FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Hybrid Optimizer for Expeditious Modeling of Virtual Urban Environments

Filipe Manuel Miranda da Cruz

Submited to the Faculty of Engineering of the
University of Porto for obtaining the Masters degree in
Informatics Engineering

Under the supervision of:
António Fernando Coelho
Auxiliary Professor at the Faculty of Engineering of the University of Porto

Under the co-supervision of:
Luis Paulo Reis
Auxiliary Professor at the Faculty of Engineering of the University of Porto

September 2008

# Abstract

There has been a growing need for expeditious modeling systems for urban environments in recent years. Applications include virtual city tours, location based services, urban planning, among others.

Creating urban models that are accurate representations of the real world can be extremely difficult. Expeditious modeling typically suffers from a lack of complete and reliable data to generate the models or requiring too much time from skilled professionals. An often acceptable compromise is the use of procedural modelling where sets of rules with results bearing close resemblance to the real world are applied to generate the virtual environment.

Parametrizing procedural modelling rules is still an open problem. The work described in this thesis focuses on solving this with the development of a hybrid meta heuristic algorithm capable of optimizing the parameters to generate environments approximating to known data. These calibrated rules can then be applied to generate other sections of the model containing missing data.

The meta heuristic developed adapts ideas from several nature based meta heuristic algorithms such as genetic algorithms, simulated annealing and harmony search. The new hybrid meta heuristic is configurable in an attempt to avoid typical meta heuristic optimum search pitfalls such as the constraint to a local optimum or long tail behaviour. The optimization algorithm was developed to solve parameterization problems of both discrete and continuous parameter types.

To prove the concept of the system, as a viable solution in a neglected field of application, this document presents several benchmark tests offering competitive performance results under a standard batch of non-convex non-linear continuous parameter functions. Additionaly a succesful real case application scenario was carried out with data from the streets of Porto. Parameters of certain modeling rules were automatically calibrated by using data from a small housing section from Rua do Almada. These rules

were re-applied in other sections of Rua do Almada with similar housing characteristics. These sections lacked complete data for a realistic modeling, but, by re-applying the previously calibrated modeling rules, a realistic modeling of these sections was obtained.

# Resumo

Recentemente tem-se sentido uma crescente necessidade de desenvolvimento dos sistemas de modelação expedita de ambientes urbanos. As aplicações destes sistemas incluem visitas virtuais, serviços baseados na localização, planeamento urbano, entre outros.

A geração de modelos urbanos que representam fielmente a realidade pode ser algo extremamente dificil de alcançar em sistemas de modelação expedita. Sofrendo dos típicos problemas de falta de conjuntos de dados que permitam uma geração integral dos modelos, ou a necessidade de empregar trabalho de profissionais qualificados durante longos períodos de tempo. Um compromisso geralmente aceitável para atingir resultados suficientemente realísticos, evitando estes problemas, surge com a aplicação de regras de modelação procedimental.

No entanto a determinação da melhor parameterização das regras de modelação procedimental continua a ser um problema. O trabalho descrito nesta tese pretende dar solução a este problema através do desenvolvimento de um algoritmo meta heurístico híbrido capaz de optimizar os parâmetros das regras de geração de modo a que os modelos gerados se assemelhem bastante com a realidade. As regras calibradas pelo optimizador através de comparações com dados conhecidos podem depois ser re-aplicadas na geração de outras zonas com falta de dados necessários.

O algoritmo meta heurístico desenvolvido adapta ideias de vários outros algoritmos baseados em processos naturais como por exemplo os algoritmos genéticos, o arrefecimento simulado e a pesquisa harmónica. O algoritmo híbrido desenvolvido é configurável, de modo a poder evitar problemas típicos em algoritmos de optimização como sendo a restrição em zonas de óptimos locais e o comportamento de cauda longa. O algoritmo permite abordar problemas com parâmetros quer do tipo discreto quer do tipo contínuo.

Apoiando o sistema como uma solução viável num campo de aplicação negligenciado,

neste documento apresentam-se testes de comparação de performance, com resultados satisfatórios, que foram aplicados a conjuntos de problemas padrão do tipo contínuo, não convexo e não linear. Adicionalmente é apresentado um caso real de aplicação, com sucesso, baseado em dados das ruas do Porto. Os parâmetros de certas regras de modelação foram calibrados automaticamente usando dados de um pequeno conjunto de habitações da Rua do Almada. Estas regras foram re-aplicadas em outras zonas dessa que possuíam características similares. Estas zonas tinham falta de dados para realizar a modelação, mas, aplicando as regras de modelação calibradas previamente, foi possível modelar realisticamente estas zonas.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This chapter presents an introduction to the contents of this thesis. It addresses the usefulness of modeling virtual urban environments in general and the importance of expeditious modeling methods in particular. This introduction also presents the reader to certain fields of artificial intelligence and how they can assist in solving certain problems associated with expeditious modeling. This chapter also states the thesis objectives and how the document is structured through the following chapters.

## 1.1 Expeditious modeling of Virtual Urban Environments

Virtual representations of scenes and objects have been thriving through the last decade accompanying the technology growth of computing in general and 3D modeling in particular. [CBSF07]

Being capable of accurately representing the cities we live in became a standard requirement for many applications in the fields of historic reconstruction, location based services and immersive interaction.

One of the main constraints from the use of virtual urban environments is the requirement to previously model scene elements based on the real world by using 3D modeling software. Using these programs requires access to skilled manpower for plenty of hours. Additionally, to create realistic modeling of urban environments with heavier complexity, such as cities for example, the required time and resources increases dramatically.

Expeditious modeling systems surged as an answer to the difficulties of manually creating and modifying models that typically share common features. These systems follow user defined rules to generate the geometry of the different elements of the 3D scene, avoiding the requirement of manual modeling the elements by professionals using standard 3D modeling tools.

The clear advantage of expeditious modeling over traditional methods is in enabling changes to the environment to take place without requiring a trained professional to reshape the objects and execute time-intensive rendering. Instead, a few rules can be altered and the entire environment quickly re-rendered.

The use and importance of expeditious 3D modeling of virtual urban environments has increased significantly in the past years. Still many difficulties are associated with the technology. Generating appropriate production rules with fine tuned parameters, or the resources required in collecting georeferenced data for example, can easily bottleneck the potential of the modeling system in achieving realistic results.

## 1.2 Applied Artificial Intelligence

Artificial intelligence is a term originally coined to represent the science and engineering of intelligent machines [MMRS55]. In current days the term is also applied to reference the study and development of intelligent or autonomous agents, "computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they were designed" [Mae95] [Rei03].

Artificial intelligence bridges the areas of computer science, psychology, philosophy, neuroscience, cognitive science, linguistics, operations research, economics, control theory, probability, optimization and logic. Application fields involving artificial intelligence include robotics, intelligent agents, multi-agent interaction, swarm behavior, knowledge discovery and data mining, speech and facial recognition, control systems, optimization algorithms, scheduling, etc.

One of the fields from artificial intelligence with particular interest to this thesis is the research of meta heuristic algorithms for solving optimum search problems of combinatorial nature. A heuristic is commonly defined as a method developed to solve a given type of problem. A meta heuristic is an heuristic method developed to solve the problem of problem solving. In layman terms, meta heuristics are algorithms that try to find the best combination to solve a given combinatorial problem.

A combinatorial problem can be roughly defined as a black box system with a certain number of parameter inputs that will execute a simulation to generate a result. This result is evaluated by a quality function. The combination of parameter inputs that returns the best possible quality function result is considered the optimum solution.

The purpose of meta heuristic algorithms is to find the optimal solution in the least amount of simulations. To achieve this objective, different meta heuristic algorithm behaviors were conceptualized and improved through the years, such as random search, hill climbing search, simulated annealing, tabu search, genetic algorithms, harmony search, neural networks, particle swarm and ant colony based optimization. The performance of the different algorithms in solving a given problem type largely varies depending on its complexity in terms of convergence of the quality function.

Other sub-fields of artificial intelligence with interest to this thesis include knowledge discovery, data mining and feature extraction. Research has been pursued in these fields regarding clustering algorithms and smart extraction of useful information.

## 1.3 Problem Statement

Expeditious modeling systems are based on the application of production rules to generate a geometric model from geographically referenced data. Several implementations of expeditious modeling systems can be found in the market, many of them developed to solve specific problems by research teams. Expeditious modelers have applications in the fields of cultural heritage, virtual city tour, computer graphics for movie animations, military simulations, serious games, etc.

Defining and determining the specific set of rules to generate a certain urban environment can become a problem. Different methods can be applied to determine the production rules. They can be based on empirical definitions or acquired by observing how 3D modeling people model the objects. Especially in the field of urban environments, the knowledge from architects and civil engineers is essential in defining good production rules. There still is a need to correctly translate the human knowledge into production rules. Finetuning the parameters of these production rules is also a problem, often requiring mathematical and logical knowledge of the system and many hours of trial and error. Depending on the complexity, manually adjusting the parameters of production rules for optimum results can be extremly difficult.

Another problem inherent to the system is the need to already have or acquire geographical referenced data. The data can often be non-existent or require unavailable resources to gather. Sampling from the target area, for example, can greatly reduce the complexity of the acquiring process, but in many cases accurate data will not ever be accessible. This situation reduces the capacity of expeditious modeling systems to solve certain problems.

## 1.4 Research Objectives

Fields of artificial intelligence, such as optimization algorithms and knowledge data mining, can run simulations in search of optimum parameters and estimate data where data is not obtainable. Applying these concepts to expeditious modeling systems can drastically improve their capacity, performance and automation.

One aim is to develop a competitive meta heuristic algorithm capable of enhancing functionality to an existing expeditious modeling framework. To certify performance competitiveness of the algorithm, the thesis should document a comparison of results of the developed solution with other optimization algorithms documented in the literature.

The system should also be able to discover an optimum solution for the unknown parameters of a set of rules that will generate the georeferenced geometry and apply these newly parametrized rules to other georeferenced areas lacking complete data to generate realistic virtual models.

As proof of concept the thesis will present an implementation embedded in an existing expeditious modeling system and showcase a real case scenario with data obtained from the city of Porto.

## 1.5 Document Structure

This chapter contains an introduction to the document. It refers to areas of knowledge involved in the thesis, a summary description of the problem and how it can be approached. It ends with a short section describing the document structure.

The second chapter contains a short overview and references to the state of the art of 3D modeling systems in general and expeditious modeling systems for virtual urban environments in particular, focused on its applications and the most relevant reference work in the field. An additional section is dedicated to the XL3D modeling system due to its role in the thesis.

The third chapter refers to the state of the art of applied artificial intelligence, vaguely covering the field of knowledge discovery and data mining and strongly focusing on stochastic optimization and evolutionary computation. Special focus is given to the limitations of standard local search meta heuristic algorithms and the advantages of hybrid harmony search and real coded genetic algorithms.

The fourth chapter is devoted to the problem statement and architecture of the implemented solution. This chapter describes the standard problems and limitations of expeditious modeling systems of virtual urban environments, paying particular emphasis to the drive of the thesis: developing a hybrid meta and hyper heuristic optimizer system capable of automatically optimize the parameters of different types of expeditious modeling rules. This chapter also lists the requirements of such a system, the constraints it implies on the implementation and how to configure it for a standard test. Additionaly, this chapter documents the main internal architecture of the developed system.

The fifth chapter refers to test and evaluation of the proposed methodology. The first part refers to a series of tests for comparative performance results of the meta heuristic algorithm developed, benchmarking the performance of the algorithm against a set of well documented global optimization mathematical equations. The second part contains a proof of concept where the meta heuristic algorithm under different possible configurations was applied to calibrate parameters of a given production rule generating environments in the XL3D expeditious modeling system.

The sixth chapter presents the conclusions of this thesis. A short summary of what was achieved including a reflection to in what way the development of this thesis contributed to the state of the art in the covered area of application. A few additional words present a short selection of ideas envisioned for further research and development in the area of application in general and the project covered by this thesis in particular.

# Chapter 2

# Expeditious Modeling of Urban Environments

Expeditious Modeling has grown into large importance in recent years. Especially applied in the generation of realistic urban environments, the applications and experimental groundwork to automate the technology have been immense. This chapter aims to present a definition of what expeditious modeling of urban environments is, where it came from, its importance and current state of the art. Also addressed are the three fields of different methodologies used to generate realistic urban models: 3D modeling tools, image analysis and procedural modeling. This chapter lists some reference work, doing a brief comparative analysis and exposing the key problems with the methodologies related to this thesis. A more in depth description of the XL3D modeling system is also presented in a final sub-chapter, having been given more attention due to its involvement in the work developed for this thesis.

## 2.1 3D Modeling

In generic terms 3D modeling is a term coined for generating and shaping virtual representations of 3D objects and scenes. With the progress of standard computation technology, virtual representations of the real world thrived from the basic wireframe 3D vector graphics to low polygon object models with flat or smooth shading as seen in the left side of Figure 2.1 into heavily complex and realistic scenes with huge quantity of vertices and polygons, offering more detail to 3D objects as examplified in the right side of Figure 2.2.

Figure 2.1: Model of a smooth shaded duck.

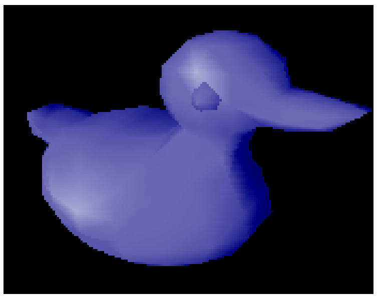
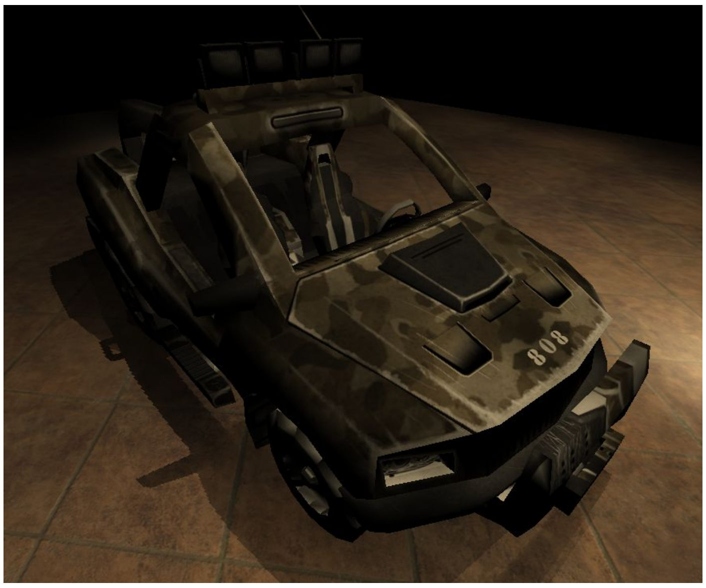


Figure 2.2: Model of an assault vehicle.



Computer display adapters date back to the 1960s when the first video monitors were developed. It would still take years of advances in the field of micro processors to enabled 3D acceleration floating point graphics processor units (GPU) to enter the user market in the late 1990s. Hardware 3D acceleration graphics cards became a standard for most desktop computer in the late nineties. Similarly, new models of laptop computers from recent years all have 3D acceleration chips inbuilt. Even latest models of mobile technologies such as mobile phones and PDAs are now embedding 3D acceleration chips to their hardware.

These advances have converged to allow faster real time rendering of virtual 3D scenes, enabling higher realism to be incorporated in real-time applications with every passing year taking advantage of the latest rendering pipeline hardware capacity.

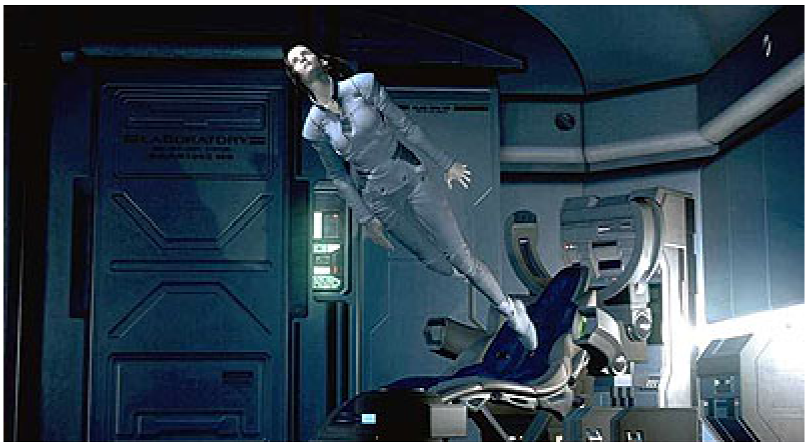


Figure 2.3: Screenshot of the computer animated movie ”Final Fantasy: The Spirits Within”.

Still, generating realistic 3D data with high resolution and attention to detail can be quite time consuming and require highly trained people using expensive 3D modeling tools. Many thriving industries however demand this level of high quality 3D modeling at low cost and quick creation period.

An example is the ever growing industry of computer video games development. Titles of best-selling genres such as first person shooters, sports simulations and massive multi player on-line role playing games have grown into multi million euro entertainment selling industries. Most of the high profile video games sold require real time rendering of highly realistic 3D modeling.

Another industry in demand for high quality 3D modeling is the so called serious games industry. In serious games, applications based on the same technologies as computer video games, are developed especially aimed to assist in training people, testing reactions and skills, or confronting users with specific scenarios. The use of these applications ranges amply including military simulation, specialized training, or treating certain phobia pathologies.

A non real-time application of 3D modeling can be found in the movie industry. Advances in motion tracking enabled layered objects and special effects to fit harmoniously into a pre-recorded scene. Some popular animation movies have been completely made solely using 3D virtual worlds and characters, some of which achieving extremely realistic results as seen in Figure 2.3, a screenshot of the movie ”Final Fantasy: The Spirits Within” dating back to 2001.

A slightly different approach to film making that uses video game engines is coined as machinima [Mar04]. Mostly consisting of independent short film movie makers who have discovered a healthy budget cut in using video game engines to direct their movies and animated series.

## 2.2 Expeditious Modeling of Virtual Urban Environments

Even if visualizing the 3D model is already fast enough to present realistic camera motion of highly complex scenes, the data for these scenes still needs to be pre-generated accurately enough to match the real world. Creating a realistic 3D model of the real world requires a heavy amount of human and logistic resources. In this sense, expeditious modeling systems can offer invaluable help to projects with tight budgets, schedules or special restrictions.

Expeditious modeling is a term referring to fast generation of 3D models. Systems capable of generating complex models from production rules and prototypes are an example. There are many intrinsic difficulties inherent to expeditious modeling systems. The most common are: obtaining an accurate 2D topography; gathering geographically referenced modeling data; defining modeling rules capable of generating areas out of essential missing information; calibrating the modeling rules parameters.

Getting an accurate 2D topology of a city can be troublesome, even though 2D maps are often commonly available, their level of detail usually does not go much beyond separating streets, building and garden boundaries. Crucial additional details may well be unexistant but required to realistically render a model of the city. Verifying data on location can become a heavy burden in terms of resources. Municipalities may often have such data already collected in a semi regular basis but tend to be typically burocratical towards allowing free and full access.

Additionally to acquiring a 2D topography, it's necessary to gather its georeferenced data, not only the geographical delimitations of the different urban elements but also their assigned characteristics. A road can be of different types: dirt, stone paved, freeway. Similarly, buildings can have many different characteristics associated with them in terms of composition, including everything from the simple height of the building to its architectural composition. This kind of information is much harder to obtain than the simple topology and is also more prone to changes through time.

Modeling large areas without a complete georeferenced set of information becomes one of the most important challenges for expeditious modeling. Correctly estimating data to replace missing information is vital to ensure the realism of the virtual model.

Another difficulty lies in defining the set of rules that will be applied to generate each object of the virtual scene. Creating non calculus intensive rules that enable enough detail and realism to show in the urban model requires some base knowledge in architecture and optimized 3D modeling. Calibrating parameters for the set of rules applied is a difficult task at hand.

These are the difficulties an expeditious modeling system must avoid, but to create it there are several approaches to obtaining a 3D model, these are:

1. Generating the data manually using 3D modeling tools;
2. Acquiring data through image analysis;
3. Procedural modeling.

### 2.2.1 3D Modeling Tools

Generating the 3D data manually can be achieved with the use of specially designed software tools. These are typically referred to as 3D modeling tools. Their matured development allows good results but they require expertize which is often a big detering factor for less savvy computer users. To accomplish even relatively simple modeling tasks the user is required to really get to know the tool first, often requiring a few sessions of manual reading and tutorial follow-ups.

Modeling complex objects with high attention to detail can grow extremely time consuming even for experienced users. Many technological developments have been implemented in the 3D modeling tools available in the market that attempt to facilitate usability and handling of complex scenes. Distributing objects by layer in a scene for example can help focus on the important details while quickly shutting off from the unimportant parts. Instantiating the same object over distinct locations instead of duplicating its geometry is another example of a way to save memory and speed up rendering time.

Despite some, among many more, of these advances in 3D modeling tools, to model a realistic city, for example, still requires a team of expert professionals and plenty of work hours.

