | | | | | | |
| Patch 1 (Carpet) | - | 1.0 | 6 | 2.1 | 71 | 9 | 1.8 | 33 | 30 | 147 | 90 |
| Patch 2 (Agri.) | - | 1.0 | 9 | 6.3 | 152 | 9 | 6.0 | 16 | 246 | 303 | 6 |

Table 8.24 PRG parameters (PDCs) of the multi-objective GAPD solutions in evaluation test 5

Solving the multi-objective problem hierarchically, using single-patch GAPD, yields a

better combined answer than the multi-objective method using multi-patch GAPD. HIG-AG (which is identical to SOG-AG) is better than MOG-AG and HIG-CARP is better than MOG-CARP.

Although the multi-objective GAPD did not find an optimal solution, it performed reasonably well. The two MOG patches were almost as good as the SOG and HIG patches. In terms of efficiency, multi-patch GAPD is very good. The data layers have 490 rows by 761 columns so that there are about 360,000 possible seed positions for each patch. For two patches, that gives about 1.3×10^{11} combinations of seeds. When the number of patch configurations is considered (see Appendix IV), it is obvious that the search space is huge. In both single-patch and multi-patch modes, GAPD was run for up to 300 generations with a population of 300, so it tried at most 9×10^4 alternatives, which is a tiny proportion of the search space. The search space for the single-objective problem is much smaller than for the multi-objective problem and, therefore, it is not surprising that single-objective GAPD and hierarchical GAPD found better solutions than multi-objective GAPD.

8.6.5 Discussion

Of the five evaluation tests, this one is the closest to a real problem in the size of the dataset as well as in the data and criteria. The performance of GAPD with the large dataset is promising. GAPD searches efficiently and the quality of the solutions is good. The IR heuristic is inadequate for the problem because it gives insignificant weight to spatial factors and an exhaustive search of all possible alternatives is impractical.

GAPD's single-patch mode finds better solutions than its multi-patch mode. As in the other evaluation tests, this is probably because the search space is smaller and less complex. Nevertheless, the multi-patch GAPD performed very well.

Only minor modifications to the program code were required to enable GAPD to operate with multiple suitability layers. The question of how far the need for code modification

affects the generality of GAPD is discussed in the summary at the end of this chapter.

In absolute terms, both the multi-patch and single-patch GAPD solutions found better patches for agriculture than for the carpet industry, whereas the IR algorithm found a better patch for carpets. This is a reflection of the differences in the algorithms: IR is totally constrained by the spatial pattern in the data whereas GAPD can trade-off criteria for optimal utility.

One very interesting result is that the GAPD patches included zero cells. GAPD grew across boundaries to get the best configuration. This illustrates that GAPD optimises both configuration and composition. It also raises the issue of interpreting barriers. In the example barriers are not constraints. In other applications barriers may be constraints and barrier cells can not be incorporated into the patch. The PRG could be easily amended to reflect this and it would be interesting to test GAPD with this interpretation of barriers. IR effectively treats barriers as constraints and so the advantage of GAPD may not be so great when the more rigorous interpretation of barriers applies.

For agriculture, the maximum cell values in the GAPD patches are the same as those for IR but for the carpet patch they are worse. Of the three GAPD methods, the multi-objective GAPD generates the carpet patch with the worst utility. This is probably because the search space in the neighbourhood of the agriculture patch is very complex. Patches located near the agriculture patch will tend to have low fitness because their growth is constrained by the agriculture patch. There is a highly-suitable patch in that locality, as found by single-objective GAPD, but it is a very hard to find because neighbouring solutions are very poor.

The shape bias parameter was 1 in every solution. This means that GAPD has identified an ideal shape and optimises the location of that shape. This is in contrast to the feasibility tests in Chapter 7 when the shape bias value changed for different maps. Also the agriculture patches are all the same shape but the carpet patches are all different.

These issues are all related to the question of the appropriate use of a tool like GAPD. While to some extent the performance may be improved by technical means, such as better programming techniques and the use of more computer power, there are still unresolved questions. Given a set of inputs, GAPD is autonomous and it handles large complex search spaces very well. By itself, GAPD does not address uncertainty, which is perhaps best dealt with by using GAPD in an interactive, exploratory way.

8.7 Summary

8.7.1 Review

GAPD was tested in five very different situations. Only minor modifications were needed to the program code to adapt GAPD to each case.

In test 1, GAPD was used to locate a single compact patch and shown to be better than an alternative heuristic. In test 2, GAPD designed a network of patches subject to spatial constraints. Test 3 combined elements of tests 1 and 2 in a multi-objective problem. In test 4, different versions of GAPD were compared in a single-objective multi-patch problem. The last test was a multi-objective problem with conflicting objectives: it was a modified version of a case study from Kathmandu.

These tests not only serve to verify that GAPD works but also, as expected, raise some general points. These are set out below.

8.7.2 Generality of GAPD

One of the criteria for evaluating GAPD is its generality and applicability to different situations. The genetic driver uses a universal search heuristic that requires feedback about the quality of different alternatives but which is otherwise independent of the problem domain. Primitive raster GIS functions can be used to build complex measurement functions. Evaluation functions can be arbitrarily complex. The question

is how difficult is it to formulate evaluation functions relative to devising heuristics? In the tests, the evaluation functions were almost trivial and, intuitively, it seems that evaluation functions should not present the same difficulties as heuristics. The question is unresolved, however, and needs further investigation.

The question of adapting the PRG to particular problems is also pertinent. If it is necessary to alter the PRG component for each problem, then it is difficult to write a generic software system - but this does not compromise the generality of the method. The PRG method was originally invented as an heuristic for generating promising patch shapes and, as such, it is appropriate in certain problems. Minor changes, for example to the number of input layers or to the cell scoring mechanism as in test 4, are not major issues. However, there is the possibility that a more generic PRG algorithm may be found.

As a generic decision support tool, GAPD could be improved by developing generic routines that could be linked together and that would apply to many situations. At a more philosophical level, the applicability of GAPD to different situations should be investigated in another research project.

8.7.3 Performance

The performance of GAPD was never rigorously evaluated but some observations can be made. Test 5 is the closest to a real problem and uses by far the largest input maps. The run-times for tests 4 and 5 were very long compared to those for the first three tests. It seems that the cell allocation level has a very significant effect on run-time. The size of the raster also affects run-time because the search space is bigger when the rasters are larger. However, GAPD is efficient and the increase in run-time due to increasing the search space is less than linear; run-time depends more on the time taken to evaluate the fitness of each alternative. The GIS measurement functions and the MCDM evaluation functions used in the tests are not greatly affected by allocation level because the algorithms process all cells irrespective of whether they are in a patch or not. The

situation is different with the PRG component. Run-time of the PRG algorithm is very dependent on allocation level. The number of cells processed in each iteration increases more than linearly with size and the number of iterations increases linearly, therefore total run-time increases more than linearly with total patch size. The efficiency of the PRG algorithm appears to have a stronger effect on run-time than the size of the input maps. This is a topic for more rigorous analysis.

Consideration of the size of the search space and the number of evaluations performed by GAPD shows that it solves a complex problem in spatial geometry. The analysis in Appendix IV indicates that the number of alternative patches can be estimated as being of the order 2^N , where N is the number of cells. For a patch of size 10, the number of alternatives is 2^{10} which is about 10^3 . For a patch of size 20 the number grows to 2^{20} (or 10^6) and so on. If a fast computer has terraflop performance (i.e. can perform of the order 10^{12} floating point operations (flops) per second) then it can do about 10^{16} in an hour, 10^{17} in a day etc. Table 8.25 shows how complete enumeration quickly becomes infeasible as patch size increases. The right hand column tabulates the number of flops for different units of time. The left hand column shows the patch size, the number of alternative configurations and the solution time using complete enumeration measured in everyday time units taken from the right hand column. All the figures are approximate but actually underestimate the scale of the problem because patch generation and evaluation cannot be done with a single floating point operation. GAPD tackled problems with patches of 500 cells in less than 2 days and performed roughly 105 patch evaluations out of 10^{150} (2^{500}) possibilities.

| Problem size and time to solve | | | Computer resources per unit time | |
|--------------------------------|------------------------|-------------------------------|----------------------------------|-----------------|
| Patch size | Number of alternatives | Time for complete enumeration | Time unit | Number of flops |
| 10 | 10^3 | second | second | 10^{12} |
| 30 | 10^9 | second | hour | 10^{16} |
| 50 | 10^{15} | day | day | 10^{17} |
| 70 | 10^{21} | year | year | 10^{20} |
| 100 | 10^{30} | millennia | millennium | 10^{23} |

Table 8.25 Solution time for complete enumeration, in everyday time units, on a computer with terraflop performance.

No attempt was made to tune GAPD to the problems by altering the PRG or by changing the genetic operator probabilities. Varying the GAPD parameters such as population size and operator probabilities is a legitimate option since genetic algorithms should be tuned to problems. Varying the parameters is easy to do and does not present operational difficulties, as long as the run-time is short. Although it would be simple to change the GAPD parameters, it would have to be done in a controlled experiment and would take a considerable amount of effort. That kind of in-depth analysis was dropped in favour of running a broader series of tests.

8.7.4 Implementation

The fact that this is a genetic algorithm research project, not a software development project, is reflected in the way the code has been developed. Issues such as useability and program efficiency are subordinate to pragmatic considerations of time and resources. Although it is a desirable goal to produce a single system, with a simple interface, in practice this kind of development requires a lot of extra work. Consequently, the program code has been developed as a number of parallel versions of a basic idea. Each example problem has been developed independently in a separate directory. Of course, a lot of code is common to all the examples but it exists as a series of copies and not as a central resource. The task of integrating different versions of the programs is a matter for further work. The advantage of keeping parallel versions is

mainly that once a version works and is tested, it is preserved.

Ideally, there should be a single implementation of a full-function, multi-patch GAPD employing one genetic code. One step towards such an ideal would be to build libraries of GIS measurement functions, evaluation functions and versions of the PRG component. GAPD could be developed as a library of functions similar to the GRASS GIS. That would be an interesting project to develop GAPD further.

8.7.5 Appropriate use of GAPD

Perhaps the most fundamental question is the appropriate use of GAPD. The value of GAPD is in searching efficiently for good quality solutions. GAPD does not guarantee to find optimal solutions but this is not necessarily a bad thing. Given the complexity of spatial planning, and allowing for uncertainty, the whole idea of a global optimal solution is questionable. However, it is certain that some alternatives are better than others, objective criteria can be used to distinguish between them and that GAPD can be used to identify good alternatives.

8.7.6 Conclusion

Three criteria for evaluating GAPD were set out at the start of this chapter. Against these criteria GAPD performs well:

- it is effective - it finds good solutions;
- it is efficient - it evaluates only a small proportion of the possible solutions; and
- it is adaptable - domain-specific components are simple to implement.

9. REVIEW AND RECOMMENDATIONS FOR FURTHER RESEARCH

9.1 Summary

The primary objective of this research was to build an autonomous computer system that designs optimal patch configurations to meet multiple objectives subject to multiple criteria. The genetic algorithm for optimal patch design, GAPD, achieves this objective and is the first system to do so.

GAPD combines four components, a genetic search driver, a translation function, raster GIS measurement functions and MCDM evaluation functions. The translation function, performed by PRG, is the key element that brings together three established technologies.

The feasibility of using a computer system automatically to design optimal patch configurations was demonstrated in a number of tests. These tests showed that: measurements of patch configuration are dependent on scale and the metric used; and that patch shape is a meaningful concept in raster systems when patch size is sufficiently large. Evaluation functions that measure the effects of patch configuration, such as core area, are better than shape indices because they relate to patch value in a straightforward way. Feasibility tests (see 7.3) showed that PRG can represent all patch configurations and, therefore, GAPD can search the whole problem space. The efficiency of GAPD was proved by the fact that it can find a global optimum in a large search space with the same effort required to search a smaller space exhaustively (see 7.4).

GAPD was applied to a range of hypothetical problems using real data: single-patch problems, with single objectives and data layers to multi-patch problems with multiple conflicting objectives and multiple data inputs. Minor modifications were made to adapt the genetic driver and PRG components to suit different situations. GAPD was more successful on single-patch problems than multiple-patch problems. This is probably a reflection of the fact that the search spaces are much smaller in single-patch problems.

No attempt was made to tune the system to get better results so the performance of GAPD in the tests is probably less than could have been achieved. The fifth evaluation test (see 8.6) shows how the method could be applied to a real problem. The problem is based on a case study from Kathmandu (Eastman *et al.*, 1993). In the original landuse planning problem, configuration is irrelevant. The test problem uses the same data and criteria with an additional constraint that allocated land is placed in one of two patches.

Configuration is a recognised factor in landuse planning, but no autonomous integrated system has been developed previously. GAPD has a number of advantages over other *ad hoc* approaches.

- GAPD goes beyond suitability mapping. GAPD can operate on multiple attribute layers. Suitability mapping demands that compensatory MCDM methods be used. The visual interpretation of suitability maps by people can be misleading but GAPD is objective.
- GAPD avoids the problems associated with using landscape and shape indices to evaluate configurations. In GAPD, it is the effects of configuration on utility that are measured not some spurious index.
- GAPD handles dynamic criteria at the appropriate level. It avoids the need to invent problem-specific heuristics.
- Any MCDM method can be incorporated into GAPD. The genetic driver only requires feedback about the relative value of different alternatives. This means non-compensatory methods can be used instead of the compensatory methods normally used in raster GIS.
- Arbitrarily complex utility functions can be used. Measurement and evaluation functions are independent of the PRG and genetic driver components.
- GAPD can incorporate some degree of uncertainty, such as soft constraints.
- Costs can be incorporated without affecting the driver or the PRG.
- GAPD is autonomous and objective. Assumptions and judgements are reflected in the inputs to GAPD and are, therefore, explicit.
- GAPD exploits the best features of different techniques. Each component is used

appropriately.

9.2 Recommendations for further research

The GAPD methodology addresses the main issues in optimal patch design but there are many areas for further research. These include technical developments of the computer system, application to real problems and investigations of general issues. Two major GIS research areas particularly relevant to patch design are uncertainty and time. The issue of uncertainty was not covered in depth. It is acknowledged that uncertainty is an important issue for further work. GAPD should be considered a decision support tool to be used in an appropriate context, not as an automatic decision-making system. The dynamic nature of systems is also acknowledged to be an issue. To some extent, the guidelines for nature reserve design in Chapter 3 take account of dynamics, for example by considering meta-populations and disturbance regimes.

Some suggestions for further research follow.

Technical developments

- Investigate alternative translation components by either replacing PRG or improving the PRG algorithm. Ideally, the PRG algorithm should be totally independent of the problem domain. Currently, it is the combination of underlying data with shape scoring rules that allows PRG to grow arbitrarily complex shapes. To be independent, the PRG algorithm would have to ignore the input data and still grow arbitrarily complex shapes.
- Optimise the program code for the genetic driver and PRG components.
- Couple GAPD and GIS more closely, either by implementing GAPD within a GIS or by linking GIS functions to GAPD.
- Improve the genetic algorithm search driver. Hybrid algorithms often are more

successful than the canonical algorithm in optimisation problems. Different selection strategies would allow more flexibility in the MCDM routines. The question of integrating GAPD into a Spatial Genetic Algorithms framework such as that proposed by Krzanowski (1997) should be looked at.

- Implement GAPD in a parallel processor. Genetic algorithms are inherently parallel. There are different strategies for decomposing spatial problems to run as parallel algorithms (Armstrong & Densham, 1992).
- Develop GAPD as a product, possibly as an Internet tool.

Application to real problems

- Explore issues of defining criteria and evaluation functions and also the question of how the output from GAPD is interpreted.
- Investigate the applicability of GAPD to different problems. In what situations is GAPD a good method? What are the problems? When does GAPD not work and why?
- Perform sensitivity analysis. To what extent is GAPD sensitive to uncertainty in data, criteria and objectives. How does the sensitivity within GAPD relate to sensitivity in the real world?

General issues

- Experiment with different MCDM methods particularly with non-compensatory methods.
- Investigate the use of landscape indices through experiments to relate landscape indices to value. It is probable that a number of standard indices would have to be combined.

- Use simulation models as evaluation functions. Good simulation models capture the dynamics of systems. They are computationally expensive but feasible with parallel systems and powerful computers. Simulation models are potentially very flexible evaluation functions.
- Integrate GAPD within a DSS - this is probably the most appropriate way to use GAPD.
- Identify a core of measurement and evaluation functions that can be used as primitives to build arbitrarily complex functions.
- Investigate useability. An Internet tool, as suggested above, would be one way to approach this.
- Test GAPD performance. What sort of computer resources are needed? What is the relationship between increasing resources and increasing the quality of solutions? Are there any general principles which can be used as guidelines for operating GAPD?
- Investigate ways of handling error and uncertainty. Uncertainty can be incorporated using techniques such as Monte-Carlo modelling, fuzzy sets and probability theory. Alternatively, GAPD can be integrated into a DSS.

9.3 Integration with GIS

GAPD was implemented as a standalone suite of functions coded in C. One of the objectives was to investigate ways of integrating GAPD with GIS. There are broadly two alternative routes: tight coupling and loose coupling. With tight coupling all components are closely linked under a single interface, sharing common data structures and running on a single platform. Loosely coupled systems run as separate entities, possibly with different interfaces, sharing data through the exchange of files and maybe even running on different platforms. Modern operating systems and software

engineering paradigms like object orientation are blurring the distinction between tight and loose coupling because systems are written as software components which can call and be called by other products. Many systems include tools for building interfaces and these interfaces have a common look and feel. This has the effect of creating the impression that a single system is running when in fact there are several.

Although the current implementation of GAPD includes raster GIS functions within itself, it is actually loosely coupled with GIS. A GIS is required to process the input, which must come from somewhere, and output data. GAPD generates patch maps but GIS software is needed to display them. GAPD reads and writes IDRISI files so GAPD is loosely coupled with IDRISI. It could be more closely coupled by customising an IDRISI interface that makes the GAPD routines available. A windows style interface would allow users to select datasets, choose criteria and weights and specify objective functions to GAPD. The interface would effectively hide GAPD from the user without actually tightly coupling the systems. There are a number of other options using different GIS. One would be to couple GAPD with GRASS - a library of GIS functions. GAPD could be written to call GRASS functions the way it calls measurement functions now and to read and write GRASS datasets. All the GIS functions of GRASS would then be available to GAPD, or vice versa, and everything could be run from a GRASS interface. GAPD routines could be incorporated into commercial GIS packages like ARC/INFO as new functions and become part of the GIS.

9.4 Other aspects not previously covered

A number of aspects of the project have not been discussed in the main body of the thesis because they are not central to the project. These include investigations of potential applications and other techniques that could be integrated with GAPD.

In developing GAPD, and testing it on hypothetical problems, I was always conscious that GAPD should have the potential to be a practical decision-support tool. I had discussions with a number of representatives of planning and conservation organisations

including the Planning Department at Hampshire County Council (HCC), the Hampshire Wildlife Trust (HWT), and the Forestry Commission. These discussions indicate that there are potential uses of a system like GAPD but also many serious difficulties with the sort of applications I had originally envisaged. There is still a large gulf between the complexities of the real problems and the methods developed in this thesis. There are two major practical difficulties. In practice it is very difficult to get data, even for the organisations concerned, because of the costs involved. It is also difficult to define objective functions because of the complexity of real situations and the number of interests to be satisfied. These two considerations alone put such an application outside the scope of this research.

However, the discussions do indicate that the method would be useful for identifying options rather than for drawing up rigid plans. For example, the HWT would be interested in identifying sites that would be worth purchasing if the opportunity arose. The planning department at HCC would want to use such a system to identify potential sites that could then be used as an input to planning discussions. A method using objective functions is possibly better than one based on interpretive methods in the sense that it is more convincing when dealing with non-specialists.

In the discussions it became clear that the appropriate use of technology is an issue. The planning system in the UK, and the pattern of land ownership, make it extremely unlikely that any automated algorithmic planning mechanism would be *politically feasible* on its own. Rather, technology should provide a toolkit for use in an interactive decision-making process, ideally facilitated by a DSS.