Popular freeware systems like POV-Ray developed in the early 1980's paved the way for more demanding commercial applications such as Autodesk 3D Studio Max, NewTek Lightwave 3D, Maxon Cinema 4D, DAZ 3D Bryce and Autodesk Maya, just to name a few most prominent names. Other software tools such as Nevercenter Silo, Pixologic ZBrush, Luxology Modo and Blender also have a place in the market with a niche demand for cheaper licenses, simplified usability and specified field of application.

It's beyond the scope of this thesis to dwell much deeper into what 3D modeling tools are available in the market and comparing them in terms of performance, complexity, usability, price, support, etc. Will suffice to note that many exist, are amply available, capable of great results given a sufficient amount of training work hours and skilled tutoring, but will not play any role in the work to be described in the following chapters. More comprehensive studies can easily be found on the internet comparing the most popular solutions. Two worth mentioning links are an article by Benoit Saint-Moulin entitled *3D applications 2007 comparisons table* [1] and a communitary wiki page *Comparison of 3D tools* at CGSociety website [2].

### 2.2.2 Image Analysis

A way to obtain 3D model data automatically is by analysing and interpreting 2D images of the real world. A well active part of modern science literature refers new progresses in acquiring 3D data out of 2D still and motion imagery. This field of research is often referred to as computer vision. Some of its applications include object recognition, eyes and face detection, fingerprint comparison, assisted medical diagnostics, military simulations, automated guiding systems, etc.

Beyond acquiring or recognizing 3D objects from 2D still images, a connected field of research referred to as motion tracking shares similar principles by recognizing key points and 3D objects from video motion frames. Both of these fields are still thriving, their references in the literature date back to little more then a couple decades, with big developments being documented every year.

One of the hardest difficulties the robustness of these processes face is the dynamic variation of the lightning conditions from the real world. Scenes that look the same under a human vision accustomed to rapidly variable lighting conditions, can be very hard to compare to a computer algorithm processing a digital signal.

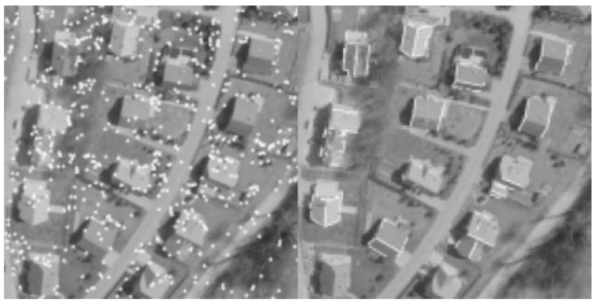
[1]http://www.tdt3d.be/articles_viewer.php?art_id=99

[2]http://wiki.cgsociety.org/index.php/Comparison_of_3d_tools

The work of Pollefeys [PPK+98] demonstrates an application in reconstructing archaeological sites in virtual reality. This is achieved by using automatic 3D surface acquisition systems over the Roman site of Sagalassos in Southwest Turkey.

Other examples dating to the same period come from Faugeras [FLR+95] and Gruen [GW98] with a higher focus on acquiring city buildings topology from aerial pictures. These studies refer to the basis of 3D model extraction from 2D imagery by identifying key points and attempting to group them to form faces of planar regions as can be seen in Figure 2.4. These techniques offered some acceptable result but still required heavy supervision and human interaction. More recently, the work of Zedebin [ZKGGK06] still pursues automatic 3D modeling of cities by aerial topographic imagery analysis.

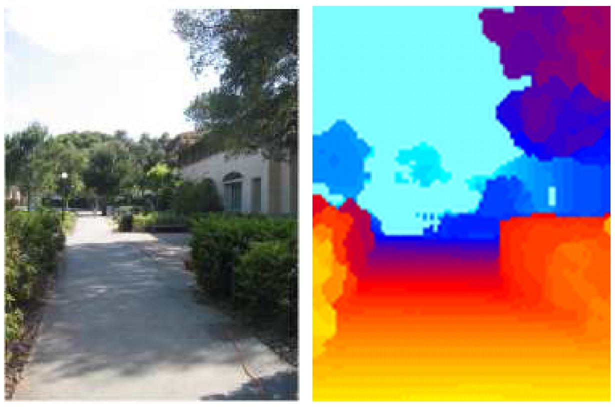
Figure 2.4: Screenshots of key point acquiring and planar regions determination [FLR+95].



Another strong reference in the area of image analysis is the thesis of Debevec "Modeling and Rendering Architecture from Photographs" [Deb96]. Presenting an approach for modeling and rendering existing architectural scenes from sparse sets of still photographs. The authors modeling approach combines both geometry-based and image-based techniques in two components. "The first component is an interactive photogrammetric modeling method which facilitates the recovery of the basic geometry of the photographed scene. The photogrammetric modeling approach is effective, convenient, and robust because it exploits the constraints that are characteristic of architectural scenes. The second component is a model-based stereo algorithm, which recovers how the real scene deviates from the basic model." The author claims that by making use of the model, the technique could robustly recover accurate depth from widely-spaced image pairs allowing it to model large architectural environments with far fewer photographs than image-based modeling approaches.

More recent advances in depth recognition in monocular images by Saxena [SSA07] have equally pushed the envelope to automatic 3D object reconstruction. With the use of a Markov Random Field, Saxena has proved possible to obtain the 3D locations and orientation of several small planar regions as can be seen in Figure 2.5.

Figure 2.5: Screenshots of monocular 3D reconstruction [SSA07].



## 2.2.3 Procedural Modeling

Despite the high quality capacity of some of the presented automated solutions for acquiring 3D data out of 2D imagery, most of them still require human interaction or supervision at some level in order to achieve optimum results.

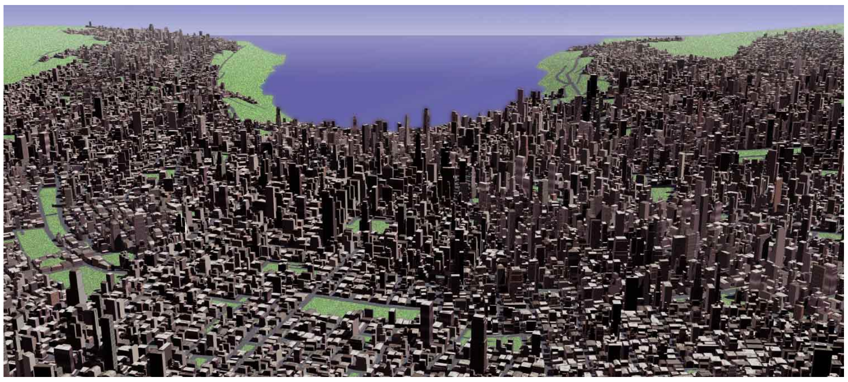
Procedural modeling is a term refering to solutions that typically apply modeling rules to georeferenced data in order to automate and simplify the 3D modeling process. Production rules can be applied to georeferenced data to generate complete urban environments with roads, gardens and all sorts of different types of buildings.

For example, 3D prototypes can be modeled manually by an expert with 3D modeling tools and get invoked by a procedural modeling process. This can be useful to clone reoccurring objects or for replacing more important buildings with versions containing greater complexity only achieved through manual modeling.

Other procedural modeling systems imply obtaining data by image analysis processes already discussed previously in this chapter. This data can partially construct an object and define guidelines that may facilitate the remaining reconstruction process.

An highly interesting example of one of these techniques can be found in the work of Shlyakhter et al [SRDT01]. The realistic modeling of vegetation adds a significant dimension of realism to a scene. Trees, in particular, form an integral part of architectural landscapes. Shlyakhter et al developed a procedural modeling method

Figure 2.6: Screenshot of large generated city [PM01].



to create 3D tree model resembling a specific foliaged tree from a set of instrumented photographs.

The system is capable of extracting the basic structure of a tree from image analysis of several photographs taken under different perspectives. Having the basic structure the system applies L-System concepts originally documented by Lindenmayer [Lin68] and Prusinkiewicz [Pru86] to produce branches and foliage with some resemblance to the original.

The work of Parish [PM01] proposed at SIGGRAPH[3] an engine for generating a city based on production rules similarly adapting L-System concepts to 3D modeling, vaguely resembling work done by Shlyakhter et al adapted to more global urban modeling as can be observed in Figure 2.6. Authors claim that from various image maps given as input, such as land-water boundaries and population density, the system could generate a system of highways and streets, dividing the land into lots, and creating the appropriate geometry for the buildings on the respective allotments. Additionaly, for the creation of city street maps, L-Systems were extended with methods that would allow the consideration of global goals and local constraints and reduce the complexity of the production rules.

Continuing the work presented by Parish, a few years later Wonka et al [WWSR03] presented at SIGGRAPH 2003 a system capable of generating instant realistic architecture by exploiting vertical and horizontal coherence of buildings facades as can be seen in Figure 2.7. The authors developed a new method for the automatic modeling of architecture. "Building designs are derived using split grammars, a new type of

[3]http://www.siggraph.org - one of the most reknown computer graphics conferences

Figure 2.7: Screenshot of facade generation [WWSR03].

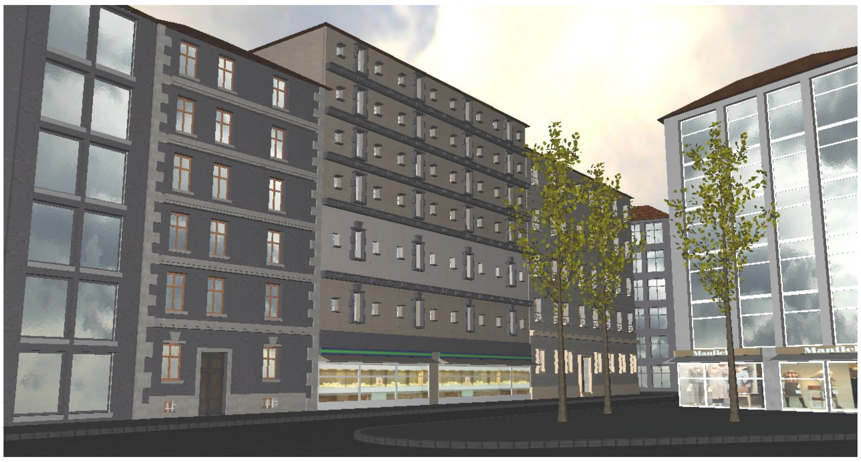
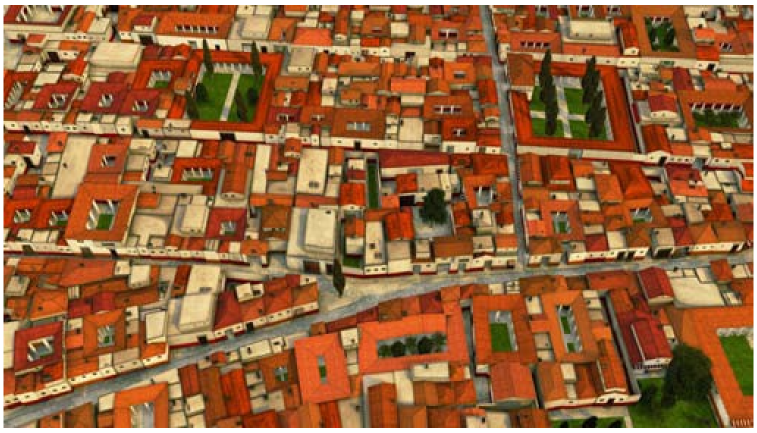


Figure 2.8: Screenshot of detailed generated city [MWH+06].



parametric set grammar based on the concept of shape. The paper also introduces an attribute matching system and a separate control grammar, which offer the flexibility required to model buildings using a large variety of different styles and design ideas. Through the adaptive nature of the design grammar used, the created building designs can either be generic or adhere closely to a specified goal, depending on the amount of data available."

A few years later Muller et al [MWH+06] present further developments of the system, now applied to the instant modeling of more complex buildings and structures as can be seen in Figure 2.8. The authors claim to have built on the idea of the split rule originally presented by Wonka [WWSR03] as an "important ingredient for CGA shape. As split grammars maintain a strict hierarchy, modeling is fairly simple, but also limited. However, after introducing rules for combinations of shapes and more general volumetric shapes such as roofs, the strict hierarchy of the split-grammar can no longer be enforced. We can confirm, that the idea of split rules is a very suitable primitive to generate facade details, but we did not find it suitable for many forms of mass modeling. We made use of the control grammar to generate procedural variations together with simple stochastic rules. We believe that our model of context sensitive shape rules, together with the interplay of one, two, and three dimensional modeling is an elegant and efficient solution to a challenging problem. Besides this major conceptual contribution, we are also the first to address application related details, such as the definition of the most important shape rules, the concise notation, and modeling examples detailing various modeling strategies."

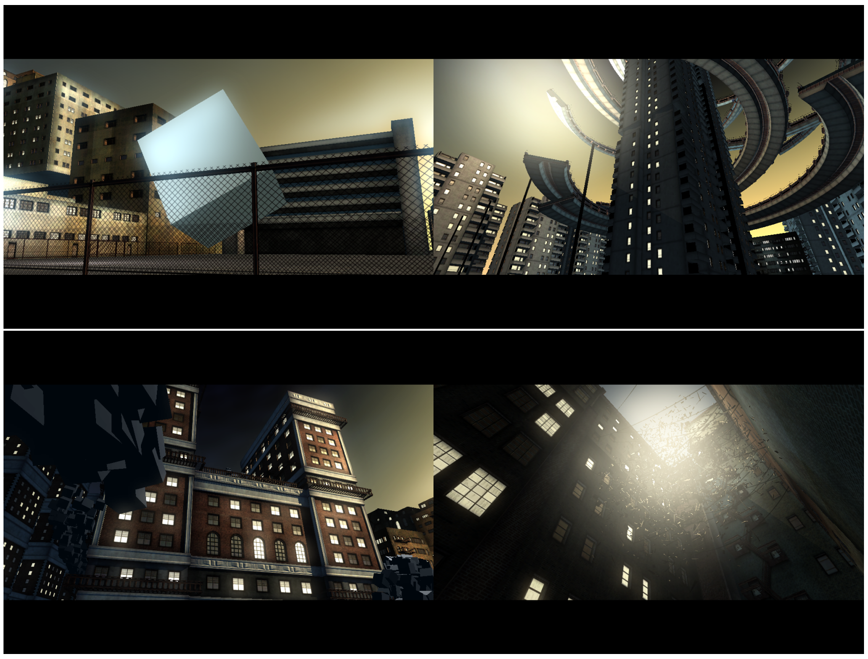
A recent paper (May/June 2008) by Finkenzeller [Fin08] presents different approaches to "Detailed Building Facades" also following the work of Muller. In the same publication an article titled "Procedural Urban Modeling in Practice" written by Watson et al [WMW+08] presents a good state of the art on procedural urban modeling, refering to the previous work of Parish, Wonka and Muller and additionally refering to an application of the engine in recent video game titles published by Electronic Arts such as "Need For Speed", seen in Figure 2.9.

Figure 2.9: Screenshot of the video game "Need For Speed".

A related SIGGRAPH 2008 contribution which is also worth mentioning is the article by Lipp et al "Interactive Visual Editing of Grammars for Procedural Architecture" [LWW08].

It's important to stress that other expeditious modeling procedural systems exist presenting similar work, Schilling [SC03] for example, developed a system which demon-

Figure 2.10: Scenes from the 177kb demo "fr-041: debris." by Farbrausch.



strated how integrating existing 2D and 3D data automatically through procedural rules can easily generate credible representations of a city. In this case, the purpose and demonstration of the system was the generation of a virtual 3D-tour through the city of Heidelberg.

Another good example of use of procedural modeling for the generation of expeditious urban environments is a recent production released in the demoscene [TSS04], it's called "fr-041: debris." and was presented in 2007 by the group Farbrausch[4]. They present a 7 minute realtime presentation generated from a 177kb executable without use of external data. All sound, textures and 3d models as seen in Figure 2.10 are procedurally generated.

## 2.3 The XL3D Modelling System

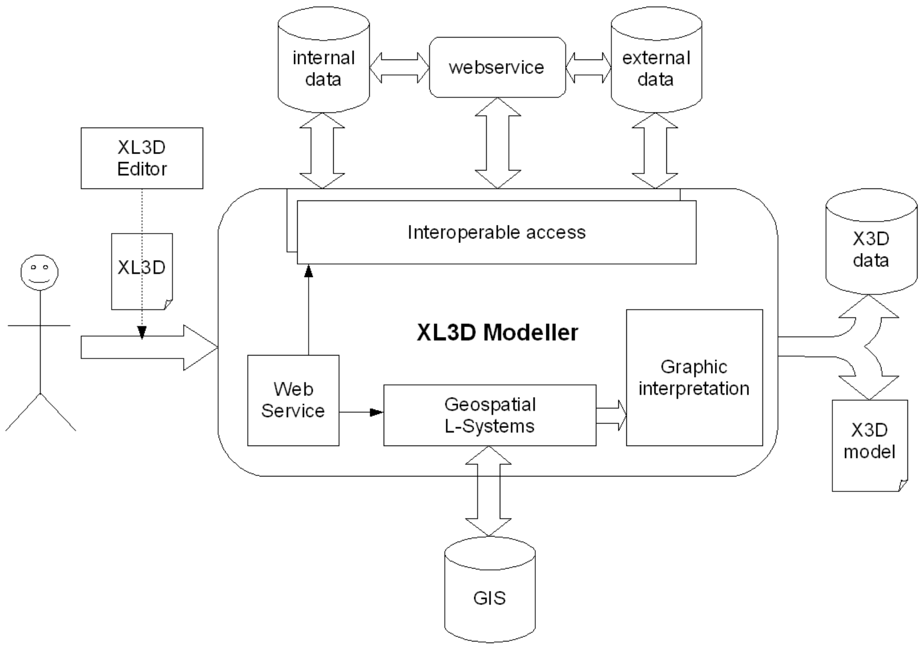
Another contribution to procedural modeling is named XL3D (Extensible L-System based 3D modeling system) proposed by António Coelho [CBSF07]. This technology will be used in the methodology proposed in further chapters of this document and as

[4]http://www.farbrausch.de/ - group of friends active in the demoscene for the last 10 to 15 years, most of which work in the game development industry

such lends itself to a more detailed description in this section.

The XL3D modeling system operates automatically from a modeling specification. This specification defines the location of the relevant data, the structure of the required 3D models and a set of geospatial production rules representing the required knowledge to model the structure as an L-System. The process is specified declaratively through an XML document based on an XML schema specially developed for this modeling system and homonymously refered to as XL3D. The architecture of the system can be seen in Figure 2.11.

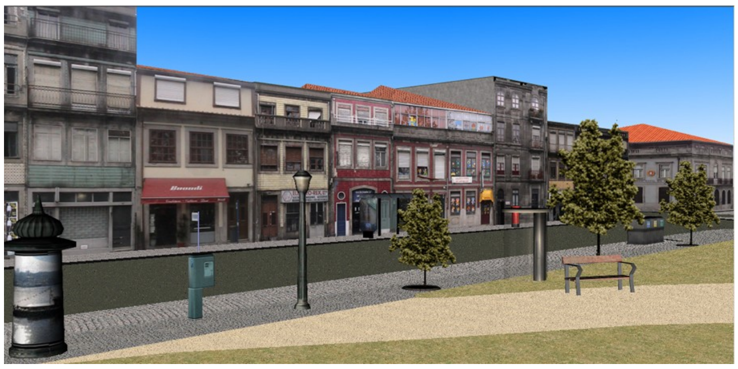
Figure 2.11: XL3D architecture.



Each XL3D document is composed by a modeling project that defines the wanted models, the data sources, the modeling processes and the prototypes used for their graphical interpretation.

Each model is structured hierarchically as an modeling entity tree. Each modeling entity reflects a user defined concept pertaining to the problem. Each entity is modeled through modeling procedures which after beeing parameterized with the data source references and the interpretation prototypes, automatically generate the corresponding three dimensional models as the one seen in Figure 2.12. The modeling procedures encapsule a set of production rules which define the geospatial L-Systems. The prototypes interpret the resulting modeling process string, generating the X3D models. The data sources refer the data location and how the relevant information can be converted into geospatial strings which can be used to form axioms.

Figure 2.12: Screenshot of Praça da Républica [CBSF07].



## 2.4 Summary

This chapter offers the reader a brief insight on the state of the art of 3D modeling in general and expeditious modeling for virtual urban environments in particular. Listing some of the most important tools and modern methodologies to generate realistic virtual models of urban environments from the real world.

This chapter relates to the thesis in the sense that it offers perspective insight to the field of 3D modeling in general and lists some of the common problems of expeditious urban modeling systems. These common problems will be addressed later in this thesis and solved by applying techniques derived from the field of artificial intelligence.

# Chapter 3

# Combinatorial Optimization

Artificial intelligence is a term originally coined to represent the science and engineering of intelligent machines, as for example defined by McCarthy et al [MMRS55]. in modern days the term artificial intelligence is more commonly applied in reference to intelligent autonomous agents - defined as computational entities with some degree of awareness and interaction of its environment, including the capacity to communicate in multi agent systems [Wei99]. But the original definition of artificial intelligence also includes the research and development of computational systems capable of demonstrating behavior that can be perceived as inteligent by humans. This behavior includes traits of reasoning, knowledge, planning, learning, communication, perception and object interaction [RN03].

As such, artificial intelligence bridges the areas of computer science, psychology, philosophy, neuroscience, cognitive science, linguistics, operations research, economics, control theory, probability, optimization and logic. Application fields include robotics, intelligent agents, multi-agent interaction, swarm behavior, knowledge discovery and data mining, speech and facial recognition, control systems, search algorithms, optimization, scheduling, etc.

One of the fields of artificial intelligence relevant to solving the problem addressed by this thesis is the field of knowledge discovery and data mining. Knowledge discovery and data mining is a relatively recent area of research [CHS$^+$97], it consists in extracting useful knowledge from large amounts of data, solving problems of classification, clustering and predicting future behavior. Despite the application of some knowledge discovery and data mining principles to this thesis, they do not play a significant primary role that might warrant deeper analysis.

A field of artificial intelligence with key importance to this thesis however, is the field of optimal search algorithms and parameter optimization. The field of optimal search is based in researching methodologies that can solve combinatorial problems more efficiently.

## 3.1 Parameter Optimization

An optimization problem can be defined in general terms as determining the optimum combination of parameter values of a given vector $\vec{v} = (v_1, ..., v_n) \in M$ that best satisfies a given objective function $f : M \rightarrow \Re$ by minimization (or maximization) $f(\vec{x}) \rightarrow max$.

A similar but slightly more formal description, taken from wikipedia[1]:

> "In mathematics and computer science, an optimization problem is the problem of finding the best solution from all feasible solutions. More formally, an optimization problem $A$ is a quadruple $(I, f, m, g)$, where
>
> - $I$ is a set of instances;
> - given an instance $x \in I$, $f(x)$ is the set of feasible solutions;
> - given an instance $x$ and a feasible solution $y$ of $x$, $m(x, y)$ denotes the measure of $y$, which is usually a positive real.
> - $g$ is the goal function, and is either min or max.
>
> The goal is then to find for some instance $x$ an optimal solution, that is, a feasible solution $y$ with $m(x, y) = g\left\{m(x, y')|y' \in f(x)\right\}$"

The set of feasible solutions is commonly refered to as the *search space*. Each combination of values a *solution* which has an associated so called *fitness* or *quality* value. Obtaining the *quality* value for each *solution* typically involves a computer simulation processes, where a model must be first generated following the given vector parameter combination and only then the objective function can evaluate the *quality* of the solution.

The system can also have a stochastic nature, meaning that it can return different *quality* values for the same *solution* input parameters. The impact of this factor can often be reduced by applying statistic methods and error compensation techniques.

[1]http://en.wikipedia.org/wiki/Optimization_problem - 2nd July 2008

These special cases will not be approach any deeper in this document, since they go beyond the scope of this thesis.

Combinatorial problems can be under the influence of several parameter and multi objective constraints. A more formal mathematical definition of complex combinatorial optimization problems can be consulted under the work of Bäck [Bac96].

The input parameters can be of two fundamental types: discrete and continuous. Some of the different approaches and techniques applied to solve combinatorial problems impose a discretization of any continuous nature of parameters prior to their application. However, discretizing continuous values to reduce the *search space* and allow the application of discrete methodologies can impose unwanted constraints. Optimization algorithms that handle continuous (also called *real based*) parameters have thrived in the literature as well documented by Pardalos [PR02] [DP05].

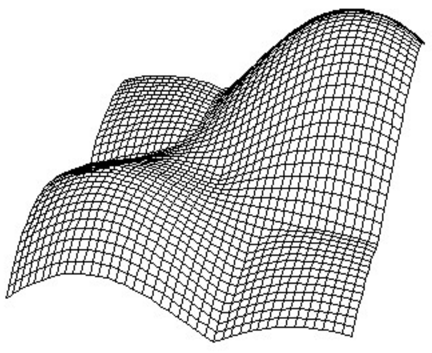
Even when only dealing with discrete values of parameters, it is often impractical to simulate every possible combination of $\vec{v}$ to ascertain the optimum solution. The quantity of different parameters and corresponding search range of each parameter easily increases the complexity and *search space* of the problem exponentionally. Each simulation process usually takes a lengthy time to conclude, and as so, it is important to avoid extensive search. Preferably we would like to unravel the optimum solution by searching only a fraction of the entire *search space*.

In continuous problems, the quality landscape of the search space is typically shaped in the form of one or several hills depending on how the parameters are related, as can be seen in Figure 3.1. Considering certain small constrained sections of the *search space* the optimum value may change, the best value in these local sections is refered to as a *local optimum*, whereas the best global value of the entire *search space* is refered to as *global optimum*. One of the strongest concearns for performance of certain optimizaion algorithm, depending on the complexity of the problem, is that they should not get caught in a *local optimum* believing to have found the *global optimum*.

Considering the problem as a black box, where there is no knowledge on the correlation between the input parameters and the *quality* value of the *solution*, the only way to determine the optimal combination of the problem is to test several *solutions* from the *search space*. The *solutions* are selected by the algorithm considering the *solution quality* history and the known complexity of the problem in terms of convergent *search space*.

Several algorithms have been widely documented in the literature as capable of re-

Figure 3.1: Solution quality landscape for a 2D optimization problem.



ducing the required number of iterations and increasing the probability that a *global optimum*, or very near optimum, *solution*, is reached while testing only a limited fraction of the *search space*.

The performance of the individual algorithms varies depending on the characteristics of the problem. The selected algorithm must have a good balance of interests: On one hand ensuring a minimal iterations will be required to find the optimum; on the other, keeping to a low probability the danger of converging to a *local optimum* instead of the *global optimum*.

## 3.2 Local Search

One of the simpler families of algorithms developed to solve combinatorial problems is referred to as local search (LS). Some authors in the literature also commonly refer to these algorithms as random search algorithms (due to their stochastic nature of educated guessing) or more simply meta heuristic algorithms [GK03].

LS methodologies consist of combinatorial problem optimization algorithms that seek to find an optimum solution by recursively analysing neighboring solutions within the search space. Traditionaly a solution is considered neighbor to another when sharing some or most of their features that define the *search space*, in some discrete problems a neighbour solution is considered every feasible solution that can be reached from the current solution by a move or swap operand. In continuous *search space* the neighbor solution is defined by sharing a fixed number of parameters among both solutions or being within a certain parameter vicinity scope. The general protocode for a typical minimization problem using LS is presented in Algorithm 1.

```
bestSolution = SelectRandomSolution(searchSpace);
while (!Stop_Criteria) do
    thisSolution = SelectNeighbourSolution(thisSolution, searchSpace);
    if (thisSolution.quality $\leq$ bestSolution.quality) bestSolution = thisSolution;
end
```

**Algorithm 1**: Local search base code

The stop criteria for LS algorithms is usually adapted to the problem and taking advantage of the properties of the chosen heuristic methodology. The most common stop criteria for an LS algorithm usually are:

- surpassing a certain number of total iteration steps;
- repeating a certain number of iterations without significant improvement of quality;
- reaching a certain quality threshold.

### 3.2.1 Random Search

There are several definitions of random search (RS) found in the literature. Some authors generally refer to RS as the LS family of algorithms. The definition we wish to present in this sub-section actually pertains to the pure stochastic method for analysing combinatorial problems search spaces, and can in itself be considered a variant of an LS algorithm.

RS follows a stochastic logic. It systematically tests the quality of new randomly selected candidate solutions from the search space until a certain criteria is met, comparing each tested solution with the best solution found so far and replacing it when having found a better quality one. The base protocode can be seen in Algorithm 2, again for a minimization problem.

```
bestSolution = SelectRandomSolution(searchSpace);
while (!Stop_Criteria) do
    thisSolution = SelectRandomSolution(searchSpace);
    if (thisSolution.quality $\leq$ bestSolution.quality) bestSolution = thisSolution;
end
```

**Algorithm 2**: Random search base code

This stochastic methodology based on random chance can deliver acceptable results in some types of combinatorial problems but the heavy number of iterations that it might require to ensure having found the optimum solution is unacceptable in other. As such, RS is only occasionally used to offer a vague overview of the search space or to find a decent quality initial candidate solution as a starting point for other algorithms to base their search upon.

### 3.2.2 Hill Climbing

One of the simplest LS algorithms is referred to as the hill climbing (HC) algorithm. The HC algorithm pursues the neighbors of the best possible solution found so far. Literature regarding HC can be consulted on the works of Rich [RK90] and more recently Russel [RN03]. HC type of algorithms can be very effective in problems with linearly dependent constraints. Their performance depends greatly on the definition of neighboring solution. A big problem with the use of HC algorithms is their inability to guarantee reaching the optimal value in problems of many local optimums.

Another important limitation of the HC algorithm includes the possibility of reaching fitness quality plateaus, where all neighboring solutions share a very close quality value, often causing the algorithm to wander through the local search space not being capable finding, or taking too many iteration steps discovering, the best direction converging towards the optimum.

The most basic form of HC will simply test all, or just a few randomly chosen, neighboring candidate solutions from the current best solution and replace it by the best of the neighboring solutions tested. Repeating the process for the next local search space of neighboring solutions until a certain stop criteria is met, typically that no better quality solutions could be found in the local search space of the neighborhood. The base protocode can be seen in Algorithm 3. The most common stop criteria for hill climbing is having no better quality neighbour.

```
bestSolution = SelectRandomSolution(searchSpace);
while (!Stop_Criteria) do
    thisSolution = SelectBestNeighbourSolution(bestSolution, searchSpace);
    if (thisSolution.quality $\leq$ bestSolution.quality) bestSolution = thisSolution;
end
```

**Algorithm 3**: Hill climbing base code

There are a few variations of the basic HC algorithm principle, one of them is referred to in the literature as the steepest ascent hill climbing (SAHC) algorithm. SAHC acts similarly to the previously described standard HC algorithm, its difference is that it probes all possible neighboring solutions of the entire search space path found so far before choosing the best one and repeating the process iteratively. This allows a certain form of smarter back-tracking to the search process of the original HC.

Another variant, that seeks to minimize the risk of having found a local optimum instead of a global one, is the random start hill climbing (RSHC). RSHC will restart itself with a new initial candidate solution a few times whenever it reaches a quality improvement dead end. This will improve the probability of having found a search path that will lead to the global optimum.

Yet another variant, the stochastic hill climbing (SHC) algorithm, choses the next best solution randomly from a list of all the higher quality neighbors available. In some of its implementation the probability of choice may depend on the steepness of the uphill move.

Other variants of HC have also been documented in the literature with successful results. As most optimal search algorithms the quality of their performance depends largely on the problem they are required to solve. A recent comparative study can be found in the work of Jacobsen and Yucesan [JY04].

### 3.2.3 Tabu Search

A variant of LS algorithms is referred to as tabu search (TS). This methodology, credited to Glover and Laguna [GL02], differs from the remaining algorithms by maintaining in memory a list of recently tested candidate solutions with the purpose of avoiding unnecessary retesting. The tutorials of Hertz [HTdW95] and Gendreau [Gen03] provide valuable conceptual information on the basic, intermediate and more advanced forms of TS.

Authors claim that that basic TS can be seen as simply the combination of LS with short-term memories. “Usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the specific information that is recorded. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations; others are based on key characteristics of the solutions themselves or of the moves."

The most basic implementation of the TS algorithm for a minimization problem can be consulted in Algorithm 4. More complex variants of TS typically have more than one tabu list assigned to different problem knowledge specifications for example. Also a parameter defined as *aspiration criteria* is designed to either prevent a certain solution from either entering a tabu list or removing it from the list when certain parameters have been met. This logic allows the algorithm to move into neighbourhood solutions which have some of their parameters defined as tabu but might hold an improvement in quality. The stop criteria is similar to other LS heuristics (maximum number of iterations, maximum number of recent iterations without improvement, reaching a threshold) but in complex tabu schemes the search is usually stopped after completing a sequence of phases, the duration of each phase being determined by one of the traditional stop criterias mentioned.

```
tabuList = emptyList;
bestSolution = SelectRandomSolution(searchSpace);
while (!Stop_Criteria) do
    thisSolution = SelectNeighbourSolution(bestSolution, tabuList, searchSpace);
    if (thisSolution.quality $\leq$ bestSolution.quality) bestSolution = thisSolution;
    UpdateTabuList(thisSolution, tabuList);
end
```

**Algorithm 4**: Tabu search base code

Also present in more advanced implementations of TS are the concepts of *intensification* and *diversification*. *Intensification* serves to ocasionally shift the algorithm focus into a specific neighbourhood family that might hold improved quality, while *diversification* attempts to ensure the entire *search space* is beeing considered.

> "The idea behind the concept of search intensification is that, as an intelligent human being would probably do, one should explore more thoroughly the portions of the search space that seem "promising" in order to make sure that the best solutions in these areas are indeed found. From time to time, one would thus stop the normal searching process to perform an intensification phase. In general, intensification is based on some intermediate-term memory, such as a recency memory, in which one records the number of consecutive iterations that various "solution components"

> have been present in the current solution without interruption. (...) One of the main problems of all methods based on Local Search approaches, and this includes TS in spite of the beneficial impact of tabus, is that they tend to be too "local" (as their name implies), i.e., they tend to spend most, if not all, of their time in a restricted portion of the search space. The negative consequence of this fact is that, although good solutions may be obtained, one may fail to explore the most interesting parts of the search space and thus end up with solutions that are still pretty far from the optimal ones. Diversification is an algorithmic mechanism that tries to alleviate this problem by forcing the search into previously unexplored areas of the search space. It is usually based on some form of long-term memory of the search, such as a frequency memory, in which one records the total number of iterations (since the beginning of the search) that various "solution components" have been present in the current solution or have been involved in the selected moves."

Additionally it is worth noting that many positive results have been documented in the literature regarding hybrid implementations of TS with other meta heuristic algorithms to increase their standard performance. Two examples are the work of Youseff [YSA01] and Mishra [MPT$^+$05].

### 3.2.4 Simulated Annealing

Simulated annealing (SA) is based in the natural process of annealing of metallurgic materials. A slower cooling process would typically originate stronger ligaments than by rapid cooling process. This is explained at an atomic level by the fact that slower cooling allows larger elements to arrange themselves in a more efficient configuration before the remaining elements connect around them. Generating a tighter configuration with stronger connections.

The algorithm was first presented by Kirkpatrick [KGV83] originating from an adaptation of the Monte Carlo methodology applied to thermodynamic systems that would become referred as the Metropolis-Hastings algorithm [MU49]. The term monte carlo dates back to physicists working on nuclear weapon projects in the Los Alamos National Laboratory in the 1940s, refers to the use of random values and applied probabilistic comparisons to assist solving non deterministic problems.

Kirkpatrick, Gelatt and Vecchi [KGV83] defined simulated annealing as a meta heuristic algorithm that would search for new neighboring solutions of the combinatorial problem and accept them based on a probability ratio. This probability ratio varies depending on the quality difference of the solutions and a global decreasing temperature value defined by a special formula defined in Equation 3.1.

$$p(\Delta f, T) = \begin{cases} e^{\frac{-\Delta f}{T}} & \Delta f \leq 0 \\ 1 & \Delta f > 0 \end{cases} \tag{3.1}$$

The stopping criteria for SA is usually passing a certain temperature threshold, given that the temperature value $T$ decreases on each passing iteration by a certain rate which can be variable to better fit the specific problem. As shown in the protocode of Algorithm 5, the probability of accepting a solution of lower quality decreases with the temperature. This process gives room for solutions to wander through a wider range of the search space during initial iterations and slowly reduce that wandering capacity, turning the algorithm into a standard hill climbing in the final iterations. This behaviour helps avoiding a typical flaw where the optimization process gets stuck in a *local optimum* neighborhood.

```
currentSolution = bestSolution = SelectRandomSolution(searchSpace);
while (!Stop_Criteria) do
    thisSolution = SelectNeighbourSolution(currentSolution, searchSpace);
    if (thisSolution.quality $\leq$ bestSolution.quality) bestSolution = thisSolution;
    $\Delta f$ = bestSolution.quality - thisSolution.quality;
    if ( $p(\Delta f, T) >$ random() ) currentSolution = thisSolution;
    UpdateTemperature($T$);
end
```

**Algorithm 5**: Simulated annealing base code

The downside of SA is the inability to guarantee that a global optimum, instead of a local optimum, can always be reached or how many iteration steps it can take to guarantee it. These questions highly depend on the complexity of the problem and the linear dependency between the parameters and the quality function. The temperature descent ratio is also required to be fine tuned paying special attention to the characteristics of the problem. Uncareful parameter tuning can easily affect the performance of the algorithm, reducing it to the same of a basic hill climbing implementation.

Some authors have achieved higher performance by combining implementations of SA with the previously described tabu search (TS) algorithm. The work of Mishra [MPT+05] for instance is a good example of such a successful application.

## 3.3 Population Based

A cluster of optimization algorithm is refered to as population based (PB), they differ from LS algorithms in the sense that each passing iteration shall test a family of solutions instead of one. This enables a different mindset where parallel evolution plays a more important role over the *solution* history or the stochastic balance of the algorithms previously analysed. The biggest downside of PB algorithms in comparison to LS is having to additionaly balance the number of *solutions* per iteration to ensure a reliable convergence minimizing the required iteration steps and probability of reaching a *local optimum* instead of the *global optimum*.

Several algorithms documented in the literature can be classified as PB such as the more recent work of Yang [Yan05] and Grosso et al [GLS07]. This document focuses on two PB families specifically, genetic algorithms and harmony search: Genetic algorithms being inspired by nature evolution, is one of the most popular families of PB, harmony search being inspired by musical search for perfect combined melody.

### 3.3.1 Genetic Algorithms

Genetic algorithms is the name given to the nature inspired system based on the Darwinian concept of the survival of the fittest, where each solution to our combinatorial problem is uniquely encoded and part of a group or family of solutions which evolves by each passing generation. The individual solutions are crossbred and mutated to give birth to new generations of solution families sharing traits with the original parent family of solutions. Original reference to this algorithm dates back to the work of Holland in 1975 [Hol92]. A more modern reference can be found in the work of Goldberg [Gol02].

Several implementations of the algorithm exist. Most implementation found in the literature follow the original concept of binary encoding and elitist selection of each new generation. The standard implementation refers to the problem's parameters being encoded into a binary dna string. Each possible parameter combination - our solution - is defined by a unique dna string which can be crossbred with other dnas or mutated alone to generate new dna solutions, yielding composite traits from the parents.

There are quite many variants discussed in the literature referring to the many possible ways to crossbreed the binary encoded solutions and the influence of the mutation factor in different types of problems. A comparatively recent and quite interesting new approach to the original genetic algorithms concept is the so called real-value (also referred to as real coded or real based) variations of the algorithm handling continuous parameters.

Real based genetic algorithms have been taking a strong interest in the past decade, the hand book of genetic algorithms by Davis [Dav01] contains a good general reference and some new developments to the work on this area made by Michalewicz [Mic94]. Additionally, the work of Arumugama [ARP05] is worth consulting for some information regarding new real based cross breeding techniques and their comparative work. Also worth mentioning are new real based mutation factors presented and analysed comparatively under the work of Deep and Thakur [DT07b] and the recent simulated annealing inspired hybrids documented the works of Wang et al [WWR05] and He and Hwang [HH06].

For the purpose of this thesis, there is a higher interest in the real based implementations of genetic algorithms. In real based genetic algorithms the fundamental steps are still present alike their original binary implementation, despite a few conceptual modifications to handle real-based parameters, these fundamental steps are:

- Parent Selection;
- Cross Breeding;
- Mutation;
- Validation.

**Parent Selection** Parent selection typically involves choosing which solutions of the current generation are fit enough to breed into the next generation. Parent selection is an important phase of the algorithm since it controls the elitism factor of the process, elitism is usually the main force what drives the generations of algorithm into a neighborhood of optimum solutions.

Allowing too many solutions to get selected for the next step of the algorithm could generate a logistic overflow, making the algorithm test many unnecessary new solutions with a low probability of offering any improved quality. A careful balance is required to avoid wasting resources but still ensure we are selecting enough parents to avoid a premature convergence to local optimum.

The most basic selection criteria is the random selection of a fixed number of solutions from the population. Another common variant is the elitist selection, where simply the first n top quality solutions from the population are selected. A more modern approach is the so called roulette wheel selection, where each solution is assigned a space on the roulette wheel proportional to its quality and the solutions with the largest portion on the wheel have the greatest probability to be selected from the population. Another popular selection method is referred to as tournament selection, where a certain number of solutions is chosen randomly from the population and the best solution from that group is selected as a parent, the process repeating itself until the complete new population is filled.

Many selection criteria are documented in the literature. It's worth noting that the use of hybrid versions where different criteria are applied to select a given percentage of the population is also not at all uncommon.