In other cases, there are situations where whole landscapes are planned by a single organisation. These include large managed forests, watersheds and military land. Forests are often managed for recreation and conservation as well as timber production. The forestry commission in the UK is now putting more emphasis on recreation and conservation and, to this end, is attempting to re-configure forests to change the old pattern of large geometric blocks to a more natural pattern of smaller blocks with

irregular edges (Hibberd, 1991). The more natural pattern is expected to provide a better environment for wildlife and to be more appealing to people. Pattern also has an economic impact by affecting the amount of wind damage and the risk of fires. In areas where soil erosion and flooding are problems, watersheds should be managed strategically. Controlling the pattern of landuse may be a method of mitigating such environmental problems. The US military manage large tracts of land mainly as training ranges but they also have other responsibilities and objectives. They require integrated landuse planning systems (Sydelko et al., 1997).

Sometimes objectives are clear and easy to quantify, as in minimising flood damage, but there may be less clear objectives. The preservation of landscape quality is desirable where a landscape is recognised as having a significant cultural or conservation value. The holistic approach used in landscape ecology may be appropriate to this situation. If a suitable index of landscape character can be derived, then alterations to the landscape by changes in landuse and the siting of new developments can be evaluated by comparing the indices of the new landscape with the old. Alternatives which minimise the change would be chosen in preference to others which have a greater impact.

Two technologies which may be of use in landuse planning are artificial neural networks and computer simulation models. Neural networks are good at pattern recognition and classification. They operate as a black box, so that the user does not know how the neural network arrives at decisions. If it were possible to train a neural network to discriminate between good landscapes and poor landscapes then the neural network could fulfil the evaluation role in the optimisation system.

Simulation models could make very good evaluation functions because they model the dynamics and stochastic aspects of natural systems. Ecological populations fluctuate and simulation models that predict the changes in populations through time may be better than static evaluation functions. For example, the minimum population size expected in the next N years could be used as the objective function. Likewise, hydrological models could predict the amount of flood damage or forestry models could predict the likely

amount of damage through fire.

Simulations are relatively time consuming and are, therefore, not immediately suitable for coupling with genetic algorithms which require many evaluations and, therefore, many simulations. However, as computer processing power increases and with advances in parallel processing techniques they may become feasible in the near future.

9.5 Contribution

The main contribution of this research is the development of GAPD, the first generic, autonomous computer system for optimal patch design. This research makes further contributions to GIS, planning and landscape ecology. In addressing optimal patch design as a generic spatial planning problem this research puts optimal patch design on the GIS research agenda. Optimal patch design is a generic raster GIS technique with much potential to help in site selection and landuse planning problems. The applications in conservation which this thesis has emphasised have analogues in other application domains to which the same concepts and techniques can be applied. This thesis contributes to landscape ecology by developing a tool by which the theories and principles of landscape ecology can be used in a prescriptive manner.

In building GAPD the research has addressed four key issues:

- measuring configuration and the effects of configuration;
- evaluating configuration and the effects of configuration;
- representing configuration; and
- generating alternative configurations.

Further developments of the system and further research into these issues is needed.

This project has established a framework for formulating optimal patch design problems and developed a conceptual model for an autonomous decision support tool, GAPD. The research has reviewed effects of configuration in ecological systems and established some basic properties which can be measured, evaluated and related to utility. The functional

components of GAPD and appropriate technologies to fulfil those functions have been identified. The work culminates in the implementation of a working system that responds to changes in criteria and objectives, is efficient and is effective. The applicability of GAPD to different situations has been demonstrated with hypothetical problems and real applications that will benefit from GAPD have been identified.

The feasibility (Chapter 7) and evaluation (Chapter 8) tests have demonstrated the potential of GAPD. The technique successfully addresses the key issue - the incorporation of composition and configuration together - because it evaluates patch configurations explicitly and the evaluations direct the search algorithm to good solutions. GAPD is effective in finding good solutions in complex search spaces. The solutions are not optimal but they do not need to be because GAPD can be linked with a DSS. Within a DSS, GAPD can be used to solve uncertain, loosely structured problems. There is scope for continued development of GAPD and this project has laid a foundation for further research into optimal patch design.

Appendix I. Technical notes

All the code for GAPD was written in the C programming language using Borland Turbo C 3.5 for DOS (Schildt, 1991). The first program tests were run on a 486 PC under DOS. With minor modifications, the programs also ran on a DEC ALPHA under UNIX. The IDRISI GIS was used for managing raster datasets, for producing graphics and for some analysis during development. IDRISI was also used to verify some of the solutions to the sample problems.

Initially the idea was to use the IDRISI raster GIS for the evaluation functions. IDRISI is made up of a number of stand-alone executable programs which should make it easy to integrate with other software. IDRISI does not have a good interface with C programs but C programs can call IDRISI programs by making calls to the DOS operating system. However, the linkage between DOS programs and IDRISI was found to be too cumbersome and slow and overall execution time on a 486 PC was not acceptable. The time to evaluate the utility of a solution is critical in a genetic algorithm. The major problem was that IDRISI returns values in ASCII text files which have to be read and this extra file handling involves a considerable performance overhead. For the simple processing needed in most of the evaluation functions it was found to be much better to code them in C and link them with the genetic search driver and the PRG code into a single executable program. Coding evaluation functions did not take long and they were checked by comparing with the results from the equivalent IDRISI functions.

The only non-trivial program is the `cost_spread` function. This program uses a version of a spread function algorithm described by Jianping and Lathrop (1995). The cost-spread function was coded to use integer distances instead of floating point values. In this way execution time was reduced but accuracy was lost. It was hoped that the faster algorithm would make inter-site distance calculations feasible but execution time was still too long.

More recent, or future, versions of IDRISI may have better linkages and faster

algorithms. Also, other GISs with raster capabilities could be used. GRASS, which is a library of C programs, is designed to be integrated with user written C programs. ARC/INFO has a GRID subsystem and a GRID macro language but the system is not flexible. FRAGSTATS is a library of routines for measuring patch and landscape attributes (McGarigal & Marks, 1994).

GAPD reads and writes raster files which are in IDRISI image format and have associated IDRISI document files. The GAPD reads and writes IDRISI document files. This facilitates the use of IDRISI for house keeping and the display and analysis of results.

Programming and linkage

The logical components are coded as separate entities but linked as one module. The genetic search driver is also the driver for GAPD and it calls the other functions. The translation, evaluation and measurement functions are all called as C functions. Each component makes use of a large number of other sub-routines held in library files. Routines accept a number of arguments, including pointers to raster images in memory.

There are two possible mechanisms for linking the generic parts of GAPD to the application specific parts. Components can be linked at execution time (dynamic linkage) or compilation time (static linkage). Dynamic linkage is more efficient and is the preferred way for software development. In GAPD, functions are linked at compilation time.

Currently, sub-routine calls are hard-coded and GAPD is re-compiled for every problem with only the relevant functions. The choice of which evaluation function to use at execution time could be controlled dynamically via the control file. This could be achieved using static or dynamic linkage. With static linkage, however, the executable program would be very large enormous because all sub-routines would be linked together. This is inefficient because, in each application, most sub-routines are

superfluous. Dynamic linkage is more flexible and is the preferred way in commercial software development.

Appendix II. Psuedo code.

II.1 Basic region-growing algorithm

SRG, PSG and PRG use the same basic algorithm but with different scoring procedures. In SRG the cell score is taken from the input layer: PSG and PRG calculate a score using the SDC. Examples of scoring procedures are given below.

The algorithm marks cells on the output layer as belonging to one of three categories: patch, boundary, background. Neighbours are cells which share a common edge: diagonally linked cells are not neighbours. The input and output layers are denoted by, in, and, out, respectively.

```
//initialise the out raster to all background
begin
for r=1 to no.rows
  for c=1 to no.cols
    out[r][c] = background
  endfor
endfor
//mark the seed and its four neighbours
out[seed_row][seed_col] = patch
out[seed_row-1][seed_col] = boundary
out[seed_row+1][seed_col] = boundary
out[seed_row][seed_col-1] = boundary
out[seed_row][seed_col+1] = boundary
//establish the bounding rectangle
min_row = seed_row-1
max_row = seed_row+1
min_col = seed_col-1
max_col = seed_col+1
count = 1
//grow the patch to the specified size
while count < patch_size
  best_score = WORST_POSSIBLE
  best_row = 0
  best_col = 0
//identify the best of the boundary cells
for r=min_row to max_row
  for c=min_col to max_col
    if out[r][c] = boundary
      cell_score = get_score()
      if cell_score is better_than best_score
        best_row = r
        best_col = c
```

```

        best score = cell score
    endif
endif
endfor
endfor
//add the best cell to the patch and mark the new boundary cells
out[best_row][best_col] = patch
count = count + 1
if out[best_row-1][best_col] = background
    out[best_row-1][best_col] = neighbour
if out[best_row+1][best_col] = background
    out[best_row+1][best_col] = neighbour
if out[best_row][best_col-1] = background
    out[best_row][best_col-1] = neighbour
if out[best_row][best_col+1] = background
    out[best_row][best_col+1] = neighbour
//redefine bounding rectangle
if best_row-1 < min_row
    min_row = best_row - 1
if best_row+1 > max_row
    max_row = best_row + 1
if best_col-1 < min_col
    min_col = best_col - 1
if best_col+1 > max_col
    max_col = best_col + 1
endwhile
end

```

Examples of scoring procedures

Score calculation for SRG:

cell_score = in[r][c]

Score calculation for PSG:

dx = r-seed_row

dy = c-seed_col

dist = sqrt(dx² + dy²)

θ = arctan(dx/dy)

mod = R + ((1-R) * cos((θ *A)+O))

cell_score = mod * dist

Score calculation for PRG:

$\text{static_score} = \text{in}[\text{r}][\text{c}]$

$\text{diag} = \sqrt{(\text{min_row} - \text{max_row})^2 + (\text{min_col} - \text{max_col})^2}$

$\text{dx} = \text{r} - \text{seed_row}$

$\text{dy} = \text{c} - \text{seed_col}$

$\text{dist} = \sqrt{\text{dx}^2 + \text{dy}^2}$

$\theta = \arctan(\text{dx}/\text{dy})$

$\text{mod} = R + ((1-R) * \cos((\theta * A) + O))$

$\text{shape_score} = (\text{diag} - \text{mod_dist}) / \text{diag}$

$\text{shape_score} = \text{normal_factor} * \text{shape_score}$

$\text{cell_score} = (\text{static_score} + \text{shape_score}) / 2$

Notes:

The value of the constant WORST_POSSIBLE and the meaning of the relation is_better_than vary. For PSG the minimum scoring cell is the winner while for SRG and PRG the highest is the winner.

In PRG, normal_factor is used to normalise shape_score to make it compatible with static score. Static score should be on a scale of 0 to MAX and the normal_factor is then equal to MAX.

II.2 GAPD pseudo code

The following symbols are used:

P = population size

G = maximum number of generations

Max_T = maximum number of tries permitted without improvement

H = number of parameters (chromosomes) in the header

N = number of patches (ie. parameter groups in the tail)

T = number of patch parameters per patch

D = number of domains

MinD[i] = minimum value of domain i

MaxD[i] = maximum value of domain i

Prob_cross = probability of crossover (likewise Prob_mutation etc.)

old_pop[] = current population

old_pop[i].parm[j] = j'th parameter (chromosome) of individual i

old_ind = individual in old_pop[]

new_ind = individual in new_pop[]

new_pop = next population

data_layers = all relevant input GIS layers

II.2.1 Main program

```
//main module
//Initialise
begin
initialise_pop(P,old_pop[])
count = 1
max fitness = 0
tries = 0
// run G generations or until Max_T exceeded
while count < G and tries < Max T
  best_so far fitness = 0
  best_so far = 0
  sum fitness = 0
  for i=1 to P      evaluate all individuals
    grow_patch(old_pop[i],data_layers,patch_map)
    old_pop[i].fitness = evaluate(patch_map)
    sum fitness += old_pop[i].fitness
```

```

    if old pop[i].fitness > best so far fitness
        best-so far = i
        best so far fitness = old pop[i].fitness
    end if
    if best so far_fitness > max fitness
        max_fitness = best so_far fitness
        tries = 0
    else
        tries++
    end if
end for
generate pop(P,old pop,sum_fitness,new pop)
new pop[0] = old pop[best_so far]      keep best of old pop
count ++                               increment generations count
end while
//record results and finish
write patch map(best so far,data layers)
write text_file(best so far)
end

```

II.2.2 initialise population

```

initialise_pop(P,pop[])  generate P random individuals
begin
for i=1 to P              for every individual
    for j=1 to H          for every parameter in the header
        x = random(MinD[j],MaxD[j])
        pop[i].parm[j] = x
    end for
    for j=1 to N          for every patch in the tail
        for k=1 to T      for every parameter in the patch
            x = random(MinD[k+H],MaxD[k+H])
            pop[i].parm[H+((j-1)T)+k] = x
        end for
    end for
end for
end for
end

```

II.2.2 generate new population

```

generate_pop(P,old pop[],sum_fitness,new pop[])
begin
for i=1 to P              for every individual
    a = roulette select(old pop[],P,sum_fitness)
    prob = random(0,1)
    if prob < Prob cross
        b = roulette select(old pop[],P,sum_fitness)
        i++
        crossover(old pop[a],old pop[b],new pop[i],new pop[i+1])
        iterate for loop
    end if
end for

```

```

if prob < Prob mutation
  mutate(old pop[a],new pop[i])
  iterate_for_loop
end_if
if prob < Prob_sum
  b = roulette_select(old pop[],P,sum_fitness)
  i++
  sum(old pop[a],old_pop[b],new_pop[i],new_pop[i+1])
  iterate_for_loop
end_if
if prob < Prob_creep
  creep(old_pop[a],new_pop[i])
  iterate_for_loop
end_if
if prob < Prob_average
  b = roulette_select(old_pop[],P,sum_fitness)
  i++
  average(old_pop[a],old_pop[b],new_pop[i],new_pop[i+1])
  iterate_for_loop
end_if
new_pop[i]=old_pop[a]  copy individual unchanged
end_for
end

```

```

mutate(old ind,new ind)
begin
num_codes = H + (T*N)
D = H+T
new_ind = old ind
k = random(1,num_codes)  select a chromosome
if k > D
  j = H + (k-H)%T
else
  j = k
end_if
x = random(MinD[j],MaxD[j])
new_ind.parm[k] = x
end

```

```

creep(old_ind,new ind)
begin
num_codes = H + (T*N)
D = H+T
new_ind = old_ind
k = random(1,num_codes)  select a chromosome
if k > D
  j = H + (k-H)%T
else
  j = k
end_if
range = MaxD[j] - MinD[j]
delta = range/10

```

```

flip = random(0,1)
if flip >= 0.5
    x = old ind.parm[k] + delta
else
    x = old ind.parm[k] - delta
end if
if x > MaxD[j]
    x = MinD[j] + (x-MaxD[j])%range
else
    if x < MinD[j]
        x = MaxD[j] - (MinD[j]-x)%range
    end if
end if
new ind.parm[k] = x
end

crossover(old ind1,old ind2,new ind1,new ind2)
begin
num codes = H + (T*N)
D = H+T
new ind1 = old_ind1
new ind2 = old_ind2
k = random(1,num_codes)  select a chromosome
new ind1.parm[k] = old_ind2.parm[k]
new ind2.parm[k] = old_ind1.parm[k]
end

sum(old ind1,old ind2,new_ind1,new_ind2)
begin
num codes = H + (T*N)
D = H+T
new ind1 = old ind1
new ind2 = old ind2
k = random(1,num_codes)  select a chromosome
if k > D                  identify the domain
    j = H + (k-H)%T
else
    j = k
end if
range = MaxD[j] - MinD[j]
x = old ind1.parm[k] + old_ind2.parm[k]
y = old ind2.parm[k] - old_ind1.parm[k]
if x > MaxD[j]
    x = MinD[j] + (x-MaxD[j])%range
else
    if x < MinD[j]
        x = MaxD[j] - (MinD[j]-x)%range
    end if
end if
if y > MaxD[j]
    y = MinD[j] + (y-MaxD[j])%range
else

```

```

    if y < MinD[j]
        y = MaxD[j] - (MinD[j]-x)%range
    end if
end if
new ind1.parm[k] = x
new ind2.parm[k] = y
end

average(old ind1,old ind2,new ind1,new ind2)
begin
num codes = H + (T*N)
D = H+T
new ind1 = old ind1
new ind2 = old ind2
k = random(1,num codes)    select a chromosome
if k > D                    identify the domain
    j = H + (k-H)%T
else
    j = k
end if
x = old ind1.parm[k] + (2*old ind2.parm[k]) / 3
y = old ind2.parm[k] + (2*old ind1.parm[k]) / 3
new ind1.parm[k] = x
new ind2.parm[k] = y
end

roulette_select(pop[],P,sum fitness)
begin
r = random(1,sum fitness)
sum = 0
i = 0
while (r<sum) and i<P
    i++
    sum = sum + pop[i].fitness
endwhile
return (i)
end

```

Appendix III. Operational modes for GAPD.

GAPD can be run in various modes depending on how static criteria are handled. An efficient way is to pre-process static criteria to generate suitability maps and cost surfaces. Static criteria are processed once for each cell. Dynamic criteria are evaluated in GAPD for each patch configuration. Dynamic criteria are input to GAPD as measurement and evaluation rules.

Four operational modes are illustrated in Figures III.1 through III.4. The examples correspond to the modes used in the evaluation tests.

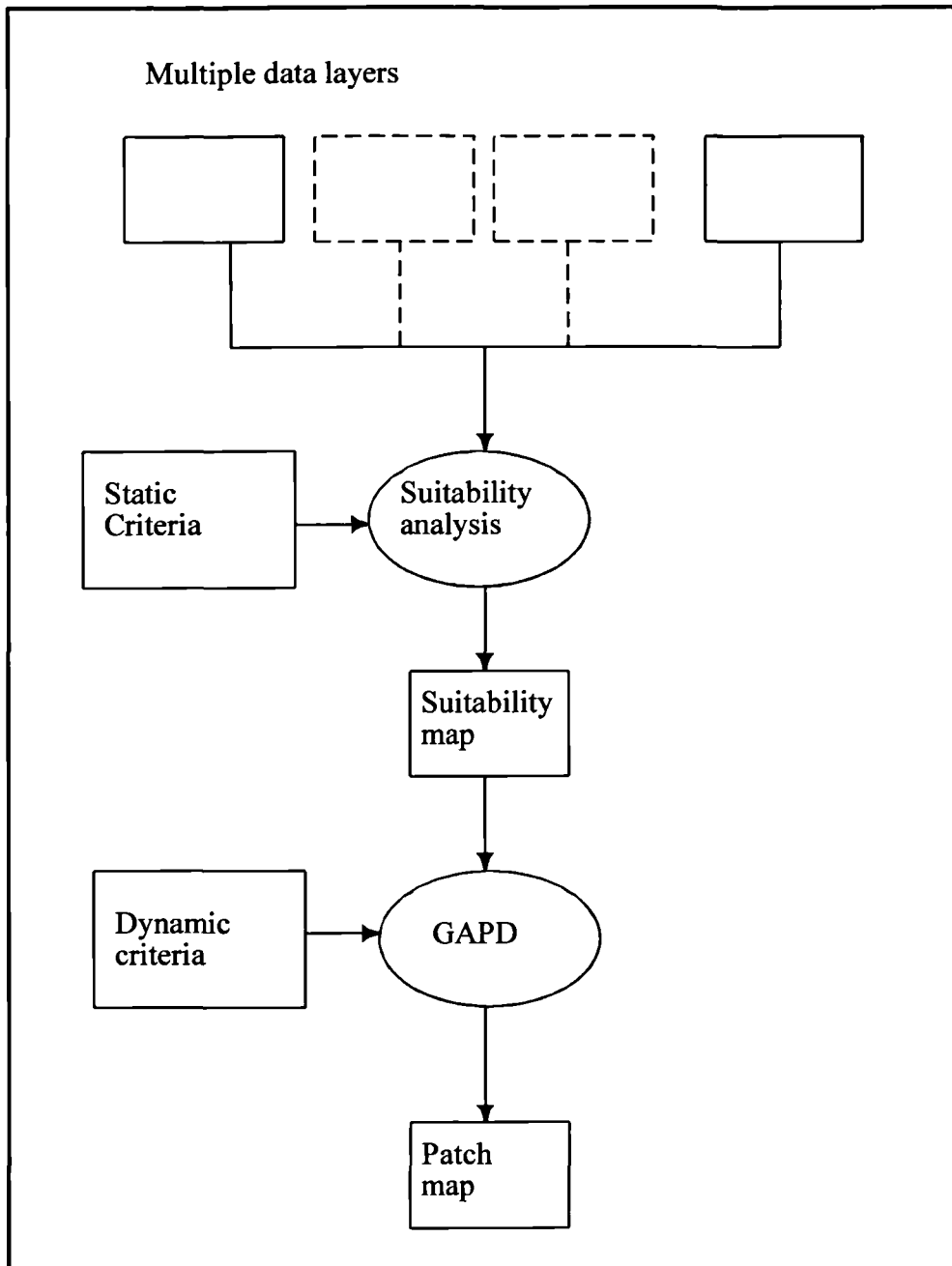


Figure III.1 Simple operational mode for GAPD: all static criteria are pre-processed to generate a single suitability layer. This mode was used in evaluation tests 1 and 2.