**Cross Breeding** Cross breeding (also referred by some authors as cross over) is the process of creating a new solution based on the traits of the selected population, it can surge from two or more parents. There are many different techniques for crossbreeding solutions, the most basic types are single point crossbreeding and multi point crossbreeding. These are applicable to binary encoding of solutions only. Real based genetic algorithms apply other crossbreeding methods.

Single point crossbreeding refers to a single point of a dna string that will be randomly selected as the trimming point. One half of the new dna string will belong to one of the parents, the other half to the other parent. Multi point crossbreeding works similarly to single but selecting several points which define several strings of dna that get exchanged. Multi parent crossbreeding has also been known to offer good results. Basic implementation involves selecting a short random number of parents and multi point crossbreed several dna sections between them.

Real based genetic algorithms require different crossbreeding methods from binary genetic algorithms. Arithmetic cross-over for example follows a linear recombination of two vector solutions affected by a certain random factor as can be seen in Equation 3.2.

$$x' = \lambda x + (1 - \lambda)y \tag{3.2}$$

$$y' = (1 - \lambda)x + \lambda y \tag{3.3}$$

An extremely popular real based cross breeding method is referred to as average convex cross-over. It's basically a pondered average between the parameters of two parent solutions. The weighted average of two vectors x1 and x2 are calculated as $\lambda_1 x_1 + \lambda_2 x_2$ where the multipliers are restricted to $\lambda_1 + \lambda_2 = 1; \lambda_1 > 0; \lambda_2 > 0$ Depending on the restrictions applied to the multipliers on the arithmetic equations, these methods are referred to as three kinds: convex cross-over, affine cross-over, and linear cross-over. The special case of $\lambda_1 = \lambda_2 = 0.5$ is the one referred to as average convex cross-over or intermediate cross-over by other authors.

Another cross breeding method worth mentioning is the direction based cross over. This operand, also known as heuristic cross over, has awareness of the quality of the solutions and creates the offspring under a weighted average tipping to the top quality parent. The method respected the following arithmetic rule $x' = r(x_2 - x_1) + x_2$ between the higher quality parent $x_1$ and the lesser quality parent $x_2$, where r is a random number between 0 and 1.

Many other more recently proposed cross over operands exist, another worth mentioning is the work of Deep and Thakur [DT07a] with the so called Laplace crossover. This method consists of generating a number $\beta$ following the Laplace distribution as described in Equation 3.4 and applying it arithmetically with both parent solutions, generating two offspring described in Equation 3.5.

$$\beta = \begin{cases} a - b\log_e(u) & u' \leq \frac{1}{2} \\ a + b\log_e(u) & u' > \frac{1}{2} \end{cases} \tag{3.4}$$

$$x' = x + \beta|x - y| \tag{3.5}$$

$$y' = y + \beta|x - y| \tag{3.6}$$

Variants and hybrids of these cross-over methods arent at all uncommon in many scenario applications. One common modification with promising results is achieved by modifying the multiplier values of some of these cross-over methods into a dynamic form. Another common implementation is a hybrid version or variation where the different methods are applied in different percentages to the population. Also recently popular are multi-parent cross over methods where the offspring has its solution vector influenced by more then two parents at a time.

As in most optimizer algorithms, these variations can offer good results in certain applications, but their overall performance is largely influenced by the complexity and co-relation of the parameters from the problem that is being solved.

**Mutation** The mutation step in the algorithm sequence is considered as essential step in forcing the search process to search outside the limits imposed by the initial population. Mutation operands of the standard genetic algorithm implementation consist of modifying some of the binary bits of the dna string solution in a more or less controlled yet random manner.

Due to their nature, the real based genetic algorithms mutation operands are significantly different from their binary dna string counter parts. Comparatively speaking, most cross over operands for real based genetic algorithms take away some of the importance that the mutation operand had in the binary genetic algorithms. Some authors disagree and did focus research in new mutation operands capable of improving performance under an adequate selection of the remaining parameters facing the problems complexity.

Michalewicz [Mic94] proposed several mutation operands, one of which is most worth mentioning due to its popularity is the so called non-uniform mutation. It functions as described in Equation 3.7 generating point $x^{t+1}$ from $x^t$ on each passing iteration. $t$ is current generation number, $r$ is a uniformly distributed random number between 0 and 1. $x_i^1$ and $x_i^u$ are lower and upper bounds of the ith component of the decision vector, $u$ is a uniformly distributed random number in the interval $[0, 1]$, $T$ is the maximum number of generations and $b$ is a parameter determining the strength of the mutation operator. Due to its mathematical constraints the algorithm has a broader search space in the first iterations and a more locally concentrated around the parents search space in the final iterations of the process.

$$x_i^{t+1} = \begin{cases} x_i^t + (x_i^u - x_i^t)(1 - u^{(1-\frac{t}{T})^b}) & \text{if } r \leq 0.5, \\ x_i^t - (x_i^t - x_i^1)(1 - u^{(1-\frac{t}{T})^b}) & \text{if } r > 0.5, \end{cases} \tag{3.7}$$

A more recent mutation operand also worth referring is the power mutation proposed by Deep and Thakur [DT07b]. This operand is based in power distribution. Its distribution function is given by $f(x) = px^{p-1}, 0 \leq x \leq 1$ and its density function is given by $F(x) = x^p, 0 \leq x \leq 1$ where $p$ is the index of the distribution. To generate the mutated point $y$ from the original parent $x$, a random number $s$ is created following this distribution and the Equation 3.8 where $t = \frac{x-x^1}{x^u-x^1}$ and $x^l$ and $x^u$ are lower and upper bounds of the decision variable and $r$ is a uniformly distributed random number between 0 and 1. The strength of mutation is governed by the index of the mutation $p$ which can be configured to vary between 0 and 1. For small values of p less perturbance in the solution is expected and for large values of $p$ more diversity is achieved.

$$y = \begin{cases} x - s(x - x^1) & \text{if } t \leq r, \\ x + s(x^u - x) & \text{if } t > r, \end{cases} \tag{3.8}$$

Many other real based mutation operands have been presented in the literature, however, the interest of this thesis lies solely on the more popular and recent, not a complete overview on the state of the art.

**Validation** Validation is the final step of genetic algorithms. It verifies if the mutated offspring is within valid parameters of the problem. In binary implementations of genetic algorithms it had a more crucial role to the entire process, since most of the cross over and mutation algorithms functioned unaware of the encoding applied to the dna string which getting scrambled would often result in non-sensical representations. For real based genetic algorithms the validation step is not so comparatively important but is still applicable. All values of the vector must fall within the assigned range. Certain problems also have specific constraints amongst the vector parameters which can be checked in the validation step.

### 3.3.2 Harmony Search

Harmony search (HS) is a fairly recent meta heuristic, population based, algorithm. It takes its principle from musician improvisation sessions where musicians try different note combinations to find the best melody. Its theory and industry applications are fairly well documented in the literature by Geem [GKL01], Lee [LG05] and Mahdavi [MFD07].

HS is applicable to problems handling discrete and continuous variables alike, fitting a middle range of applicability where most meta heuristic algorithms can only handle discrete values and gradient-based mathematical algorithms can only be applied to continuous variables.

The algorithm works by initializing a random population of possible vector *melody* solutions, each solution is a combination of *notes* under a certain possible range. The algorithm uses a probability threshold to decide if part of the new solution vector is copied from previous memory of good performances or instead randomly improvised. Additionally, each *note* can have its *pitch* slightly randomly adjusted, within a certain *pitch bandwidth*, depending on another probability threshold. Improved solutions are kept in memory and the process is repeated a certain number of iterations. The base protocode can be seen in Algorithm 6.

```
population = GeneratePopulation(searchSpace);
while (!Stop_Criteria) do
    thisSolution = NewRandomSolution(searchSpace);
    foreach (note in thisSolution) {
        if (copyNoteProbability > rand()) note = CopyNote(population[rand()]);
        if (adjustPitchProbability > rand()) note ±= bw * rand();
    }
    UpdatePopulation(thisSolution,population);
end
```

**Algorithm 6**: Harmony search base code

Some implementations of the HS algorithm vary the threshold probabilities or the *pitch bandwidth* during the iterative process to condition the algorithm into a simulated annealing alike behavior. Depending on the characteristics of problem, this variation can be very effective in providing improved results over the standard HS implementation.

## 3.4 Other Algorithms

Many other meta heuristic and non meta heuristic algorithms have been documented in solving combinatorial problems of optimal search. There is no best algorithm to solve all types of combinatorial problems. Their performance varies depending on the type and complexity of the problem and the constraints of the parameters. Often certain properties and characteristics of a particular algorithm can be applied to another algorithm to generate hybrid form with improved performance for the problem at hand alone. Several schools or families of algorithms have been documented regarding these issues. Considering this, even if this thesis will not approach them directly, it is important to note the characteristics and describe the common traits of these other optimization algorithms commonly applied in solving combinatorial problems.

**Ant Colony Optimization** Proposed by Dorigo [Dor92], ant colony optimization is based in the natural process of ant hill dynamics searching for food. In the real world, the ants typically wander randomly by trails leaving a pheromone trail, this trail tends to evaporate over time but is reinforced when the ant returns with food sooner than to be expected, other ants when encountering heavier concentration of pheromones will follow the trail to collect the food reinforcing the trail some more. This behavior is mapped to a virtual scenario where simulated ants walk combinatorial problem graphs attempting to find the shortest connection.

There are several variants of the algorithm, the most known are the elitist ant system, the max-min ant system and the rank-based ant system. What varies between them is mostly the method in which the pheromones are dropped along the trails in order to control the probability ratio of other ants following always the same trail or scatter more easily. Ant colony optimization algorithms are particularly interesting for dynamic scenarios where the connections between the optimum solution can easily break and the algorithm is forced to find an alternative path without restarting the entire analysis of the problem. Successful applications have been documented in network routing and urban transportation systems.

**Constraint Programming** Constraint Programming (CP) [Dec03] [Apt03] is a programming paradigm where relations between variables are stated in the form of constraints on variables over a given domain. A solution for a Constraint Satisfaction Problem using CP is an assignment of a value to each of the variables that satisfies all the problem constraints. CP is typically used for solving decision problems but can also be extended to solve optimization problems by defining an evaluation function that maps the problem variables into a real value. The optimization process consists in a process that, after achieving a given valid solution, posts a new constraint stating that a new solution must have a higher/lower value (maximization/minimization problems) that the one previously achieved.

CP is typically more tailored for solving decision problems and not optimization problems like the ones analyzed on this paper. CP has been successfully applied in solving problems in different areas of application such as automated timetable assignments, digital circuit verification, air traffic control and several fields of engineering.

**Cross-Entropy Method** Proposed by Rubinstein [RK04], the cross-entropy method is a general monte carlo approach to continuous optimum search combinatorial problems. The method originated from the field of rare event simulation, where very small probabilities need to be accurately estimated. The method works by generating candidate solutions via a parameterized probability distribution. The parameters are updated via cross-entropy minimization, so as to generate better samples in each passing iteration.

**Cultural Algorithm** The term cultural algorithm was originally introduced by Reynolds [Rey99] as a parallel approach to genetic algorithms. The difference is that the population of solutions in cultural algorithm includes a sense of belief in the population, where knowledge of several types (domain specific, situational, temporal, spatial) is updated by the best solutions after each iteration and communicated to other solutions leading them to alter their behavior or genome itself.

**Covariance Matrix Adaptation Evolution Strategy** A recent algorithm introduced by Nikolaus Hansen with competitive results [Han06] is refered to as Covariance Matrix Adaptation Evolution Strategy (CMA ES). Originally developed for solving multi dimensional non-linear non-convex local optimization problems, the algorithm was adapted with competitive results to also solve rugged *search space* global optimization and multi-objective problems [IHR07].

The algorithm follows a population based architecture that converges by determining a covariance matrix from of a multivariate normal distribution. The covariance matrix is adapted in each generation step to follow the sampled *search space* convergence. Higher population count provides a more reliable matrix adaptation but increases the number of required solution tests.

The CMA ES algorithm is a good state of the art example of evolution strategies algorithms thriving in the field of optimization.

**Evolution Strategies** Evolution strategies [Rec71] is another methodology that operates in shared grounds with genetic algorithms, analysing the quality / fitness values of possible generated solutions to alter the selection parameters. Generating new solutions by mutation and selection of existing ones. The interesting trait from this methodology is the self-adaptation process applied to adjust the internal selection parameters dynamically.

**Gradient Descent** Gradient descent and gradient ascent are terms coined for an optimization algorithm based on the mathematical relation of the gradient of the function, following base guidelines of the Newton's method for finding the root value of n-dimensional equations, applied to optimization. It's only applicable to functions (or estimations of functions) whose derivatives, or gradients, can be calculated.

To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or the approximate gradient) of the function at the current point. One of the most known implementations of the methodology is referred to as the Broyden-Fletcher-Goldfarb-Shanno(BFGS) method [Bro70] [Fle70] [Gol70] [Sha70].

**Memetic Algorithms** Presented by Moscato [Mos89], the term memetic algorithms refers to a whole family of hybrid versions of genetic algorithms which apply traits of local search algorithms to customize the heuristic process to the problem at hand. Generally speaking it is a school of meta heuristics focused on fitting domain knowledge local search processes into population based systems. It can be argued by some authors that this family of algorithms includes the previously mentioned cultural algorithm.

**Nested Partitions** Nested partitions was originally proposed by Shi and Olofsson in 2000 [SO00] as a search space segmentation based approach to global optimization.

In their work the authors “propose a new randomized method for solving global optimization problems. This method, the Nested Partitions (NP) method, systematically partitions the feasible region and concentrates the search in regions that are the most promising. The most promising region is selected in each iteration based on information obtained from random sampling of the entire feasible region and local search. The method hence combines global and local search.”

Their upcoming book “Nested Partitions Method, Theory and Applications”, dated for release in 2009 [SO09], might also be worth consulting.

**Neural Networks** Artificial neural networks [Hay99] have also been applied in solving optimization problems. Artificial neural networks consists of simulating the learning process of neurons of the brain. Biologically speaking in crude terms a web of neurons exchange electro-chemical synapses to shape an output recognition signal. This network of connections is somehow adapted to the characteristics of the input information. Artificial neural networks simulate this behavior by connecting several layers of simulated neurons which bear weights of influence among each other. The weights of the networks are adjusted by a process of training using series of precategorized examples. The literature is rich in different methods of training (also referred by some authors as learning) and network configurations.

**Particle Swarm Optimization** Particle swarm optimization (PSO) was originally coined by Kennedy and Eberhart [KE95], and further developed by Clerc [Cle06] as a stochastic, evolutionary and population based methodology. PSO applies the concepts of swarm intelligence to the global optimization paradigm. Swarm intelligence is referred to as the collective behavior of decentralized, self-organized systems of simple agents interacting locally with one another and with their environment. These systems follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior.

In PSO each member of the population has its position and directional velocity properties. Each generation of new solutions is calculated based on these properties. Their values alter depending on factors such as proximity distance to other particles, the neighbouring particles best quality solution, the global particles best quality solution. By scattering and regrouping the swarm explores the search space. By maintaining flock proximity the swarm converges to the optimum. Some variants of PSO assign global and local swarm leaders to have a stronger influence on the flock.

Hybrid variations of PSO have been documented in the literature, with good results, by applying technics derived from other heuristics to customize PSO to a specific problem. Variants include processes of discretization of the search space, principles of repulsion, co-evolution and stochastic memetic algorithms. The population based parallelism of PSO offers a good starting point for many hybrid experiments.

## 3.5 Summary

This chapter presented a generic introduction to the field of artificial intelligence based optimization techniques. Specific attention was given to the optimization algorithms used later in the thesis: local search algorithms random search, hill climbing and simulated annealing; and the population based methods genetic algorithms and harmony search. Other algorithms were also referred in this chapter, either for their known importance in the field of optimization algorithms or their direct influence in some of the decisions documented in this thesis.

It is important to extract from this chapter the diversity of options when it comes to optimization methods in search for the optimum in combinatorial problems. There is no single best choice. The performance of optimization algorithms is often related to the type and complexity of the problem in question. Bad configurations of good algorithms can easily undermine the convergance performance. Pondered decisions must be taken and expert analysis considered before deciding what algorithm under what configuration should be applied to solve each certain type of problems.

Some of the typical pitfalls of optimization methods that should be taken into consideration include either overly restricting or overly widening the neighbor solution search space scope, respectively leading to either getting stuck in local optimum search space hills, or requiring too many iterations to find the global optimum. Another typical pitfall of optimization algorithms is the so called long tail behavior, where badly configured neighbor search space scope relations cause inner constrains that will not allow the algorithm to converge to the global optimum, instead causing the iteration process to wander in plateaus of local optimum. When developing a new optimization algorithm or configuring optimization algorithms a balance must always be kept in mind in order to ensure the algorithm will perform in a robust and reliable manner, capable of avoiding these pitfalls in problems of different type and complexity.

# Chapter 4

# Parameter Optimizer for Expeditious Modeling

Expeditious modeling of urban environments is an area where procedural modeling applying pre-defined rules is common. The modeling rules applied can often grow large in complexity and their parameters difficult to tune. In this regard, by applying knowledge from the artificial intelligence fields of stochastic optimization and evolutionary computation we can obtain automatic systems to determine the optimum parameter combinations for different types of combinatorial problems, improving expeditious modeling systems to a higher standard of automation, reliability and realism.

## 4.1 Problem Statement

The global objective is to minimize the cost and time of resources required to generate realistic virtual models of urban environments. To achieve this there are several tools and methodologies which can be applied and followed, as described in Chapter 2.

Image analysis methods can assist in modeling, however they can't always be applied due to lack of proper equipment or direct access to the real world model. Pure manual 3D modeling approaches also have logistic problems, requiring heavy learning curves to master the tools and a great number of work hours to develop models that resemble the real world in high detail.

Procedural modeling methodologies have been developed to allow a georeferenced pipeline where the modeling process occurs based on acquired data and sets of production rules. These methodologies have the advantage of being capable of reapplying the same generation rules to different locations and datasets without requiring heavy manual 3D modeling. Access to reliable datasets of georeferenced data grows in importance. The development of relevant production rules and fine tunning of their variable parameters becomes extremely important.

For example, it often occurs that it is impossible, very troublesome or highly cost-ineffective to acquire complete accurate data for an entire city we are trying to model. However it is often possible, and relatively accessible to gather information from certain key-defining areas within a city and assign the data to a georeferenced database. Production rules can be optimized with the data collected to generate models resembling the real world area of gathered information. Such production rules can then be reapplied on areas of similar characteristics and more limited information in order to generate realistic models.

## 4.2 Requirements, Methodology and Implementation

The solution envisioned to solve this problem is the development of a hybrid meta heuristic algorithm capable of calibrating parameters of typical procedural modeling systems specially developed for expeditious modeling of virtual urban environments. Optimization problems in this specific area can vary significantly in type and complexity but share enough of a common ground on limitations and requirements that warrant special attention to the applied methodology.

### 4.2.1 Requirements

Several requirements were identified with specific importance to expeditious modelling systems of urban environments. Their importance, best envisioned methodology to satisfy them and the additional implementation constraints that surged from the proof of concept test case application are addressed in the following paragraphs.

**Domain Knowledge** The term *domain knowledge* refers to extra information that might be available or estimatable from optimization problem at hand. Some algorithms such as the school of memetic algorithms [Mos89] are known to take advantage of *domain knowledge* to constrain the problem or its *search space* and thus improve the performance of the optimization process. Similarly, gradient based approaches require mathematical knowledge of the problem which might not be available in some cases. In these cases, domain knowledge for gradient based systems can be estimated but it is not always possible.

Parameter optimization problems for expeditious modeling can include a broad range of problems in terms of complexity. Even though some simpler problems can ocasionally have some clear *domain knowledge* that could facilitate resolving the problem, this knowledge is often either unavailable for more complex problems or can simply be applied to redefine the problem in itself, altering the significant input parameters to reduce the complexity or changing the evaluation function of the problem to a more convergent *search space*.

Considering the ensivioned optimizer should be capable of solving different types of optimization problems with known or unknown *domain knowledge* alike, the proposed methodology assumes all *domain knowledge* is void focusing on the convergence of the hyper heuristic and meta heuristic.

As such, the proof of concept system assumes that no domain knowledge can ever be gathered from the optimization problem. Similarly, any possible dependency or co-relation in between the parameters and the quality function is considered unknown a priori.

**Input Parameter Types** A standard limitation to optimization algorithms is the type of input parameters they take. Some algorithms only operate with discrete values while others have been developed to function with continuous values. Optimization problems for expeditious modeling may require inputs of binary, discrete and continuous parameter types.

To satisfy this requirement the optimizer system must handle the input of boolean, integer and float types alike. This could be achieved by implementing a real coded meta heuristic algorithm, in detriment of algorithms that can only handle binary, discrete values or graph-like structures. The proposed methodology to support this requirement with competitive performance is to develop a hybrid optimization system.

In the proof of concept implementation a hybrid system inspired by generic real coded genetic algorithms (RCGA) was adapted to also handle boolean and discrete input parameters. The hybrid system was developed to allow the configuration of local search methodologies such as random search and hill climbing which may sometimes perform better in deterministic *search space*. Additionaly, characteristics trais from simulated annealing (SA) and harmony search (HS) were applied to the parent selection, cross breeding and mutation operands of RCGA, this improves the adaptability of the algorithm to different problem complexity scenarios which could be caused by having different types of input parameters with unknown corelation.