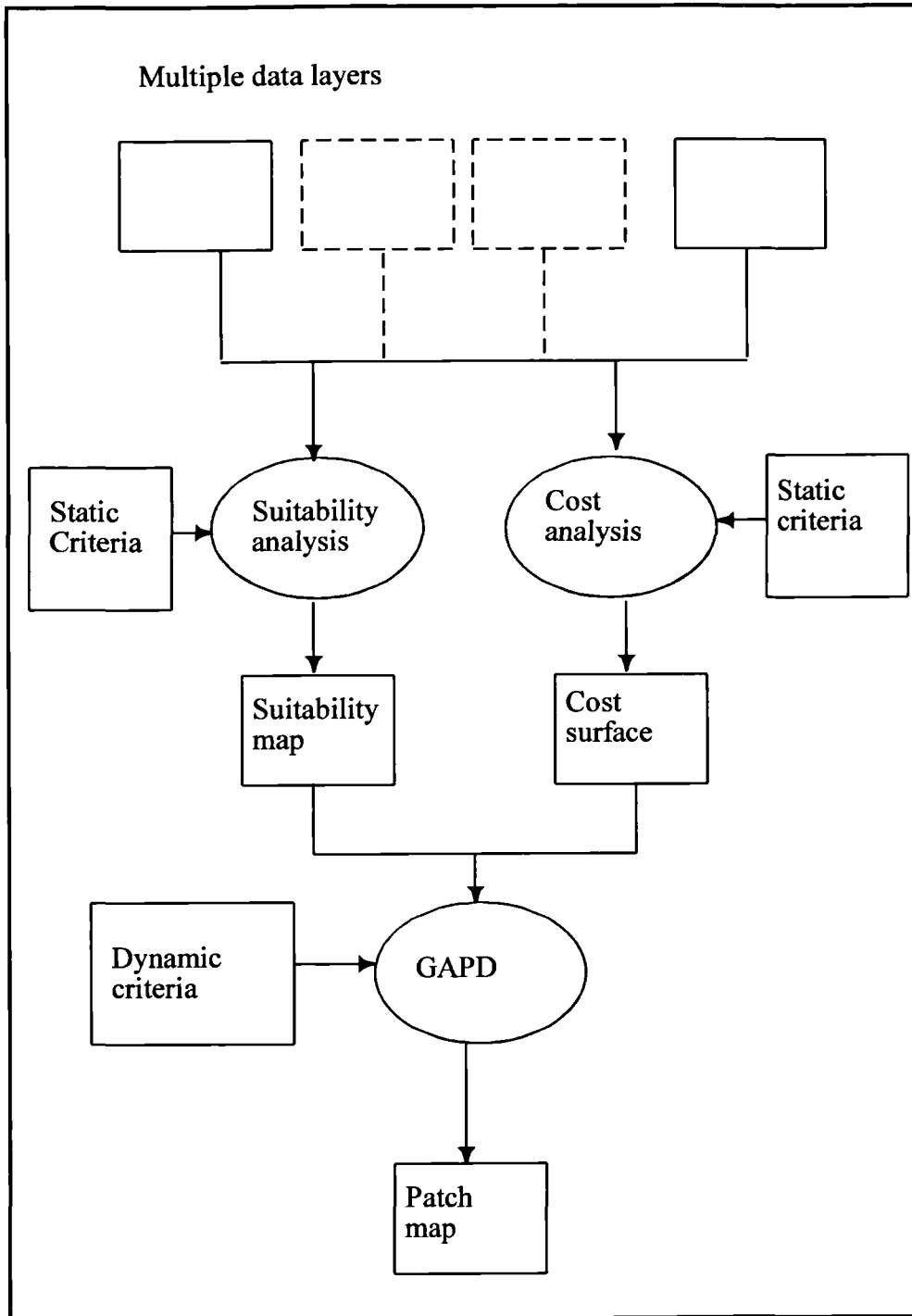


Figure III.2 Operational mode for GAPD: static criteria are pre-processed into a suitability layer and a cost surface. This mode was used in evaluation test 3.

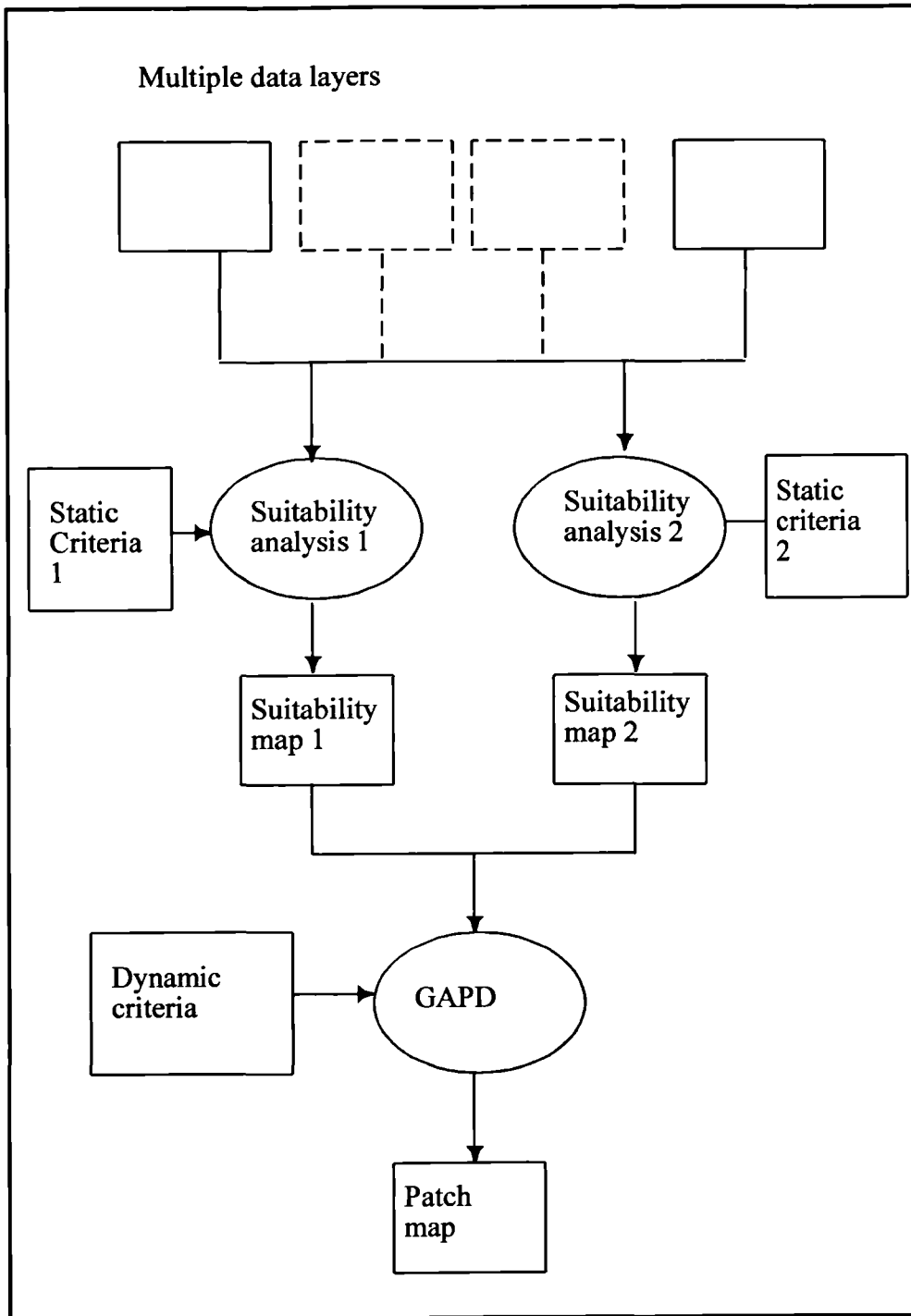


Figure III.3 Operational mode for multiple conflicting objectives: static criteria are pre-processed into two suitability maps, one for each objective. This mode was used in evaluation test 5.

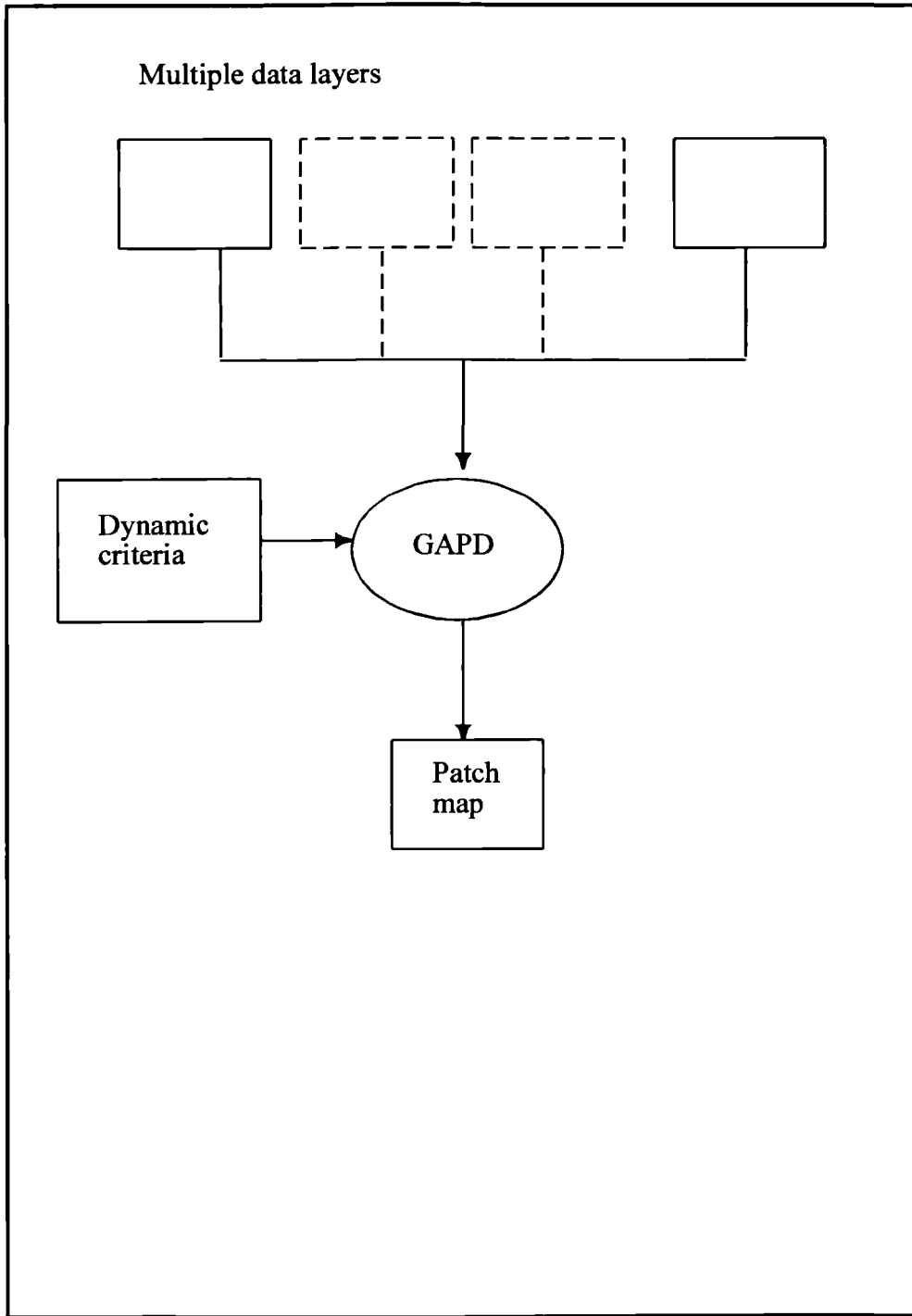


Figure III.4 Operational mode for GAPD: there are no static criteria, data layers are fed directly into GAPD. This mode was used in evaluation test 4.

Appendix IV. Estimating the size of the decision space.

Single-patch problem.

For a single-patch problem the size of the decision space is equal to the number of unique patches. This is equal to the number of seeds times the number of unique configurations for each seed. The number of seeds is simply the number of cells in the raster or in a window of the raster when boundary effects are considered. The number of configurations for a patch of given size can be estimated by considering an algorithm that generates unique configurations. The algorithm must generate unique configurations, otherwise the same patch would be counted multiple times.

First, all cells in the raster are numbered sequentially from left-to-right and top-to-bottom. The algorithm grows patches from a seed one cell at a time. At each iteration one cell is chosen from two candidates. Three rules identify candidate cells:

- they must have higher numbers than those in the patch;
- they must be neighbours of the most recently added cell; and
- only edge-linked neighbours are candidates.

For any seed the algorithm can generate many different patches depending on which candidate is chosen at each iteration: a patch configuration results from a sequence of choices. The rules ensure that:

- each patch is unique to one seed - it cannot be generated from another seed because the seed must be the lowest numbered cell in the patch; and
- each patch is unique to one sequence of choices.

Therefore, to count the number of possible configurations it is sufficient to calculate the number of choice sequences.

The seed has four edge-linked neighbours but only two have higher numbers and, therefore, there are only two candidates. Thus, the algorithm can generate two alternative configurations of a two cell patch. Likewise, each new cell generates two candidates. Each addition of a cell doubles the number of configurations, giving four

configurations of a three-cell patch, eight for a four-cell patch and so on. Hence, there are 2^{N-1} configurations of an N cell patch. The process is illustrated in Figure IV.1 which shows the doubling of the number of configurations and the unique sequence of choices leading to each configuration.

The size of the decision space cannot be computed exactly because, for large patches, boundary effects will reduce the number of configurations and because the algorithm cannot generate all possible configurations. However, the size can be estimated at $R*C*2^{N-1}$, where R = number of rows, C = number of columns and N = patch size.

Multi-patch problem.

The decision space is obviously much larger for multi-patch than for single-patch problems. Consider a solution with two patches of total size N on a map with S seeds. The number of possible locations for each patch is S. Each patch can be any size from 1 to N-1. Possible size combinations are 1 and N-1, 2 and N-2 etc. and the number of configurations are: 2^0 times 2^{N-1} ; 2^1 times 2^{N-2} etc. respectively. Thus, the total number of configurations, P, is given by equation II.1 which resolves to equation II.2. The number of possibilities is sufficiently large to make any further calculations for more complex cases superfluous.

$$P = \sum_{i=0}^{N-1} S * 2^i * S * 2^{N-1-i} \quad \text{Equation II.1}$$

$$P = (N-1) * S^2 * 2^{N-1} \quad \text{Equation II.2}$$

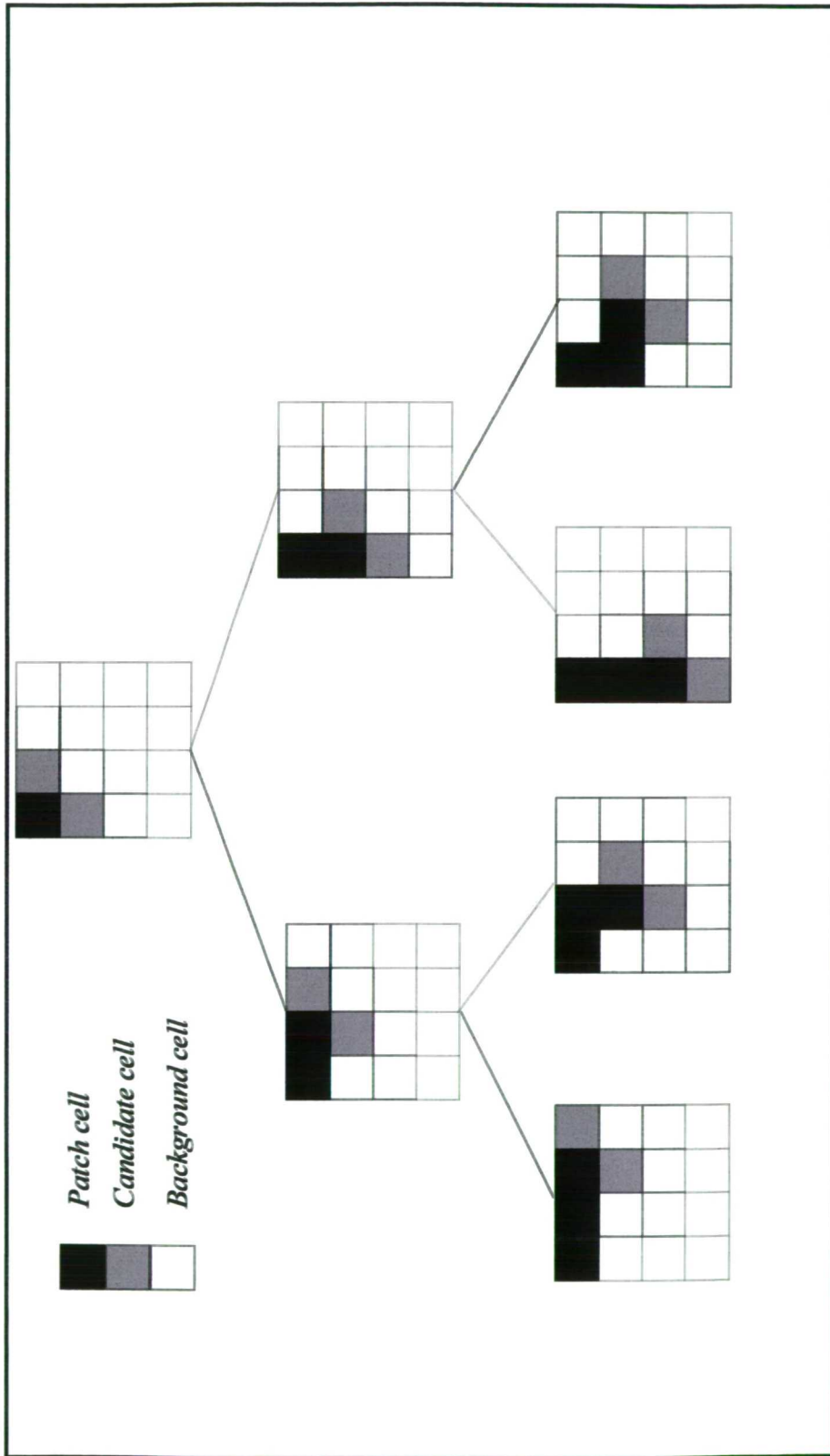


Figure IV.1 Alternative configurations of 3-cell patches generated by the unique-patch generation algorithm: at each iteration there are two candidate cells, so in successive iterations the number of configurations doubles.