**Performance** A third requirement was defined regarding the performance of the algorithm. An optimizer without a competitive performance is not very useful to any system. Parameter optimization problems for expeditious modeling can include a broad range of problems in terms of complexity. It's imperative that the developed optimizer performs competitively to other documented algorithms regardless of its complexity.

Solving linearly constrained problems (with lack of local optimums) the algorithm must outperform simple brute force or local search meta heuristic algorithms such as the random search or hill climbing. Solving non linearly constrained problems (with co-related parameters and many local optimums) the algorithm must perform on par with modern implementations of RCGA and HS.

There are at least two distinct methods to approach this limitation. The first method would be to develop an intelligent hyper heuristic that could select the adequate heuristic to apply by analysing the constraints of the problem, this approach could prove difficult considering the no *domain knowledge* requirement previously identified, it would also not be a very elegant approach when applied to solve more lightweight types of problems.

The second method to tackle this limitation would be to develop a hybrid meta heuristic method in itself, which could be adapted for hyper heuristic performance if required, but mainly aimed for solving lightweight problems initially configured by generic common sense from previous experiences configuring the system for different optimization problems.

The proof of concept implemented the later with additional performance measurement log and visualizer system to assist in the analysis and fine tune configuration of the algorithm dealing with different types of complexity problems in itself, including hyper heuristic behaviour.

**Reconfigurability** In order to support optimization problems of different complexity, a very important requirement to the system is the capacity to be easily configurable, allowing simple, fast and/or automated reconfiguration of the optimizer, adapting itself to the specifics of problem at hand.

The methodology to satisfy this requirement involves defining configuration standards that can be verified or easily altered, plus enabling the system to self-adapt or alter itself somehow to follow the needs of a specific problem.

In the proof of concept implementation a special XML schema, deeply described further ahead in this document, was developed to validate configurations that the algorithm can interpret automatically. Additional hyper heuristic capabilities were implemented to demonstrate the capacity of simplified reconfiguration of the system. The hyper heuristic trait and a selection of internal dynamic operands that determine the behaviour of the algorithm enables the optimizer some degree of self-adaptation.

**Integration** A final important requirement of the optimizer is that it must be easy to integrate with different expeditious modeling system. Facilitated integration capacity can be generally achieved with careful architecture and class structure choices.

To address this concearn special attention should be given to the optimizer invocation and solution evaluation functions of the optimizer. Smart implementation of these functions allows the optimizer to either be invoked externally to the system or be embebbed into the source code directly. Ideally only the two key steps to define any optimization problem should be required to be implemented in the system:

1. Define the generating function (type and scope of the input parameters)
2. Define the evaluation function (determine the quality fitness of the solution)

In the proof of concept implementation, as expected, special attention was given to the architecture of the system and its class structure, described more in depth further ahead in this document. The developed optimizer was embebbed into the XL3D system [CBSF07], previously documented in Chapter 2 and can be invoked under specific configurations. The two key steps of the optimization problem are addressed by:

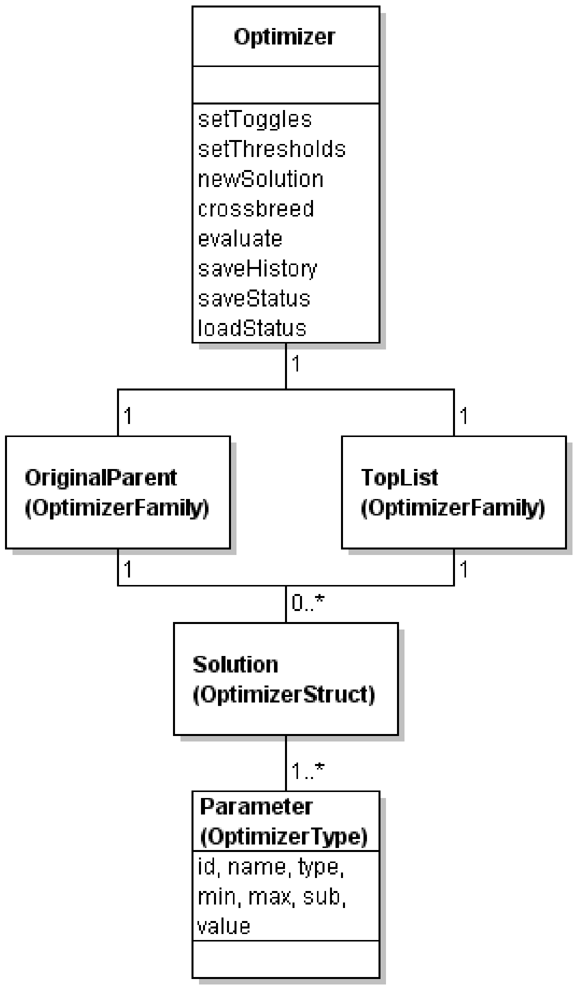
1. Writting the adequate XML configuration file that defines the problem. If required additionally programming a special function to invoke the expeditious modeling system with the defined parameters.

2. Programming a function that analyses the expeditious modeling system output and rates it accordingly.

### 4.2.2 Architecture

The foundations of the system rely on basic principles of RCGA and then incorporates other concepts derived from SA and HC. There are two families of populations resident in memory at all times. These are referred to as the *originalparent* family and the *toplist* family as can be seen in Figure 4.1.

Figure 4.1: Optimizer architecture.


Optimizer
setToggles
setThresholds
newSolution
crossbreed
evaluate
saveHistory
saveStatus
loadStatus
1
1
1
OriginalParent (OptimizerFamily)
TopList (OptimizerFamily)
1
1
0..*
Solution (OptimizerStruct)
1..*
Parameter (OptimizerType)
id, name, type, min, max, sub, value

Each of both *OptimizerFamilies* contain a list of solutions. Each solution represents a combination of values for each of the input parameters of the problem. Each parameter contains information regarding its type (boolean, integer, float) and scope (minimum and maximum values).

Each iteration step of the meta heuristic algorithm consists of creating an entire new generation of the *originalparent* family. As so, the *originalparent* family contains the newly generated solutions. While the *toplist* family contains the best solutions ever found so far, sorted by their quality. This shares traits with most standard population based heuristic systems. Each solution stores values for all parameters that are being calibrated.

The values for the entire first generation of the *originalparent* family are randomly calculated within the parameter scope. The first generation of the *toplist* family is obtained by sorting the first generation of *originalparent* family. Each new generation is obtained by cross-breeding the *originalparent* family with a chosen member of the *toplist* family. This method inspired by RCGA ideals, ensures an elitist selection behavior that will push the families to reach an optimal solution. The cross-breeding methodology for our algorithm shares traits with most standard implementations of RCGA with a concept similar to the known non uniform cross over operand.

### 4.2.3 Thresholds

There are several thresholds present that embody generalist monte carlo [MU49] stochastic optimization and specific SA ideology into the hybrid algorithm developed. As documented previously in this chapter, these thresholds can be configured to be either fixed valued or to follow a dynamic rule, the only implemented dynamic behavior so far is a simple formula of diminishing a value with each passing iteration alike the SA ideology. The value of these thresholds following the monte carlo reference is counter-measured against an always randomized value, which generates a probability of acceptance for a given action of the algorithm.

There are a total of eight threshold and two operand parameters which must be pre-configured accordingly considering the complexity of the problem. There are several problem characteristics to be taken into account when tuning these thresholds: the number of parameters to tune, the scope of type values serving as input to the problem, the correlation among the parameters, and if the problem is linearly or non linearly constrained. When dealing with *black box* type of problems, most of these characteristics are unknown and as such any configuration initially selected could only be improved by applying hyper heuristic principles to the problem.

Some of these thresholds are affected by the slow rise of entropy in the system, this allows our hybrid system to have different degrees of mutation. This dynamic property is a branded trait of SA which is often neglected in population based meta heuristics to a fixed value. Special implementations fluctuating the decision probabilities and pitch bandwidth in HS have been documented in the literature in the past. Similarly, special variations of a dynamic mutation operand in RCGA implementations are uncommon but have also been documented in the literature.

In our new hybrid meta heuristic system an internal value, referred to as *globalentropy*, increases by every generation and is calculated as a ratio corresponding to the maximum number of iteration steps applied to the optimizer, as described by (4.1). The *globalentropy* slowly rising controls the levels of mutation allowed by the algorithm described further ahead.

$$globalentropy = iterationstep/maxsteps \tag{4.1}$$

**Threshold Random New Struct** *trns* scopes real based values between 0.0 and 1.0, affecting the probability of choosing a completely random new solution. The higher this value the more probable it becomes to occur a total random creation of a new solution. A value of 0 implies that new solutions will never be random. A value of 1.0 implies that new solutions will always be random, solely depending on *globalentropy*. How this threshold is mathematically applied can be consulted in formula (4.2). HS as a similar probabilistic threshold to occasionally select entirely new solutions.

$$rand() * globalentropy \ < \ trns \ : \ randSol() \tag{4.2}$$

**Threshold Random New Type** *trnt* scopes real based values between 0.0 and 1.0, affecting the probability of choosing a completely random new value for each of the solution parameter types. The higher this value is the more probable it is to occur a totally random new value for the current parameter type of the solution. A value of 0 implies new solutions will never be random. A value of 1.0 implies that new solutions will always be random, solely depending on *globalentropy*. How this threshold is mathematically applied can be consulted in formula (4.3).

$$rand() * globalentropy \ < \ trnt \ : \ randType() \tag{4.3}$$

**Threshold Toplist Dispersion** *ttld* scopes real based values between 0.0 and 1.0, affecting the probability of choosing lower ranking *toplist* parents to cross the solution with. The higher this value the wider the scope of choice. A value of 0 is no dispersion, will always breed with best solution from *toplist*. An intermediate value of 0.5 is half dispersion, randomly choosing a member from the better half of the *toplist* family. A value of 1.0 is full dispersion, randomly selecting a member from the entire toplist family. The threshold value can be replaced with that of a function inversely proportional to the *globalentropy*. How this threshold is mathematically applied can be consulted in formula (4.4). This formula can be somewhat comparable to the roulette wheel parent selection method of RCGA [Mic94].

$$victim = (rand() * ttld * maxFamilySize) \tag{4.4}$$

**Threshold Typevalue Dispersion** *ttvd* scopes real based values between 0.0 and 1.0, affecting the parental gene influence for each value of the parameter types of the solution. The higher this value is, the higher will be the influence from the *toplist parent*. A value of 0 means no dispersion, selecting the value entirely from the *original parent* (*orv*). A value of 0.5 means half dispersion, average value between the *toplist parent* and the *original parent*. A value of 1.0 means full dispersion, selecting the value entirely from the *toplist parent* (*tlv*). The threshold value can be replaced with that of a function inversely proportional to the global entropy. How this threshold is mathematically applied can be consulted in formula (4.5). This formula can be somewhat comparable to a hybrid version of the RCGA operands of average convex cross over and direction based cross over [Dav01].

$$newvalue = (ttvd * orv) + ((1 - ttvd) * tlv) \tag{4.5}$$

**Threshold Typevalue Entropy** *ttve* scopes real based values between 0.0 and 1, affecting the probability of scope jitter for each value of the parameter types of the solution. The higher value, the broader the scope range will be. A value of 0 means no entropy, the final value won't jitter. An intermediate value of 0.5 means half entropy, the final value will jitter within 50% of the parental value difference scope. A value of 1.0 means full entropy, the final value jitters full scope. The threshold value can be replaced with that of a function inversely proportional to the global entropy. How this threshold is mathematically applied can be consulted in formula (4.7). This threshold can be compared to the *pitch* adjustment of HS and several recent RCGA mutation operands described in the literature.

$$scope = \|(OParentValue - TLParentValue)\| \tag{4.6}$$

$$igl = (1.0F - globalentropy) \tag{4.7}$$

$$ttvss = ttvss * ttvsv * igl \tag{4.8}$$

$$range = MaxParamValue - MinParamValue \tag{4.9}$$

$$scope < ttvss * range \ : \ maxscope = range * ttvsv \tag{4.10}$$

$$scope > ttvss * range \ : \ maxscope = scope * ttve \tag{4.11}$$

$$maxscope = scope * rand() \tag{4.12}$$

$$newvalue = newvalue + scope - (maxscope/2) \tag{4.13}$$

**Other Thresholds and Operands** Three other *thresholds* and two *operands* were implemented to the configuration of the algorithm with minor importance to the capacity of the algorithm or the work developed in this thesis.

- Threshold Typevalue Scope Scope (*ttvss*)
- Threshold Typevalue Scope Value (*ttvsv*)
- Threshold Nested Partition (*tnp*)
- Crossover Operand (*co*)
- Mutation Operand (*mo*)

Both *ttvss* and *ttvsv* are only applied in formulas 4.9 and 4.11, affecting slightly how the mutation operand jitters the final solution. For most of the tested development of the algorithm their values have been default to following *globalentropy.* Altering their behavior for other dynamic formulas might offer interesting results following some of the properties of certain documented RCGA operands.

The nested partition threshold refers to a yet experimental implementation of nested partition ideals applied to dynamically reducing the search space to areas with interesting results in an attempt to diminish the number of test solutions that never yield interesting results. The danger lies in configuring the algorithm to over restrict the search scope, unwantedly cutting off from the search space the global optimum. Tests with this threshold were disregarded for the scope of this thesis.

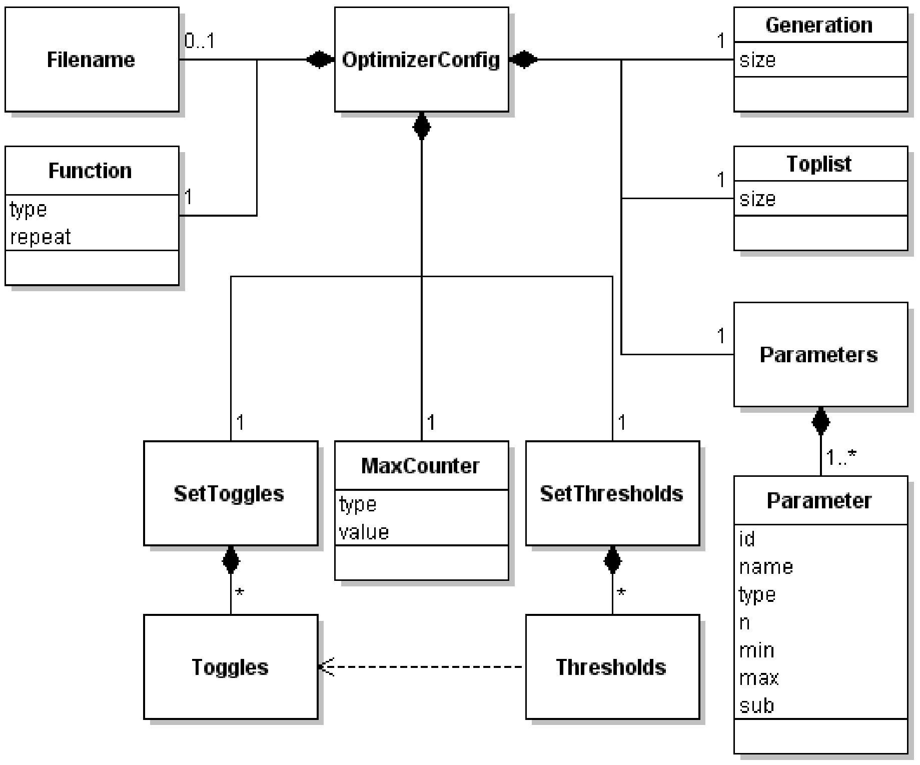
The crossover and mutation operands are design decisions aimed at further development of the algorithm to allow direct comparative work of the hybrid with some previously documented or newly conceptualized operands of RCGA and HS literature. The multiplicity of possible operands play no part on this thesis.

## 4.2.4 XML Configuration Schema

To satisfy the reconfigurability requirement of the optimizer, the proof of concept implementation supports a newly defined XML schema to read and interpret the configuration files. This section describes the implementation and provides a good introduction to the internal workflow of the optimizer system and its architecture.

A valid XML configuration file has a single *OptimizerConfig* type with the element *name* identifying the configuration. Each *OptimizerConfig* includes eight subnode types as seen in Figure 4.2.

Figure 4.2: XML configuration nodes scheme.



The *Filename* type is optional, applicable only for invoking hyper heuristic configurations, using its *name* element to identify the XML template that should be modified.

The *Function* type is required to identify the internal function *type* that will be executed in the algorithm and the number of *repeat* trials before re-attempting a new combination when handling problems with post-calc constraints.

The *MaxCounter* type defines how the iteration process of the algorithm operates, its *type* indicates different operation modes, 0 or 1 implies an counts incremental or decremental a certain *value* of steps respectively, a *type* value of 2 or 3 implies the process will repeat iteratively until the quality value drops below the *value* described.

The *SetToggles* type pertains to configure the threshold type of the algorithm. for the 8 thresholds it operates as 0 for static threshold value and 1 for a dynamic value variation following the *globalentropy* value described further ahead in this chapter. Higher values for these 8 thresholds are reserved for other dynamic operands that are not yet implemented on the system. The *co* and *mo* elements pertain to the crossover operand and mutation operand type selected, envisioning future development to implement specific operands documented for real coded genetic algorithms and harmony search implementations, among other possibilities this would allow an easy and improved comparative analysis to take place in the future.

The *SetThresholds* type registers the values for the 8 thresholds available for configuration in the algorithm, with values typically ranging between 0.0 and 1.0. How these values affect the algorithm also depends on the toggle values described on the previous paragraph and of course the internal algorithm which shall be addressed further ahead in this chapter.

The *Parameters* type englobes one or several subnodes of *Parameter* types. These *Parameter* types contain several elements, an *id* and a *name* to identify them separately, a *type* defining if the parameter is of boolean (0), integer (1) or float (2) type. An optional element *n* used for functions where the same parameter can have several instances, being that the n would indicate the number of instances, the default by omission is of course 1 instance per parameter. Additionaly the *min*, *max* and *sub* fields determine the scope of the parameter, being that the *min* and *max* must be ranged in positive numbers only due to internal convergence calculus requirements, and the *sub* element indicates how much should be subtracted to this allocated range to displace the scope to the desired range, shall only be any different of 0.0 in case of negative scopes.

The *Generation* and *Topslist* types have a single element *size* defining the dimension of the *originalparent* and *toplist* family sizes respectively. Their relation to the algorithm will be described further ahead in this chapter.

An example of a valid XML configuration used by the Optimizer follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<OptimizerConfig name="gp1_h1n">
 <Function type="2" repeat="10"/>
 <MaxCounter type="0" value="100"/>
 <SetToggles trns="0" trnt="0" ttld="0" ttvd="1" ttve="0"
             ttvss="1" ttvsv="1" tnp="0" co="0" mo="0"/>
 <SetThresholds t1="0.01" t2="0.01" t3="0.1" t4="0.0"
                t5="0.1" t6="0.0" t7="0.0" t8="0.0"/>
 <Parameters>
  <Parameter id="x1" name="x1" type="2"
             min="0.0" max="10.0" sub="5.0"/>
  <Parameter id="x2" name="x2" type="2"
             min="0.0" max="10.0" sub="5.0"/>
 </Parameters>
 <Generation size="20"/>
 <Topslist size="30"/>
</OptimizerConfig>
```

The example of the optimizer configuration above will execute 100 iterations with a *originalparent* family size of 20 and a *toplist* family size of 30, optimizing two parameters of a well known mathematical function first presented by Goldstein & Price [GP71] for benchmarking global optimization problems. This function takes 2 parameters, both scoping between -5 and 5.

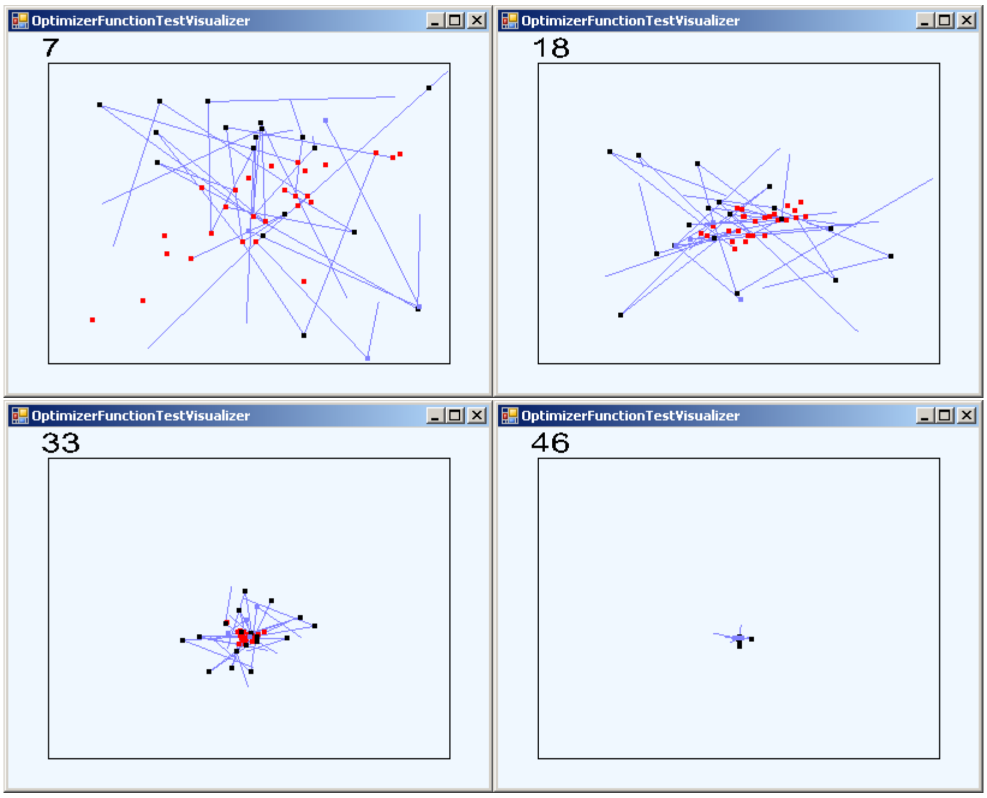
### 4.2.5 Performance Monitoring

The optimizer was developed to output a series of logs where the parameter and quality values of each tested solution is registered in physical memory by the *saveStatus* method. Also registered in physical memory are the contents of both *originalparent* and *toplist* family of solutions for each iteration step. This allows the user to analyse the data generated and determine how the algorithm is converging to the optimum.

An additional wrapper log was developed to calculate the average values of several runs of the optimizer under the same configuration. This allows the performance between different configurations to be compared statistically, analysing the progression of the algorithm through the iterative steps with information regarding their algorithmic mean and standard deviation. This information is useful to help calibrate and justify performance enhancement theories between the different configurations.

The complete history of the optimizer can also be loaded and saved into physical memory. A simple visualizer was embebbed to allow easier supervision of convergence performance characteristics of the algorihtm executing under different configurations. The visualizer, seen on Figure 4.3, was developed for viewing functions with 2 input parameters only. The screenshots from different step iterations show in red dots the *toplist* family members, in black the newly calculated *original parent* family members and with blue lines the position of the parents that originated the current solution marked with the black dot.

Figure 4.3: Visualizer steps showing convergence of a 2D function.


OptimizerFunctionTestVisualizer
7
OptimizerFunctionTestVisualizer
18
OptimizerFunctionTestVisualizer
33
OptimizerFunctionTestVisualizer
46

## 4.3 Summary

This chapter described the optimizer solution developed to assist in expeditious modeling of virtual urban environments. It refers the importance and the context of the application field. It also approaches the defined requirements and how they conditioned the methodology and the proof of concept implementation of the optimizer. Finally it covers how the system can be configured, its basic architecture, internal algorithm thresholds and how the performance can be monitored.

The optimizer algorithm is an hybrid version of real coded genetic algorithms with traits of simulated annealing and harmony search. The algorithm was developed aiming to solve global optimization problems that have no access to domain knowledge information and require support for boolean, integer and float type of input parameters. The algorithm was expected to perform competitively with other standard implementations of similar algorithms.

This chapter is crucial to this document as it exposes the context, requirements, methodology and proof of concept implementation specifics of a system developed to simplify and automate the optimum parameterization of production rules in expeditious modeling systems of virtual urban environments.

# Chapter 5

# Evaluation

In order to test the capacity and performance of the system, several benchmark tests were conducted. This chapter is divided in two sections, the first regarding the performance evaluation of the developed system under different configurations and sets of standard non-convex non-linear global optimization problems. The second section documents how the system was used in a real test case scenario. Detailing the process of calibration by parameterization of an estimated modeling formula and the results obtained by reapplying that formula in other modeling areas with missing data.

## 5.1 Optimizer Benchmark

Some tests were performed to guarantee with some degree of certainty that the new meta heuristic algorithm offers competitive results against other types of algorithms. These tests compare results of different configurations of our algorithm under four well documented equation minimization problems. Due to its nature, the optimizer system allows for different configurations of the thresholds. When selected carefully these threshold configurations mimic the behavior of basic optimum search algorithms and serve as a general comparative term with the hybrid.

The first sub section lists what configurations of the algorithm were tested. The following four sub sections present the global optimization functions used to preliminary test the performance of the algorithm during development, presenting results for each of the tested configurations. the final sub section is dedicated to analysing the performance of the algorithm in some of the problems from the CEC'05[1] set [SHL+05].

### 5.1.1 Tested Configurations

We selected a total of 7 different configurations of the meta heuristic algorithm to test the performance on distinct combinatorial problems. These configurations all shared a fixed *toplist* family size of 20, iterating 50 generations each.

Each configuration was labeled accordingly. The configuration labeled as *rnd* for example represents a pure random search algorithm configuration where each next solution is selected randomly out of the entire solution universe.

The configuration labeled as *hhc*, referring to a hybrid hill climbing, behaves as a standard hill climbing version of local search algorithm, always generating next solutions based on the best solution found so far and using a very close quartered neighbor solution scope.

The configuration labeled as *hsa*, hybrid simulated annealing, represents a standard local search algorithm and behaves diminishing the acceptance rate of other solutions beyond the best and also diminishing the neighbor scope progressively, so that in the beginning of the iteration a broader search range is available and the next solution isnt always calculated based on the best solution found so far, as more iteration steps occur their probability diminishes progressively mimicking the behavior of simulated annealing.

Other four configurations were tested with parameters previously known to offer some good results under all different kind of combinatorial problems. These were labeled as *h1n*, *h2n*, *h3n* and *h4n*. They attempt to take advantage of the hybrid nature of the algorithm representing hybrid versions of semi-elitist genetic algorithm behavior and some principles behind harmonic search. The parameters of all these configurations can be consulted in table 5.1.

To determine the efficiency and robustness of our meta heuristic algorithm we tested its configurations performance against a batch of well documented combinatorial problems originally defined by Rosenbrock [Ros60], Goldstein and Price [GP71] and Colville [Col68]. The choice for this batch of problems follows Lee's 2005 paper "a new meta

[1]IEEE Congress on Evolutionary Computation 2005 Special Session on Real-Parameter Optimization, often used in recent literature to compare performance between state of the art global optimization algorithms

Table 5.1: Tested configurations

| config | trns | trnt | ttld | ttvd | ttve |
|---|---|---|---|---|---|
| rnd | 1.0 | 1.0 | 0.0 | 0.5 | 0.5 |
| hhc | 0.001 | 0.001 | 0.0 | 0.0 | 0.1 |
| hsa | 0.01 | 0.01 | 0.0 | ge | ge |
| h1n | 0.01 | 0.01 | 0.1 | ge | 0.1 |
| h2n | 0.001 | 0.001 | 0.4 | 0.15 | ge |
| h3n | 0.001 | 0.001 | 0.1 | 0.1 | ge |
| h4n | 0.001 | 0.001 | 0.01 | 0.01 | 0.01 |

heuristic algorithm for continuous engineering optimization HS theory and practice" [LG05] regarding the applicability of HS in solving combinatorial problems effectively.

The optimal solution for each of the four tested functions can be consulted in table 5.2

Table 5.2: Tested Functions

| Functions | Optimal Solutions |
|---|---|
| Rosenbrock function (5.1) | $f^*(x) = 0.0, x_1^* = 1.0, x_2^* = 1.0$ |
| Goldstein and Price function-I (5.2) | $f^*(x) = 3.0, x_1^* = 1.0, x_2^* = -1.0$ |
| Goldstein and Price function-II (5.3) | $f^*(x) = 1.0, x_1^* = 3.0, x_2^* = 4.0$ |
| Wood function (5.4) | $f^*(x) = 0.0, x_1^* = 1.0, x_2^* = 1.0, x_3^* = 1.0, x_4^* = 1.0$ |

The results presented in the following subsections were obtained by calculating the average values of 200 simulations for each tested configuration. Each of the 200 simulations executing 100 iterations of the algorithm under the corresponding configuration with a *original parent* family size of 20 and a *toplist parent* family size of 30. The values on the following Tables pertain to the quality value of the best solution found at 0%, 40%, 80% and 100% of the 100 iteration steps. The third and the final column of these tables refers to the standard deviation calculated for the initial and final iteration step of the 200 simulations.

An additional hybrid hyper heuristic configuration *hhh* was determined for each function by applying the algorithm recursively in hyper heuristic fashion under the threshold configuration of $\{0.001, 0.001, 0.4, 0.15, ge\}$, executing 50 iterations with *original parent* family size of 20 and *toplist parent* family size of 30. The threshold configuration was estimated as having a good performance by user experience obtained while implementing the system but also largely inspired by several documentats refering to the performance between variants of SA, RCGA and HS considering their different

Table 5.3: Hyper heuristic configurations

| function | trns | trnt | ttld | ttvd | ttve |
|---|---|---|---|---|---|
| ros | 0.0007 | 0.0005 | 0.165 | 0.1397 | ge |
| gp1 | 0.001 | 0.0005 | 0.3644 | ge | ge |
| gp2 | 0.0015 | 0.0014 | 0.1346 | 0.0559 | ge |
| wood | 0.0006 | 0.0007 | 0.276 | 0.1561 | ge |

ratios that are embebbed into the hybrid system.

For each of the test functions described ahead, each of the solutions tested (500 per function) had their fitness valued as the final value average from 10 configuration simulations each. The scope of the variable configuration parameters on the hyper heuristic were as follows: *trns* [0.0005..0.002], *trnt* [0.0005..0.002], *ttld* [0.1..0.4], *ttvd* [0.05..0.2]. Scope limits to these parameters were applied to speed the process of convergence. The best hyper heuristic configuration found for each of the test functions can be consulted in table 5.3

## 5.1.2 Rosenbrock Function

$$f(x) = 100(x1 - x2^2)^2 + (1 - x1)^2 \tag{5.1}$$

Rosenbrock introduced the function shown in 5.1 in his article "An Automatic Method for Finding the Greatest or Least Value of a Function" [Ros60] which has grown to be one of the most known equations for benchmarking continuous parameter optimum search algorithms.

This equation is interesting due to its valley like shape as seen in Figure 5.1. Due to this property, the traditional hill climbing behavior algorithms usually require a great number of iterations to converge to the optimum solution. The optimum minimum solution for the Rosenbrock algorithm is $x^* = (1.0, 1.0)$ with a quality function value $f^*(x) = 0.0$.

The algorithm was applied to the Rosenbrock function problem with scope of -10, 10 for both of the variable parameters, x1 and x2. Results are presented in Table 5.4.

The initial iteration $\mu$ and $\sigma$ columns offer us a general overview of how widely parameter scoped and fluctuating quality the Rosenbrock function search space truly is. All configurations quickly converge to a local search area surrounding the optimum. Due to the small dimension of the solution space, even a pure random search can offer,

Figure 5.1: Rosenbrock function.

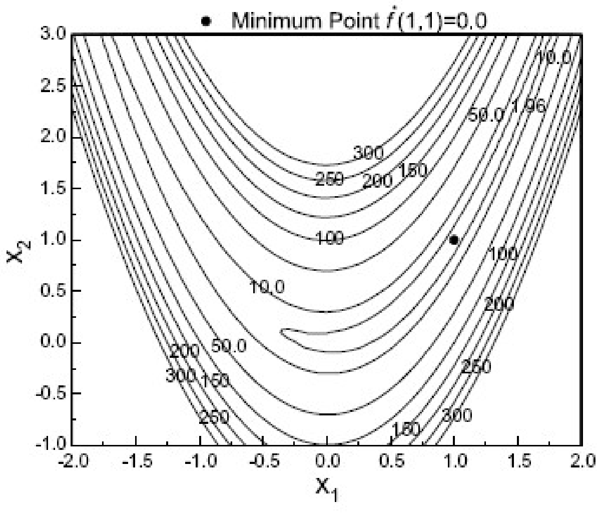
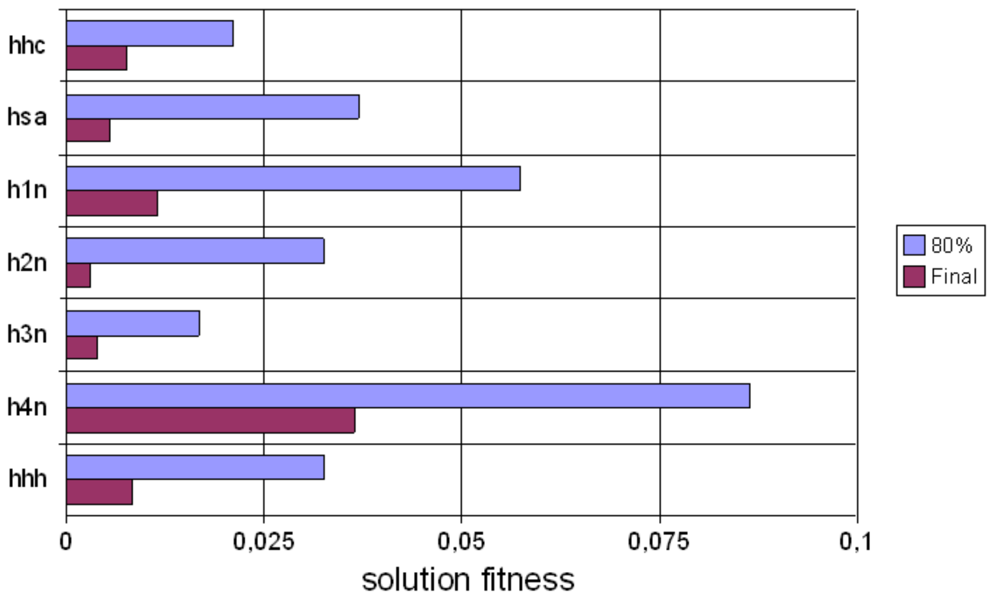


Table 5.4: Average results of optimizing the Rosenbrock function under the different configurations

| config | init $\mu$ | init $\sigma$ | 40% $\mu$ | 80% $\mu$ | final $\mu$ | final $\sigma$ |
|---|---|---|---|---|---|---|
| rnd | 259.62 | 442.48 | 1.1367 | 0.6801 | 0.61382 | 1.48371 |
| hhc | 305.13 | 525.05 | 0.5510 | 0.0212 | 0.00768 | 1.61623 |
| hsa | 256.39 | 406.75 | 1.3273 | 0.0370 | 0.00556 | 1.42254 |
| h1n | 331.74 | 585.03 | 0.9108 | 0.0574 | 0.01156 | 1.70605 |
| h2n | 294.98 | 480.78 | 0.7082 | 0.0325 | 0.00305 | 1.54659 |
| h3n | 308.88 | 507.20 | 0.6136 | 0.0168 | 0.00396 | 1.58852 |
| h4n | 284.30 | 521.83 | 0.4242 | 0.0864 | 0.03647 | 1.61127 |
| hhh | 260.85 | 425.16 | 0.6576 | 0.0327 | 0.00827 | 1.45438 |

in average, decent results under small number of iterations. To achieve more accurate results the balanced non elitist centered configurations such as *h2n* and *h3n* seem to be more reliable when roaming the search space, especially in comparison with elitist centered configurations such as *h4n* as seen in Figure 5.2. The performance of *hhh* suggests that even though hyper heuristic can be helpful in finding good solutions, the configuration found is still somewhat far from the global optimum.

Figure 5.2: Average results graph for the Rosenbrock function.



### 5.1.3 Goldstein and Price Function I

In [GP71] Goldstein and Price introduced an eight order polynomial function of two variables and four local minima as seen in Figure 5.3

Figure 5.3: Goldstein and Price function I.

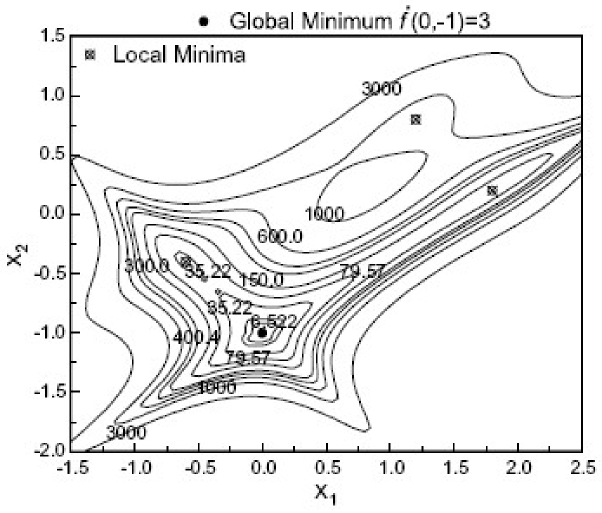


Table 5.5: Average results of optimizing the Goldstein and Price function under the different configurations

| config | init $\mu$ | init $\sigma$ | 40% $\mu$ | 80% $\mu$ | final $\mu$ | final $\sigma$ |
|---|---|---|---|---|---|---|
| rnd | 1615.42 | 3805.92 | 19.8780 | 10.3636 | 9.11594 | 4.35143 |
| hhc | 1454.01 | 3263.58 | 5.2758 | 3.0035 | 3.00001 | 4.02948 |
| hsa | 1599.58 | 4619.95 | 22.2146 | 3.1502 | 3.00003 | 4.79425 |
| h1n | 1511.71 | 3741.57 | 14.2214 | 3.1206 | 3.00003 | 4.31449 |
| h2n | 1474.86 | 3234.23 | 10.7610 | 3.0510 | 3.00005 | 4.01132 |
| h3n | 1612.40 | 3984.30 | 11.2939 | 3.0263 | 3.00001 | 4.45223 |
| h4n | 1543.38 | 3124.26 | 6.0594 | 3.0016 | 3.00001 | 3.94254 |
| hhh | 1359.91 | 3990.07 | 14.0894 | 3.0924 | 3.00020 | 4.45546 |

This function, described mathematically in Equation 5.2, has an optimum solution of $x^* = (0.0, -1.0)$ with a quality function value $f^*(x) = 3.0$. The local minima are located at $x^* = (1.2, 0.8)$, $x^* = (1.8, 0.2)$ and $x^* = (-0.6, -0.4)$ with quality function values of $f^*(x) = 840.0$, $f^*(x) = 84.0$ and $f^*(x) = 30.0$ respectively. The search space for the problem have been limited with scope of -10, 10 for both of the variable parameters.

$$\begin{aligned} f(x) = {} & (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \\ & \times (30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)) \end{aligned} \tag{5.2}$$

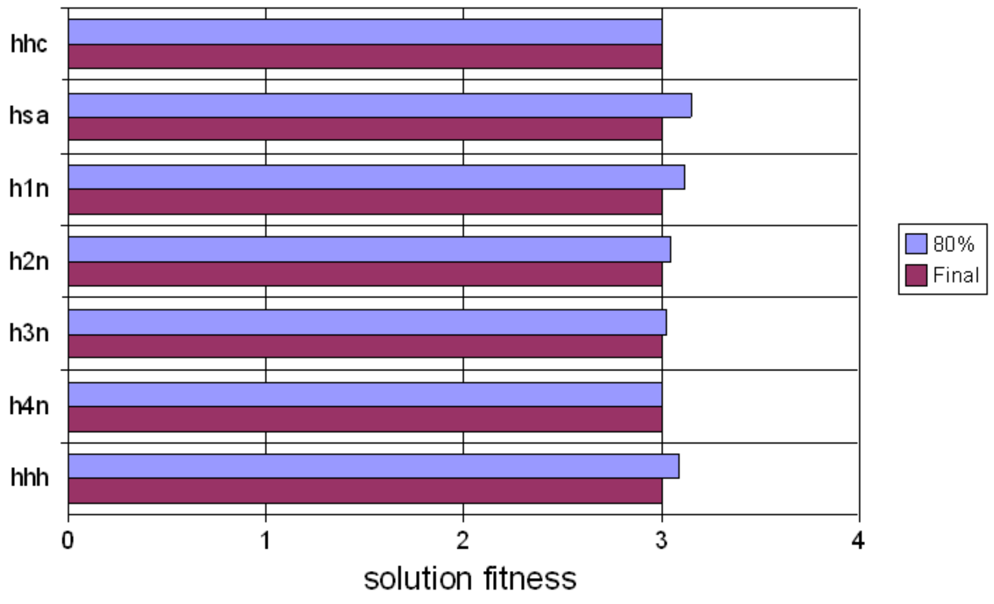
From the results in Table 5.5 we can conclude that despite very steep quality values when randomly sampling the solution search space, all configurations of the algorithm quickly converged to the optimum. Opposed to the results for the Rosenbrock function, in the Goldstein and Price function more elitist configurations such as *hhc* and *h4n* offer slightly superior results when compared to the other non elitist configurations. Overall all the configurations offer very good and close results as can be better observed in Figure 5.4.

### 5.1.4 Goldstein and Price Function II

$$f(x) = \exp\left\{\frac{1}{2}(x_1^2 + x_2^2 - 25)^2\right\} + \sin^4(4x_1 - 3x_2) + \frac{1}{2}(2x_1 + x_2 - 10)^2 \tag{5.3}$$

Another interesting function introduced by Goldstein and Price [GP71] is described mathematically in Equation 5.3. This function has many local minima, the optimum

Figure 5.4: Average results graph for the Goldstein and Price function.



solution is $x^* = (3.0, 4.0)$ with a quality function value $f^*(x) = 1.0$. The search space for the problem have been limited with scope of -5, 5 for both of the variable parameters.

From the results in Table 5.6 we can conclude that despite an extremely steep quality values shift in the quality function of the solution space, all configurations of the algorithm offer good results. These results can be better observed in Figure 5.5.

### 5.1.5 Wood Function

$$\begin{aligned} f(x) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &+ 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1) \end{aligned} \tag{5.4}$$

The fourth function tested is referred to as the Wood function. As mathematically described in Equation 5.4, this is a fourth degree polynomial function. It has been heavily used as an important convergence test in the past due to its similarity with many physics problems [Col68]. The optimum solution is $x^* = (1.0, 1.0, 1.0, 1.0)$ with a quality function value $f^*(x) = 0.0$. The search space for the problem have been limited with scope of -5, 5 for all four of the variable parameters.

Table 5.6: Average results of optimizing the Goldstein and Price function II under the different configurations

| config | init $\mu$ | init $\sigma$ | 40% $\mu$ | 80% $\mu$ | final $\mu$ | final $\sigma$ |
|---|---|---|---|---|---|---|
| rnd | 249.10 | 1719.64 | 1.2596 | 1.1738 | 1.15081 | 2.92496 |
| hhc | 318.92 | 1980.36 | 1.0474 | 1.0347 | 1.03433 | 3.13887 |
| hsa | 401.09 | 2207.65 | 1.0581 | 1.0414 | 1.03470 | 3.31411 |
| h1n | 180.21 | 1408.29 | 1.0529 | 1.0425 | 1.03417 | 2.64697 |
| h2n | 180.69 | 1408.68 | 1.0482 | 1.0369 | 1.03438 | 2.64733 |