List of references

- ANDREY, P., AND TARROUX, P., 1994, Unsupervised image segmentation using a distributed genetic algorithm. *Pattern recognition*, 27, 659-673.
- ARMSTRONG, M.P., AND DENSHAM, P.J., 1992, Domain decomposition for parallel processing of spatial problems. *Computers, Environment and Urban Systems*, 16, 497-513.
- ASPINALL, R., AND VEITCH, N., 1993, Habitat mapping from satellite imagery and wildlife survey data using a bayesian modelling procedure in a GIS. *Photogrammetric Engineering and Remote Sensing*, 59, 537-543.
- BAKER, W.L., 1992, The Landscape Ecology of large disturbances in the design and management of nature reserves. *Landscape Ecology*, 7, 181-194.
- BAKER, W.L., AND CAI, Y., 1992, The rule programs for multiscale analysis of landscape structure using the GRASS geographical information system. *Landscape Ecology*, 7, 291-302.
- BARRETT, G.W., AND PELES, J.D., 1994, Optimizing habitat fragmentation - an agrolandscape perspective. *Landscape and urban planning*, 28, 99-105.
- BENSON, B.J., AND MACKENZIE, M.D., 1995, Effects of sensor spatial-resolution on landscape structure parameters. *Landscape Ecology*, 10, 113-120.
- BHANDARKAR, S.M., ZHANG, Y., AND POTTER, W.D., 1994, An edge detection technique using genetic algorithm-based optimisation. *Pattern recognition*, 27, 1159-1179.
- BROOKES, C.J., 1993, A region growing approach to incorporating shape into multi-criteria evaluation in GIS. MSc Thesis, University of Leicester, Leicester.
- BROOKES, C.J., 1997a, A parameterized region-growing programme for site allocation on raster suitability maps. *International Journal of Geographical Information Science*, 11, 375-396.
- BROOKES, C.J., 1997b, A genetic algorithm for locating optimal sites on raster suitability maps. *Transactions in GIS*, 2, 91-107.
- BUCKLEY, J.J., AND HAYASHI, Y., 1994, Fuzzy genetic algorithm and applications. *Fuzzy Sets and Systems*, 61, 129-136.

- BUECHNER, M., 1987, Conservation in insular parks: Simulation models of factors affecting the movement of animals across park boundaries. *Biological conservation*, 41, 57-76.
- BURROUGH, P.A., 1989, Fuzzy mathematical methods for soil survey and land evaluation. *The journal of soil science*, 40, 477-492.
- BURROUGH, P.A., AND HEUVELINCK, G.B.M., 1992, The sensitivity of boolean and continuous (fuzzy) logical modelling to uncertain data. *Proceedings of the 3rd European Conference on Geographical Information Systems, Munich*.
- CAGAN, J., AND MITCHELL, W.J., 1993, Optimally directed shape generation by shape annealing. *Environment and Planning B*, 20, 5-12.
- CARROLL, J.E., AND LAMBERSON, R.H., 1993, The owl's odyssey. A continuous model for the dispersal of territorial species. *Journal of applied mathematics*, 53, 205-218.
- CARVER, S.J., 1991, Integrating multi-criteria evaluation with geographical information systems. *International Journal of Geographical Information Systems*, 5, 321-339.
- CATCHPOLE, C.K., AND PHILLIPS, J.F., 1992, Territory quality and reproductive success in the Dartford warbler *Sylvia undata* in Dorset, England. *Biological conservation*, 61, 209-215.
- CHEN, J., NEWKIRK, R.T., DAVIDSON, G., AND GONG, P., 1994, The development of a knowledge-based geographical information-system for the zoning of rural areas. *Environment and planning B*, 21, 179-190.
- CHUVIECO, E., 1993, Integration of linear programming and GIS for land-use modelling. *International Journal of Geographical Information Systems*, 7, 71-83.
- CULLINAN, V.I., AND THOMAS, J.M., 1992, A comparison of quantitative methods for examining landscape pattern and scale. *Landscape Ecology*, 7, 211-227.
- D'ANGELO, D.J., HOWARD, L.M., MEYER, J.L., GREGORY, S.V., AND ASHKENAS, L.R., 1995, Ecological uses for genetic algorithms: predicting fish distributions in complex physical habitats. *Canadian journal of fish and aquatic science*, 52, 1893-1908.
- DAVIS, F.W., STOMS, D.M., ESTES, J.E., AND SCEPAN, J., 1990, An information systems approach to the preservation of biological diversity. *International Journal of Geographical Information Systems*, 4, 55-78.

- DAVIS, L.D., 1991, *Handbook of genetic algorithms* (New York: Van Nostrand Reinhold).
- DENSHAM, P.J., 1991, Spatial Decision Support Systems. In: *Geographical Information Systems: Principles and Applications*, edited by MAGUIRE, D.J., GOODCHILD, M.F., and RHIND, D.W. (London: Longman), 403-412.
- DENSHAM, P.J., AND RUSHTON, G., 1992, Strategies for solving large location-allocation problems by heuristic methods. *Environment and planning A*, 24, 289-304.
- DIBBLE, C., 1994, Beyond data: handling spatial and analytical contexts with genetics based machine learning. *Advances in GIS research proceedings, Volume 2. Sixth international symposium on spatial data handling. Edinburgh, Scotland. September 1994.*
- DIBBLE, C., AND DENSHAM, P.J., 1993, Generating interesting alternatives in GIS and SDSS using genetic algorithms. *GIS/LIS Proceedings Minneapolis, Minnesota.*
- DOAK, D.F., MARINO, P.C., AND KAREIVA, P.M., 1990, Spatial scale mediates the influence of habitat fragmentation on dispersal success: implications for conservation. *Theoretical population biology*, 41, 315-336.
- DRUMMOND, J., AND RAMLAL, B., 1992, A prototype uncertainty sub-system implemented in ITC's ILWIS PC based GIS and tested in a Dutch land reallocation project. *Proceedings of the 3rd European Conference on Geographical Information Systems, Munich.*
- EASTMAN, J.R., 1993, *IDRISI: A grid based geographic analysis system. Version 4.1.* (Worcester, Massachusetts: Clark University Graduate School of Geography).
- EASTMAN, J.R., JIN, W., KYEM, P.A.K., AND TOLEDANO, J., 1995, Raster Procedures for Multi-Criteria/Multi-Objective Decisions. *Photogrammetric Engineering and Remote Sensing*, 61, 539-547.
- EASTMAN, J.R., KYEM, P.A.K., TOLEDANO, J., AND JIN, W., 1993, *Explorations in Geographical Information Systems technology, volume 4, GIS and decision making* (Geneva: United Nations Institute for Training and Research).
- EDWARDS, P.J., MAY, R.M., AND WEBB, N.R. (editors), 1993, *Large-Scale Ecology and Conservation Biology* (Oxford: Blackwell Scientific Publications).
- ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, 1992, *ARC/INFO User's Guide* (Redlands, California: ESRI).

- FABRICIUS, C., AND COETZEE, K., 1992, Geographic information systems and artificial intelligence for predicting the presence or absence of mountain reedbuck. *South African journal of wildlife research*, 22, 80-86.
- FAHRIG, L., AND MERRIAM, G., 1985, Habitat patch connectivity and population survival. *Ecology*, 66, 1762-1768.
- FEDOROWICK, J.M., 1993, A landscape restoration framework for wildlife and agriculture in the rural landscape. *Landscape and urban planning*, 27, 7-17.
- FORMAN, R.T.T., AND GODRON, M., 1981, Patches and structural components for a landscape ecology. *BioScience*, 31, 733-740.
- FORMAN, R.T.T., AND GODRON, M., 1986, *Landscape ecology* (London: John Wiley).
- GAME, M., AND PETERKEN, G.F., 1984, Nature reserve selection strategies in the woodlands of central Lincolnshire, England. *Biological conservation*, 29, 157-181.
- GARRIGA, H.M., AND RATICK, S.J., 1996, Simulating Land Use Allocation Decisions with Cellular Automata in Geographical Information Systems. *Proceedings of the 1st International Conference on Geocomputation, Leeds*, 371-384.
- GILPIN, M.E., AND SOULE, M.E., 1986, Minimum Viable Populations: Processes of Species Extinction. In: *Conservation Biology The Science of Scarcity and Diversity*, edited by SOULE, M.E. (Sunderland Massachusetts: Sinauer), 19-34.
- GOLDBERG, D.E., 1989, *Genetic algorithms in search, optimisation, and machine learning* (Reading, Massachusetts: Addison-Wesley).
- GOODCHILD, M.F., SUN, G.Q., AND SHIREN, Y., 1992, Development and test of an error model for categorical data. *International Journal of Geographical Information Systems*, 6, 87-104.
- GROSSBERG, S. (editors), 1988, *Neural networks and natural intelligence* (London: MIT Press).
- GUSTAFSON, E.J., AND PARKER, G.R., 1992, Relationships between landcover proportion and indices of landscape spatial pattern. *Landscape Ecology*, 7, 101-110.
- GUSTAFSON, E.J., AND PARKER, G.R., 1994, Using an index of habitat patch proximity for landscape design. *Landscape and urban planning*, 29, 117-130.

- HANSKI, I., AND THOMAS, C.D., 1994, Metapopulation dynamics and conservation: a spatially explicit model applied to butterflies. *Biological conservation*, 68, 167-180.
- HANSSON, L., AND ANGELSTRAM, P., 1991, Landscape ecology as a theoretical basis for nature conservation. *Landscape Ecology*, 5, 191-201.
- HENEIN, K., AND MERRIAM, G., 1990, The elements of connectivity where corridor quality is variable. *Landscape Ecology*, 4, 157-170.
- HIBBERD, B.G. (editors), 1991, *Forestry Practice* (London: HMSO).
- HOBBS, M.H.W., 1994, Analysis of a retail branch network: A problem of catchment areas. *Proceedings of the GIS research UK 1994 conference, Leicester*, 31-39.
- HOBBS, R.J., SAUNDERS, D.A., AND ARNOLD, G.W., 1993, Integrated landscape ecology: a Western Australian perspective. *Biological conservation*, 64, 231-238.
- HOLLAND, J.H., 1975, *Adaptation in natural and artificial systems* (Michigan: University of Michigan press).
- JANKOWSKI, P., 1995, Integrating geographical information systems and multiple criteria decision-making methods. *International Journal of Geographical Information Systems*, 9, 251-273.
- JIANPING, X., AND LATHROP, R.G., 1995, Improving simulation accuracy of spread phenomena in a raster-based Geographic Information System. *International Journal of Geographical Information Systems*, 9, 153-168.
- JOHANSSON, K., 1995, Fragmentation index as a region based GIS operator. *International Journal of Geographical Information Systems*, 9, 211-220.
- KINNEAR, K.E. (editor), 1994, *Advances in genetic programming* (London: MIT Press).
- KIRKPATRICK, J.B., 1983, An iterative method for establishing priorities for the selection of nature reserves: an example from Tasmania. *Biological conservation*, 25, 127-134.
- KNAAPEN, J.P., SCHEFFER, M., AND HARMS, B., 1992, Estimating habitat isolation in landscape planning. *Landscape and urban planning*, 23, 1-16.
- KOZA, J.R., 1992, *Genetic Programming* (Cambridge, Massachusetts: MIT).
- KOZA, J.R., 1994, Introduction to genetic programming. In: *Advances in genetic programming*, edited by KINNEAR, K.E. (London: MIT Press), 21-42.