| h3n | 324.90 | 1979.74 | 1.0509 | 1.0382 | 1.03619 | 3.13838 |
| h4n | 531.06 | 2598.78 | 1.0454 | 1.0375 | 1.03510 | 3.59573 |
| hhh | 180.71 | 1408.68 | 1.0499 | 1.0365 | 1.03489 | 2.64733 |

Table 5.7: Average results of optimizing the Wood function under the different configurations

| config | init $\mu$ | init $\sigma$ | 40% $\mu$ | 80% $\mu$ | final $\mu$ | final $\sigma$ |
|---|---|---|---|---|---|---|
| rnd | 776.37 | 635.20 | 41.9352 | 26.1354 | 22.16171 | 1.77769 |
| hhc | 779.56 | 617.26 | 16.4876 | 2.7164 | 2.03788 | 1.75241 |
| hsa | 832.14 | 628.99 | 30.1377 | 3.6724 | 2.52855 | 1.76899 |
| h1n | 727.10 | 660.86 | 28.1528 | 3.1471 | 1.91683 | 1.81324 |
| h2n | 774.08 | 686.43 | 17.7773 | 2.2228 | 1.18706 | 1.84799 |
| h3n | 810.51 | 669.87 | 21.3257 | 2.4911 | 1.71672 | 1.82556 |
| h4n | 779.16 | 635.44 | 16.8759 | 3.1982 | 2.39205 | 1.77803 |
| hhh | 801.21 | 638.09 | 15.8935 | 2.6514 | 1.75467 | 1.78173 |

Figure 5.5: Average results graph for the Goldstein and Price function II.

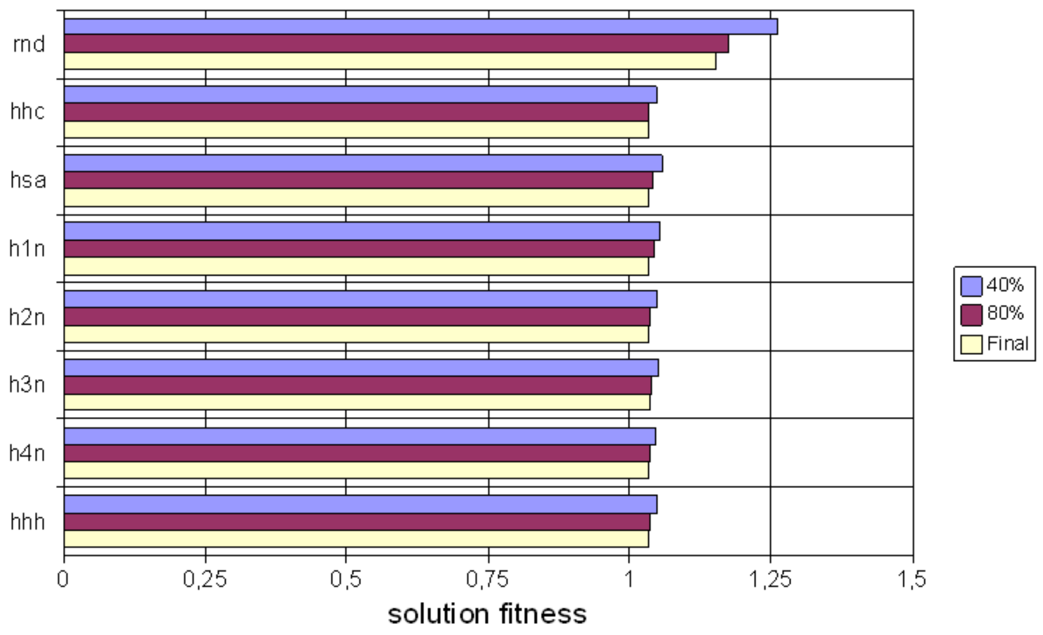
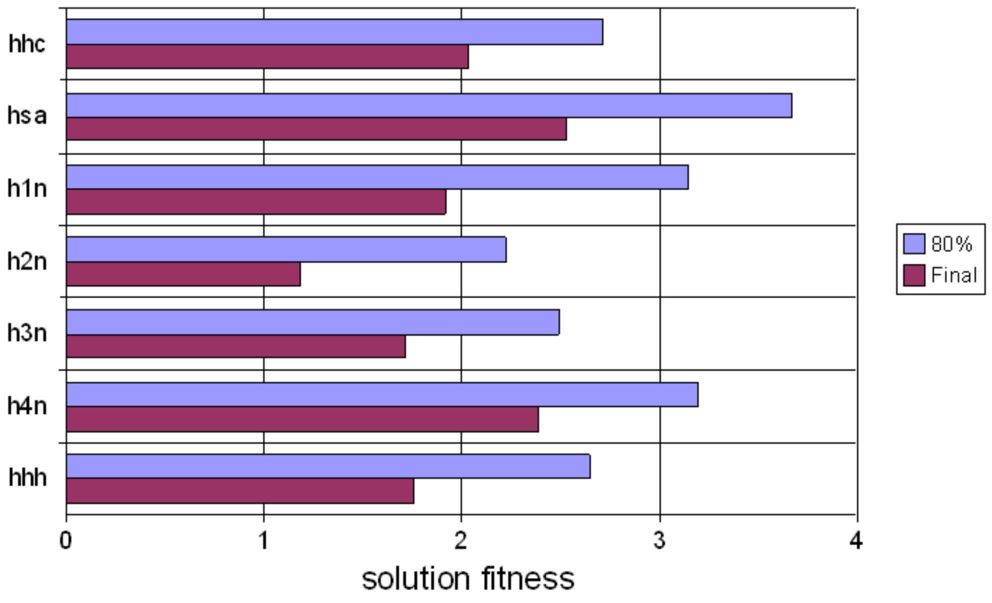


Figure 5.6: Average results graph for the Wood function.



By looking at Table 5.7 we can conclude that despite offering results much superior to the standard random search configuration *rnd*, all the remaining configurations don't quite manage to achieve satisfactory results close to the optimum quality. The Wood function's increased convergence complexity, comparatively to the other three problems tested, requires a lengthier iteration period of convergence to present good results regardless of the characteristics of the used configuration. The results can be better observed in Figure 5.6.

### 5.1.6 CEC 2005

To satisfy further academic requirements the performance of the system was additionaly tested against some of the standard batch of non-convex non-linear continuous problems from the IEEE Congress on Evolutionary Computation 2005 Special Session on Real-Parameter Optimization [SHL+05].

The configuration for the hybrid algorithm was first over fitted to one of the standard problems, the *Rosenbrock* at $D = 10$ dimensions[2]. The configuration was then tested in the *Sphere*, *Schwefel 1.2*, *Rosenbrock*, *Rastrigin* and *Weierstrass* problems at $D = 10$ and $D = 30$ dimensions each.

The average results of 25 runs of each problem can be consulted in Table 5.8. The first column shows the name of the test problem while the following columns show the average from 25 runs at the steps of 2%, 40%, 80% and 100% of the optimization process. The first 5 problems were executed at $D = 10$, with a generation of 10, executing $10 * 10^5$ function evaluations, the last 5 were executed at $D = 30$, with a generation of 30, executing $30 * 10^5$ function evaluations.

The original CEC'05 Special Session on Real-Parameter Optimization [SHL+05] consisted of benchmarking 11 algorithms each optimizing 25 different test functions under $D = 10$ and $D = 30$. Measuring which ones would solve the given problems under the target precision error of $10^{-8}$ under each function for $D * 10^5$ function evaluations.

The presented table does not relate directly to the results obtained from CEC'05 session [Han06] but analysing those results and comparing them to the performance of our algorithm allows us to conclude that despite performing below the state of the art, the hybrid system has an acceptable performance optimizing non-convex non-linear types of problems.

[2]all global optimization problems from the CEC'05 batch are designed as calculus sums of finite dimensions, increasing the number of dimensions exponentially increases the complexity of the problem

Table 5.8: Average quality function at different stages of the convergence.

| problem | 2% $\mu$ | 40% $\mu$ | 80% $\mu$ | final $\mu$ |
|---|---|---|---|---|
| Sphere | 2.880249e+004 | 5.069603e-007 | 7.443773e-010 | 2.746327e-016 |
| Schwefel 1.2 | 4.834547e+005 | 3.512164e-006 | 6.751018e-009 | 1.194683e-015 |
| Rosenbrock | 3.028539e+010 | 1.190515e+003 | 3.619151e-002 | 3.720837e-005 |
| Rastrigin | 1.533455e+002 | 3.236556e-005 | 9.360353e-008 | 3.327253e-014 |
| Weierstrass | 1.463048e+001 | 2.082120e-001 | 4.062040e-002 | 2.013143e-002 |
| Sphere | 1.133168e+005 | 7.510152e-005 | 7.610557e-008 | 5.908891e-012 |
| Schwefel 1.2 | 2.520423e+007 | 1.522236e-002 | 1.362870e-005 | 6.535891e-010 |
| Rosenbrock | 1.412603e+011 | 4.283153e+002 | 9.933642e+000 | 3.956425e+000 |
| Rastrigin | 5.652055e+002 | 7.956591e-001 | 8.582548e-006 | 6.416542e-010 |
| Weierstrass | 4.915724e+001 | 1.468030e+000 | 4.400829e-001 | 1.226722e-001 |

These results are satisfactory since the aim for the development of the hybrid system was to develop a new functional implementation of an interoperable algorithm, not an algorithm competitive with the state of the art. Despite not testing all of the IEEE CEC 2005 proposed problem set, the comparative analysis shows that the developed hybrid algorithm is capable of solving these problems despite not having been developed strategically as a state of the art pure non-convex non-linear global optimization algorithm, but as a lightweight, portable, reconfigurable and flexible new prototype to embed in expeditious modeling systems.

As previously stated, expeditious modeling systems have additional requirements beyond performing competitively against pre-defined sets of non-convex non-linear problems of continuous nature. Expeditious modeling optimizer systems are required to equally assist competitively in solving problems of different type and complexity, with mixed discrete and continuous parameters. Overall, the system was aimed to provide a solution capable of performing both the bluntly configured short iteration phases aimed to determine generic guidelines to solve an optimization problem, and the extremely fine tuned configuration for long run convergence for obtaining the global optimum of complex problems.

## 5.2 Test Case Application

A test case was prepared with XL3D modeler involving the parameterization of production rules to determine missing height values from a collection of data. The known information for each georeferenced building included the values of the perimeter, area and *bottomzvalue*.

The production rule used to estimate the height of a building implies a possible relation between the building's area (5.5), the building's perimeter (5.7) and the building's *bottomzvalue* (5.6) to the building's height. The exact formula is described mathematically in (5.8).

$$av = (cra - 1) * fca * area \tag{5.5}$$

$$ab = (crb - 1) * fcb * bottomzvalue \tag{5.6}$$

$$ap = (crp - 1) * fcp * perimeter \tag{5.7}$$

$$topzvalue = avgz + disp * (av + ab + ap) \tag{5.8}$$

The formula has a total of 8 unknown fields that can be parametrized:

- $avgz \in [100.0..120.0]$, the average height for all the buildings;

- $disp \in [0.0..1.0]$, the dispersion rate from the average height;

- $cra \in [0..3]$, area correlation signal;

- $crb \in [0..3]$, *bottomzvalue* correlation signal;

- $crp \in [0..3]$, perimeter correlation signal;

- $fca \in [0.0..1.0]$, area correlation factor;

- $fcb \in [0.0..1.0]$, *bottomzvalue* correlation factor;

- $fcp \in [0.0..1.0]$, perimeter correlation factor.

This formula is a basic empiric trial. Clearly the features of a building will never all follow this formula. In that sense more complex formulations could easily be devised with architectural knowledge and historic region background to obtain better results. This formula is merely a demonstration that under a working system, even basic relational formulas can assist in producing realistic values with a more sensible methodology compared to common trial and error sessions.

### 5.2.1 Calibration

A calibration test was carried out to compare the performance of different configurations under the complexity of the specific problem formula.

This test compared some of the optimization configurations previously presented. The parameters of the different configurations of this specific test can be consulted in Table 5.9. Each configuration of the meta heuristic algorithm was tested with a fixed *toplist* family size of 20. Each test iterated 50 generations with a family size of 8.

Table 5.9: Threshold parameters of the different configurations.

| config | trns | trnt | ttld | ttvd | ttve |
|---|---|---|---|---|---|
| rnd | 1.0 | 1.0 | 0.0 | 0.5 | 0.5 |
| h1n | 0.01 | 0.01 | 0.1 | 1.1 | 1.1 |
| h2n | 0.001 | 0.001 | 0.4 | 0.15 | 1.1 |
| h3n | 0.001 | 0.001 | 0.1 | 0.1 | 1.1 |
| hhc | 0.01 | 0.01 | 0.0 | 1.0 | 0.1 |
| hsa | 0.01 | 0.01 | 0.0 | 1.0 | 1.1 |

The quality function used was calculated as a weighted sum of the height difference from 30% randomly selected buildings from the dataset. An additional weight to the quality function was the global average height difference of the buildings from the dataset without any data flaws.

All simulations were performed twice to present a glimpse on how deeply the performance of the meta heuristics algorithm could be affected by its stochastic nature. Two runs are clearly not significant from a valid statistical reference point of view but a compromise had to be made considering the long execution time of each run. In this sense, results from two executions offer significant more information then one single run of each configuration.

Table 5.10 displays the progressive results obtained from our test case. Similarly to the mathematical test functions previously presented this table of results shows the quality value of the best solution found at 2%, 40%, 80% and at the end of the iteration process, allowing us to speculate on how each configuration depended on the initial state and perform comparatively to known random search, hill climbing and simulated annealing algorithms.

The results allow us to conclude that our hybrid meta heuristic algorithm, in this specific problem, when tuned the right way, can outperform standard random search

Table 5.10: Progressive solution quality results from the different configurations.

| config | 2% | 40% | 80% | final |
|---|---|---|---|---|
| rnd-1 | 163.95 | 54.254 | 44.854 | 44.854 |
| rnd-2 | 373.90 | 98.338 | 14.557 | 14.557 |
| h1n-1 | 272.53 | 9.319 | 9.319 | 9.319 |
| h1n-2 | 29.479 | 10.910 | 8.972 | 8.972 |
| h2n-1 | 438.53 | 24.047 | 14.425 | 14.425 |
| h2n-2 | 1038.61 | 1.758 | 1.758 | 1.758 |
| h3n-1 | 22.164 | 6.418 | 0.175 | 0.175 |
| h3n-2 | 288.56 | 3.934 | 0.798 | 0.798 |
| hhc-1 | 117.35 | 47.728 | 47.728 | 32.145 |
| hhc-2 | 1069.77 | 427.75 | 232.77 | 104.08 |
| hsa-1 | 1207.92 | 75.354 | 40.354 | 40.354 |
| hsa-2 | 315.95 | 137.05 | 42.689 | 9.804 |

*rnd*, hill climbing *hhc* and simulated annealing *hsa* implementations. The different hybrid configurations of *h1n*, *h2n* and *h3n* also demonstrate how the fine tune of the thresholds can largely influence the resulting quality of the results.

### 5.2.2 Data Correction

Further tests have been carried out aiming to demonstrate the applicability of the methodology in assisting expeditious modeling in sets where some part of the information data is corrupt or non existent.

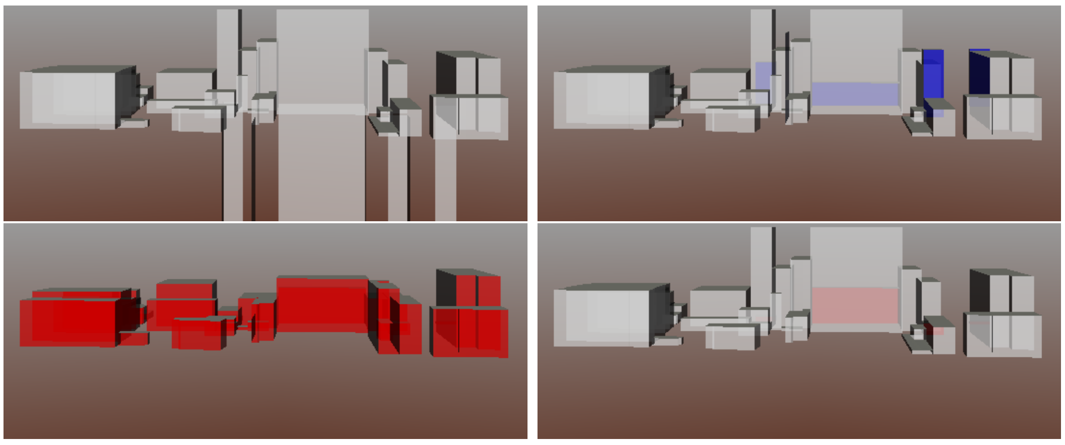
The procedural rule optimized in this specific test case was the same formula previously used, each parameter under the same constraint. However the tested configuration differ from the calibration tests previously described. The test was run with both, *toplist* and *originalparent*, family sizes of 20. Each test iterated 800 generations with a quality function evaluating the height differences of buildings of the correct height data.

The test determined the following optimum parameterization for the proposed procedural formula:

$$108.5729 + 0.9862 * (perimeter * 0.2155)$$

Figure 5.7 shows several screenshots of different modeling processes applied to the same flawed data source. The top left image shows the direct modeling using the original database values where corrupt data causes the visual glitches seen in the modeling. The top right image displays a crude approach at hiding this problem by replacing the corrupt height data with random values between the minimum and maximum height of buildings in the set. The buildings calculated with this method are rendered in blue. The buildings rendered in red are the ones that follow the optimized parameters determined by the optimizer. In the bottom left image the formula is applied to the entire dataset. In bottom right the formula is only applied to the buildings with known corrupt height values.

Figure 5.7: Screenshots of data flaw corrections rendered modeling.



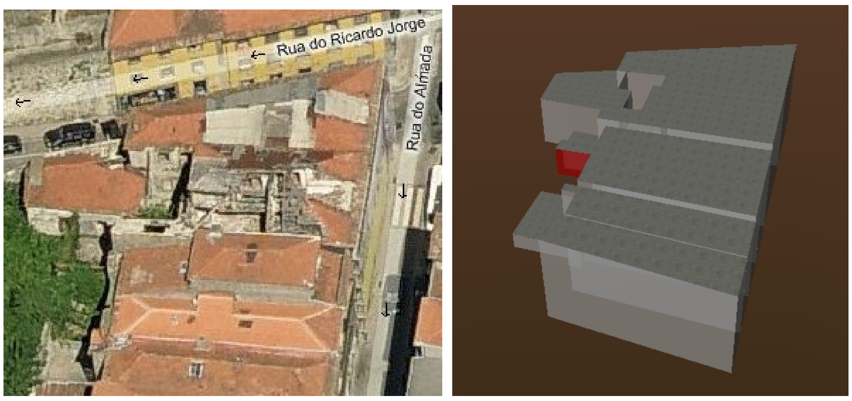
It is clear that the formula does not enable complete perfect matches, as can be concluded by comparing both images on the left, but it demonstrates that by using crude relational functions it is possible to determine important relations within the data set universe, enabling a more realistic replacement of missing, or corrupt, data from the set. So despite the calculation overhead, this methodology may prove useful for cases where domain knowledge of the procedural rules is vague and/or a trial and error methodology would be too time consuming.

### 5.2.3 Modeling Re-Application

Another test was conducted to analyse the re-application of the procedural rules in areas other than the one used for calibrating the parameters. This can only be applied to areas that share certain characteristics with the calibration area.

In this specific test a short calibration was executed on a very small area with data collected from the center of Porto. In Figure 5.8 we have on the left a satellite overview image of the actual area obtained from Microsoft Visual Earth[3] and on the right the rendered model screenshot used to calibrate the *topzvalue* function.

Figure 5.8: Comparative imagery of Rua do Almada real and calibration generated housing.



The calibration was carried out over a set of half a dozen typical buildings from a corner of *Rua do Almada* using a more constrained configuration than previous test cases. The scope of the parameter values was slightly reduced in an attempt to increase the ratio of convergence to the optimum, these reductions were applied considering the results of the previous tests. To speed up the calculation period the number of iteration steps was also reduced and divided into two phases. The first was used to determine the average height that best fits the data set. The second was used to determine the perimeter and area factor of influence.

With these tests the following formula was determined:

$$92.0360 + 0.1562 * (0.2661 * area)$$

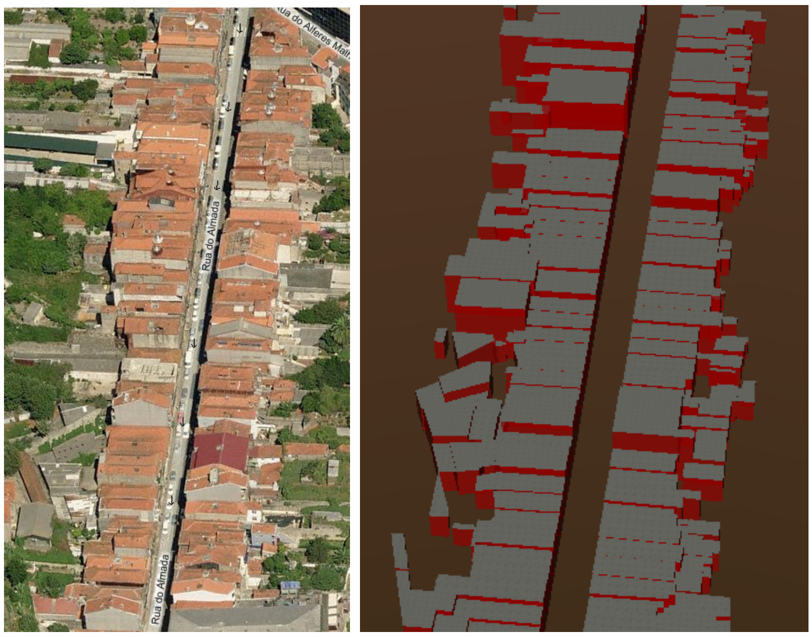
This formula implied that it could only be applied successfully to areas with a fixed *bottomzvalue*. However, the zone we wanted to apply this rule to estimate heights, the remaining full length of *Rua do Almada*, is set on a slope. To solve this issue without re-calculating any values the formula was adapted to base itself on an average height roughly estimated by analysing the data of the calibrating zone. This resolution approach has no scientific or empiric value, but is valuable in this context to demonstrate that the proposed methodology may not always work exactly as expected but still offer reliable information to generate realistic models without requiring a time expensive return to the calibration phase.

[3]http://maps.live.com

In Figure 5.9 we have a different section of *Rua do Almada*, with housing that resembles the characteristics of the small calibration zone. On the left we have the Microsoft Visual Earth overview, on the right the generated model, all housing generated using the following procedural height formula:

$$(bottomzval + 5.5) + (0.1562 * (0.2661 * area)$$

Figure 5.9: Comparative imagery of Rua do Almada real and modeling generated housing.



Comparing these two images helps us conclude that the proposed methodology can greatly assist expeditious modeling in generating realistic results from limited information and without requiring significant domain knowledge from the user.

## 5.3 Summary

This chapter of the thesis presented some results of practical applications of the developed algorithm. First by comparing the algorithm performance with several configurations under different types of problems, benchmarking against other standard meta heuristic algorithms applied to combinatorial search of optimum solutions. Secondly by applying the algorithm to the XL3D expeditious modeling system to determine missing height values from geographically referenced data.

The first section pertaining to the benchmarking reveals the development of a reliable new hybrid meta heuristic algorithm that can be configured to offer competitive results against other standard meta heuristic algorithms documented in the literature under a set of different problems. Despite performance being directly correlated with the specific problem nature, we have successfully shown that the developed algorithm can offer good results with good confidence in finding the optimum, as long as it is allowed to process enough iteration steps, faced with different sets of problems. The configurable nature of our algorithm allows the user to change the thresholds and family sizes depending on the number and correlation of the input parameters of the combinatorial problem. It is important to keep in mind that the problem of fine tuning the configuration of the new algorithm to best fit each problem goes beyond the scope of this thesis and was only briefly addressed[4] when presenting these results. The main aim was the development of a competitive new hybrid meta heuristic algorithm capable of handling all three parameter input types (boolean, integer and float), where most meta heuristic algorithms can only handle discrete representations.

The second section demonstrated how the algorithm can easily be applied to solve basic expeditious modeling problems of optimization and data mining. Applying the new hybrid meta heuristic algorithm to unravel optimum modeling parameters it becomes possible to obtain realistic data from sampled quantities of geographically referenced information. Several test cases were presented that demonstrate a more thorough calibration of the parameters and simple application cases of data correction and even generation. The methodology waves some requirements of complete and correct geographically referenced data to be available, and avoids most of the manual parameter calibration that is typically required for generating a realistic urban virtual environment.

[4]under the scope of hyper heuristics

# Chapter 6

# Conclusions

This sixth chapter presents the conclusions of this thesis. In the first sub section a short summary of what was achieved is presented, explaining in what way the development of this thesis contributed to the state of the art. The second sub section presents a short selection of ideas envisioned for further research and development.

## 6.1 Developed Work

The use and importance of expeditious modeling of 3D virtual urban environments has increased significantly in the past years. Still many difficulties are associated with the technology. Generating appropriate production rules with fine tuned parameters, or the resources required in collecting georeferenced data, for example, can easily bottleneck the potential of the modeling system in achieving realistic results.

Different methods can be applied to determine adequate production rules for expeditious modeling systems, ranging from simple empirical definitions to observing 3D modelers creating objects. Fine tuning the parameters of these modeling rules is a problem that requires knowledge from the underlying mathematics behind the rules. Attempting to fine tune these rules without domain knowledge can easily lead long sessions of trial and error.

Applying certain fields of artificial intelligence, such as optimization algorithms and knowledge data mining may increase the level of automation of the expeditious modeling system. An hybrid meta heuristic optimizer system was successfully developed, implemented and tested. The optimizer is easily adaptable and reconfigurable, aimed to solve problems where the number of input parameters of discrete and continuous nature can vary drastically and the domain knowledge of the problem may not be accessible. Global optimization of parameters for these kind of problems is a common requirement of expeditious modeling systems in general and expeditious urban modeling systems in particular.

The developed optimizer is an hybrid version of real coded genetic algorithms with traits of simulated annealing and harmony search. The algorithm was developed aiming to solve global optimization problems that have no access to domain knowledge information and require support for boolean, integer and float type of input parameters. The algorithm was expected to perform competitively with other standard implementations of similar algorithms.

The performance of the algorithm was tested under different configurations, some of which emulating the typical behavior of popular meta-heuristic implementations, documented in the literature. The statistical comparison between the different configurations revealed competitive results that offer good confidence in future applicability in other areas.

The optimizer algorithm was also applied to an expeditious modeling system under a test case scenario to determine the optimum parameter combination of certain productions rules by comparing modeling results with real data. This knowledge was applied to successfully generate realistic urban models in other locations sharing similar characteristics that missed crucial data.

These results show that the proposed system is a viable solution for several types of expeditious modeling situations, warranting further development and increased interoperability with similar modeling and simulation systems.

## 6.2 Future Work

There are many possible applications of the proposed system. Its reconfigurability and architecture were developed having in mind the multiplicity of global optimization problems that abound in expeditious modeling systems. As such, it would be interesting to observe the performance of the system operating under more complex test cases than the ones documented on this thesis, not only to strain test the performance of the system but also to inspire further development and optimization of the system.

Future developments in the optimizer algorithm include the implementation of a more comprehensive quantity of crossbreed and mutation operands with dynamic adaptation through the iterative process. Also possible is a simplified integration of domain knowledge to improve performance in problems where the domain knowledge can greatly assist in reducing the search scope. Another possible implementation that could improve performance is using gradient based parameter correlation estimations and applying principles of ant colony, nested partitions, particle swarm and parallel processing to the developed system. Especially important is the optimization of the algorithm to better handle multi objective and constrains from standard global optimization problems.

Another interesting approach could explore the hyper heuristic characteristic of the developed optimizer. It is already possible to recursively apply the algorithm to unravel the optimum configuration to solve a given type of problems, these configurations could be clustered into multiple categories considering general characteristics. The clustering and classification would enable an automatic system to determine, with certain degrees of probability, which range of configurations are better suited to solve certain types of global optimization problems. This idea bridges the areas of knowledge data mining and data extraction and could have many interesting applications in different areas.

Another idea is to expand the system to a certain type of interactive feedback methodology where the system would not determine the quality fitness value of each solution automatically. Instead the system would allow for an interactive system of sorting and prefered selection. This methodology would allow a certain creative freedom that could have several interesting applications. One of the more obvious ones would be assisting to define constraints for parameters of undeterminable quality fitness functions. Many problems lack explicit fitness functions to measure the quality of their possible solutions. This could be due to a lack of knowledge information on the production rules, or a resource incapacity to measure comparison values. This methodology would enable to avoid defining a quality function. Another possible application of the methodology would be exploring the full capacity of complex production rules. Scenarios in expeditious modeling are often subjected to user approval, it can occur that a user knows they want something to look different but don't quite know what exactly should be different and what should remain the same. An interface that allows the user to interact, approving or disaproving the degree of an alteration, can enable a system to probabilistically determine the key parameters that the user actually wanted to adjust in the modeling, and a possible relation between them.

In conclusion, there are many interesting ideas that can be explored for future work in this area. The application of artificial intelligence in general and parameter optimization in particular in the field of expeditious modeling is still very recent and has many interesting applications for both academic researchers and capital investors from the industry.

# References

[Apt03] K. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[ARP05] M. Arumugama, M. Raoa, and R. Palaniappanb. New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems. In *Applied Software Computing 6*, pages 38–52, 2005.

[Bac96] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, Oxford, UK, 1996.

[Bro70] C. Broyden. The convergence of a class of double-rank minimization algorithms. In *Journal of the Institute of Mathematics and Its Applications*, volume 6, pages 76–90, 1970.

[CBSF07] A. Coelho, M. Bessa, A. Sousa, and F. Ferreira. Expeditious modeling of virtual urban environments with geospatial l-systems. In *Computer Graphics Forum, number 4*, volume 26, pages 769–782, 2007.

[CHS$^+$97] P. Cabena, P. Hadjnian, R. Stadler, J. Verhees, and A. Zanasi. *Discovering Data Mining: From Concept to Implementation.* Prentice Hall, 1997.

[Cle06] M. Clerc. *Particle Swarm Optimization.* ISTE, 2006.

[Col68] A. Colville. A comparative study of nonlinear programming. In *Technology Report number 320-2949.* IBM New York Scientific Center, 1968.

[Dav01] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 2001.

[Deb96] Paul Debevec. *Modeling and Rendering Architecture from Photographs.* PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

[Dec03] R. Dechter. *Constraint Processing.* Morgan Kaufmann Publishers Inc., 2003.

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano, Italy, 1992.

[DP05] D. Du and P. Pardalos. *Handbook of Combinatorial Optimization*, volume 3. Kluwer Academic Publishers, 2005.

[DT07a] K. Deep and M. Thakur. A new crossover operator for real coded genetic algorithms. In *Applied Mathematics and Computation*, volume 188, pages 895–911, May 2007.

[DT07b] K. Deep and M. Thakur. A new mutation operator for real coded genetic algorithms. In *Applied Mathematics and Computation*, volume 193, pages 211–230, October 2007.

[Fin08] D. Finkenzeller. Detailed building facades. In *IEEE Computer Graphics and Applications*, pages 58–66. IEEE Computer Society, 2008.

[Fle70] R. Fletcher. A new approach to variable metric algorithms. In *Computer Journal*, volume 13, pages 317–322, 1970.

[FLR$^+$95] O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3-d reconstruction of urban scenes from sequences of images. In *Rapport de Recherche 2572*. INRIA, 1995.

[Gen03] M. Gendreau. An introduction to tabu search. In *Handbook of Metaheuristics. Series: International Series in Operations Research and Management Science, Chapter 2*, pages 37–54. Kluwer Academic Publishers, 2003.

[GK03] F. Glover and G. Kochenberger. *Handbook of Metaheuristics.* Kluwer Academic Publishers, 2003.

[GKL01] Z. Geem, J. Kim, and G. Loganathan. A new heuristic optimization algorithm: Harmony search. In *Simulation 2*, volume 76, pages 60–68, 2001.

[GL02] F. Glover and M. Laguna. Tabu search. In *Handbook of Applied Optimization*, pages 194–208. Oxford University Press, 2002.

[GLS07] A. Grosso, M. Locatelli, and F. Schoen. A population-based approach for hard global optimization problems based on dissimilarity measures. In *Mathematical Programming, number 2*, volume 110, pages 373–404, Secaucus, NJ, USA, 2007. Springer-Verlag New York, Inc.

[Gol70] D. Goldfarb. A family of variable metric updates derived by variational means. In *Mathematics of Computation*, volume 24, pages 23–26, 1970.

[Gol02] D.E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Addison-Wesley, Reading, MA, 2002.

[GP71] A. Goldstein and F. Price. On descent from local minima. In *Mathematics of Computation 25*, pages 569–574, 1971.

[GW98] A. Gruen and X. Wang. Cc-modeler: a topology generator for 3-d city models. In *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 286–295, October 1998.

[Han06] N. Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[Hay99] S. Haykin. *Neural Networks*. Englewood Cliffs: Prentice Hall, 1999.

[HH06] R. He and S. Hwang. Damage detection by an adaptive real-parameter simulated annealing genetic algorithm. In *Computation Structures, number 31-32*, volume 84, pages 2231–2243, Elmsford, NY, USA, 2006. Pergamon Press, Inc.

[Hol92] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[HTdW95] A. Hertz, E. Taillard, and D. de Werra. A tutorial on tabu search. In *In Proceedings of Giornate di Lavoro AIRO95*, 1995.

[IHR07] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. In *Evolutionary Computation, number 1*, volume 15, pages 1–28. MIT Press, 2007.

[JY04] S. Jacobson and E. Yucesan. Analyzing the performance of generalized hill climbing algorithms. In *Journal of Heuristics, number 4*, volume 10, pages 387–405, Hingham, MA, USA, 2004. Kluwer Academic Publishers.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, 1995.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimizing by simulated annealing. In *Science 4598*, volume 220, 1983.

[LG05] K. Lee and Z. Geem. A new meta-heuristic algorithm for continuous engineering optimization harmony search theory and practice. In *Computational Methods Applied Mechanical Engineering*, volume 194, pages 3902–3933, 2005.

[Lin68] A. Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. In *Journal of Theoretical Biology*, volume 18, pages 280–315, 1968.

[LWW08] M. Lipp, P. Wonka, and M. Wimmer. Interactive visual editing of grammars for procedural architecture. In *Proceedings of ACM SIGGRAPH 2008 / ACM Transactions on Graphics*, volume 27. ACM Press, 2008.

[Mae95] P. Maes. Artificial life meets entertainment: Life like autonomous agents. In *Communications of the ACM, number 11*, volume 38, pages 108–114, 1995.

[Mar04] T. Marino. *3D Game-Based Filmmaking: the Art of Machinima.* City: Paraglyph Publishing, 2004.

[MFD07] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. In *Applied Mathematics and Computation 188*, pages 1567–1579, 2007.

[Mic94] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1994.

[MMRS55] J. McCarthy, M. Minsky, N. Rochester, and C. Shannon. A proposal for the dartmouth summer research project on artificial intelligence. 1955.

[Mos89] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In *Caltech Concurrent Computation Program, Report C3P-778*, 1989.

[MPT+05] N. Mishra, M. Prakash, R. Tiwari, F. Shankar, and T. Chan. Hybrid tabusimulated annealing based approach to solve multi-constraint product mix decision problem. In *Expert Systems with Applications 29*, 2005.

[MU49] N. Metropolis and S. Ulam. The monte carlo method. In *Journal of the American Statistical Association, number 247*, volume 44, pages 335–341, 1949.

[MWH+06] P. Muller, P. Wonka, S. Haegler, A. Ulmer, and L. Gool. Procedural modeling of buildings. In *Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics*, volume 25, pages 614–623. ACM Press, 2006.

[PM01] Y. Parish and P. Muller. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, pages 301–308. ACM Press, 2001.

[PPK+98] M. Pollefeys, M. Proesmans, R. Koch, M. Vergauwen, and L. Gool. Acquisition of detailed models for virtual reality. In *Virtual Reality in Archaeology*, volume 843, pages 71–77. BAR International Series, CAA, Archaeopress, publishers of British Archaeological Reports, 1998.

[PR02] P. Pardalos and M. Resende. *Handbook of Applied Optimization.* Oxford University Press, 2002.

[Pru86] P. Prusinkiewicz. Graphical applications of l-systems. In *Proceedings of ACM SIGGRAPH 1986*, pages 247–253. ACM Press, 1986.

[Rec71] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* PhD thesis, Technical University of Berlin, 1971.

[Rei03] L. Reis. *Coordination in Multi-Agent Systems: Applications in University Management and Robotic Soccer.* PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2003.

[Rey99] R. Reynolds. Cultural algorithms: theory and applications. In *New ideas in optimization*, pages 367–378. McGraw-Hill Ltd., UK, 1999.

[RK90] E. Rich and K. Knight. *Artificial Intelligence.* McGraw-Hill Higher Education, 1990.

[RK04] R. Rubinstein and D. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics).* Springer-Verlag New York, Inc., 2004.

[RN03] S. Russel and P. Norvig. *Artificial Intelligence: A modern approach.* Prentice-Hall, 2nd edition, 2003.

[Ros60] H. Rosenbruck. *An Automatic Method for Finding the Greatest or Least Value of a Function.* British Computer Society, 3rd edition, 1960.

[SC03] A. Schilling and V. Coors. Generation of vrml city models for focus based tour animations. In *Proceeding of the Eighth International Conference on 3D Web Technology, Web3D 2003, Saint Malo, France*, pages 39–48, 2003.

[Sha70] D. Shanno. Conditioning of quasi-newton methods for function minimization. In *Mathematics of Computation*, volume 24, pages 647–656, 1970.

[SHL$^+$05] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapure, 2005.

[SO00] L. Shi and S. Olafsson. Nested partitions method for global optimization. In *Operations Research*, volume 48, pages 390–407, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, 2000. INFORMS.

[SO09] L. Shi and S. Olafsson. *Nested Partitions Method, Theory and Applications*, volume 109 of *International Series in Operations Research and Management Science.* Springer, 2009.

[SRDT01] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. In *IEEE Computer Graphics and Applications, number 3*, volume 21, pages 53–61, 2001.

[SSA07] A. Saxena, J. Schulte, and Ng A. Depth estimation using monocular and stereo cues. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[TSS04] Lassi Tasajarvi, Bent Stamnes, and Mikael Schustin. *Demoscene: the Art of Real-Time.* Even Lake Studios, 2004.

[Wei99] G. Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT Press, Cambridge, MA, USA, 1999.

[WMW$^+$08] B. Watson, P. Muller, P. Wonka, C. Sexton, O. Veryovka, and A. Fuller. Procedural urban modeling in practice. In *IEEE Computer Graphics and Applications*, pages 18–25. IEEE Computer Society, 2008.

[WWR05] Z. Wang, Y. Wong, and M. Rahman. Development of a parallel optimization method based on genetic simulated annealing algorithm. In *Parallel Computation, number 8+9*, volume 31, pages 839–857, Amsterdam, The Netherlands, 2005. Elsevier Science Publishers B. V.

[WWSR03] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. In *Proceedings of ACM SIGGRAPH 2003 / ACM Transactions on Graphics*, volume 22, pages 669–677. ACM Press, July 2003.

[Yan05] S. Yang. Population-based incremental learning with memory scheme for changing environments. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 711–718, New York, NY, USA, 2005. ACM Press.

[YSA01] H. Youssef, S. Sait, and H. Adiche. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. In *Engineering Applications of Artificial Intelligence 14*, 2001.

[ZKGGK06] L. Zebedin, A. Klaus, B. Gruber-Geymayer, and K. Karner. Towards 3d map generation from digital aerial images. In *International Journal of Photogrammetry and Remote Sensing*, volume 60, pages 413–427, September 2006.