- KRZANOWSKI, R.M., 1997, *Evaluation of Spatial Evolutionary Algorithm for Spatial Modelling*. PhD thesis, University of London, London.
- LAGRO, J., 1991, Assessing patch shape in landscape mosaics. *Photogrammetric Engineering and Remote Sensing*, 52, 285-293.
- LAMBERSON, R.H., NOON, B.R., VOSS, C., AND MCKELVEY, K.S., 1994, Reserve design for territorial species: the effects of patch size and spacing on the viability of the Northern Spotted Owl. *Conservation Biology*, 8, 185-195.
- LAURANCE, W.F., 1991, Edge effects in tropical forest fragments: Application of a model for the design of nature reserves. *Biological conservation*, 57, 205-219.
- LAURANCE, W.F., AND YENSEN, E., 1991, Predicting the impacts of edge effects in fragmented habitats. *Biological conservation*, 55, 77-92.
- LEADER-WILLIAMS, N., HARRISON, J., AND GREEN, M.J.B., 1990, designing protected areas to conserve natural resources. *Science*, 74, 189-204.
- LEDUC, A., PRAIRIE, Y.T., AND BERGERON, Y., 1994, Fractal dimension estimates of a fragmented landscape - sources of variability. *Landscape Ecology*, 9, 279-286.
- LEUNG, Y., 1988, *Spatial analysis and planning under imprecision* (Amsterdam: Elsevier science publishers).
- LOMOLINO, M.V., 1994, An evaluation of alternative strategies for building networks of nature reserves. *Biological conservation*, 69, 243-249.
- LORD, J.M., AND NORTON, D.A., 1990, Scale and the spatial concept of fragmentation. *Conservation Biology*, 4, 197-202.
- MAGUIRE, D.J., GOODCHILD, M.F., AND RHIND, D.W. (editors), 1991, *Geographical Information Systems. Principles and Applications* (Harlow: Longman).
- MARGULES, C.R., AND STEIN, J.L., 1989, Patterns in the distribution of species and the selection of nature reserves: An example from Eucalyptus forests in South-eastern New South Wales. *Biological conservation*, 50, 219-238.
- MCGARIGAL, K., AND MARKS, B.J., 1994, *FRAGSTATS Spatial Pattern Analysis Program for Quantifying Landscape Structure*. Forest Science Department, Oregon State University.
- MCKENDRY, J.E., AND MACHLIS, G.E., 1993, The role of geography in extending biodiversity gap analysis. *Applied Geography*, 13, 135-152.

- MCKENZIE, N.L., BELBIN, L., MARGULES, C.R., AND KEIGHERY, G.J., 1989, Selecting representative reserve systems in remote areas: A case study in the Nullarbor Region, Australia. *Biological conservation*, 50, 239-261.
- MINCH, R.P., AND SANDERS, G.L., 1986, Computerised information systems supporting multicriteria decision making. *Decision sciences*, 17, 395-413.
- MORRISON, M.L., MARCOT, B.G., AND MANNAN, R.W., 1992, *Wildlife habitat relationships* (Wisconsin: University of Wisconsin).
- MOSETTI, G., POLONI, C., AND DIVIACCO, B., 1994, Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*, 51, 105-116.
- MURPHY, D.D., FREAS, K.E., AND WEISS, S.B., 1990, An environment-metapopulation approach to population viability analysis for a threatened invertebrate. *Conservation Biology*, 4, 41-51.
- NICHOLLS, A.O., AND MARGULES, C.R., 1993, An upgraded reserve selection algorithm. *Biological conservation*, 64, 165-169.
- O'NEILL, R.V., GARDNER, R.H., TURNER, M.G., AND ROMME, W.H., 1992, Epidemiology theory and disturbance spread in landscapes. *Landscape Ecology*, 7, 19-26.
- OPENSHAW, S., AND ABRAHART, R.J., 1996, Geocomputation. *Proceedings of the 1st international conference on GeoComputation*, 2, 665-666.
- PEREIRA, J.M.C., AND DUCKSTEIN, L., 1993, A multiple criteria decision-making approach to GIS-based land suitability evaluation. *International Journal of Geographical Information Systems*, 7, 407-424.
- PHAM, D.T., AND ONDER, H.H., 1992, A knowledge-based system for optimizing workplace layouts using a genetic algorithm. *Ergonomics*, 35, 1479-1487.
- POTTS, J.C., GIDDENS, T.D., AND YADAV, S.B., 1994, The development and evaluation of an improved genetic algorithm based on migration and artificial selection. *IEEE transactions on systems man and cybernetics*, 24, 73-86.
- PRESSEY, R.L., AND NICHOLLS, A.O., 1989, Application of a numerical algorithm to the selection of reserves in semi-arid New South Wales. *Biological conservation*, 50, 263-278.
- PROBST, J.R., AND WEINRICH, J., 1993, Relating Kirtlands warbler population to changing landscape composition and structure. *Landscape Ecology*, 8, 257-271.

- QUINN, J.F., AND HARRISON, S.P., 1988, Effects of habitat fragmentation and isolation on species richness: evidence from biogeographic patterns. *Oecologia*, 75, 132-140.
- RIBEIRO FILHO, J.L., TRELEAVEN, P.C., AND ALIPPI, C., 1994, Genetic algorithm programming environments. *Computer*, 27, 28-43.
- ROTH, G., AND LEVINE, M.D., 1994, Geometric primitive extraction using a genetic algorithm. *IEEE transactions on pattern analysis and machine intelligence*, 16, 901-905.
- RUSSELL, R.W., HUNT, G.L., COYLE, K.O., AND COONEY, R.T., 1992, Foraging in a fractal environment - spatial patterns in a marine predator prey environment. *Landscape Ecology*, 7, 195-209.
- SAUNDERS, D.A., AND HOBBS, R.J. (editors), 1991, *Nature conservation 2: The role of corridors* (Chipping Norton, NSW: Surrey Beatty & Sons).
- SAUNDERS, D.A., HOBBS, R.J., AND MARGULES, C.R., 1991, Biological consequences of ecosystem fragmentation. *Conservation Biology*, 5, 18-32.
- SCHILDT, H., 1991, *C++ the complete reference* (Berkeley: McGraw-Hill).
- SELMAN, P., AND DOAR, N., 1992, An investigation of the potential for landscape ecology to act as a basis for rural land use plans. *Journal of Environmental Management*, 35, 281-299.
- SHANNON, C.E., AND WEAVER, W., 1962, *The mathematical theory of communication* (Urbana: University of Illinois Press).
- SHARIFI, M.A., 1992, A decision support system for land use planning development. *Proceedings of the 3rd European Conference on Geographical Information Systems*, 118-126.
- SHARIFI, M.A., 1993, GIS in support of optimal land use allocation for agricultural development. *Proceedings of the 4th European Conference on Geographical Information Systems*, Genoa.
- SHRADER-FRECHETTE, K.S., AND MCCOY, E.D., 1994, What ecology can do for the environment. *Journal of Environmental Management*, 41, 293-307.
- SMITH, P.G.R., AND THEBERGE, J.B., 1986, A review of criteria for evaluating natural areas. *Environmental management*, 10, 715-734.

SONKA, M., HLAVAC, V., AND BOYLE, R., 1993, *Image processing, analysis and machine vision* (London: Chapman and Hall).

SOULE, M.E., AND GILPIN, M.E., 1991, The theory of wildlife corridor capability. In: *Nature conservation 2: the role of corridors*, edited by SAUNDERS, D.A., and HOBBS, R.J. (Chipping Norton, NSW: Surrey Beatty and Sons), .

SPRAGUE JR, R.H., AND WATSON, H.J. (editors), 1993, *Decision support systems* (New Jersey: Prentice Hall).

SRIKANTH, R., GEORGE, R., WARSI, N., PRABHU, N., PETRY, F.E., AND BUCKLES, B.P., 1995, A variable-length genetic algorithm for clustering and classification. *Pattern recognition letters*, 16, 789-800.

SYDELKO, P.J., LI, Z., SUNDELL, R.C., AND MAJERUS, K.A., 1997, Integrated dynamic landscape analysis and modeling system (IDLAMS). *Proceedings of the GIS/LIS conference, Cincinnati Ohio* .

TAMIZ, M. (editors), 1996, *Multi-Objective Programming and Goal Programming* (Heidelberg: Springer-Verlag).

THEBERGE, J.B., 1989, Guidelines to drawing ecologically sound boundaries for national parks and nature reserves. *Environmental management*, 13, 695-702.

THEOBALD, D.M., AND GROSS, M.D., 1994, EML: A modelling environment for exploring landscape dynamics. *Computers, Environment and Urban Systems*, 18, 193-204.

TOMLIN, C.D., 1990, *Geographic information systems and cartographic modelling* (New Jersey: Prentice Hall).

TURNER, M.G., ROMME, W.H., GARDNER, R.H., O'NEILL, R.V., AND KRATZ, T.K., 1993, A revised concept of landscape equilibrium: Disturbance and stability on scaled landscapes. *Landscape Ecology*, 8, 213-227.

UNDERHILL, L.G., 1994, Optimal and suboptimal reserve selection algorithms. *Biological conservation*, 70, 85-87.

VAN GAANS, P.F.M., AND BURROUGH, P.A., 1993, The use of fuzzy logic and continuous classification in GIS applications: a review. *Proceedings of the 4th European Conference on Geographical Information Systems, Genoa*.

WAGNER, H.M., 1972, *Principles of Operations Research* (London: Prentice Hall).

WANG, F., 1994, The use of artificial neural networks in a geographical information system for agricultural land-suitability assessment. *Environment and planning A*, 26, 265-284.

WHITLEY, D., 1994, A genetic algorithm tutorial. *Statistics and Computing*, 4, 65-85.

WITH, K.A., 1994, Using fractal analysis to assess how species perceive landscape structure. *Landscape Ecology*, 9, 25-36.

XIANG, W.N., 1993, A GIS/MMP-based coordination model and its application to distributed environmental planning. *Environment and Planning B*, 20, 195-220.