Labelling Maps using Multi-Objective Evolutionary Algorithms

Lucas P. Bradstreet
Abstract

Map labelling is the problem of arranging place names on maps such that labels do not overlap and are clear to a reader. Determining the optimal label arrangement is combinatorially difficult and has so far defied total automation. As such, map making remains a costly process. Today’s map labelling techniques generally consider only the number of overlaps during optimisation and while they do succeed at minimising this objective, there are other objectives that contribute to the overall quality of a map. We will discuss a multi-objective evolutionary algorithm designed to automate map labelling when assessing overall map quality by several criteria such as label overlaps, clarity, font size and aesthetics.

This process allows a user to select their optimal compromise between each of the quality criteria. As the best compromise between these objectives is highly subjective and dependant on the map being labelled, our algorithm is superior to others in that it simultaneously finds a set of trade-off solutions rather than applying a set evaluation function that predefines the way that these quality constraints should be optimised.

We have demonstrated several example real world maps to prove that this technique is able to produce maps of higher quality than competing techniques. We have also selected several final labelled maps which illuminate the compromises between quality criteria.

Keywords: map labelling, multi-objective evolutionary algorithms, genetic algorithms

CR Categories: I.2.1 [Cartography], 1.2.1 [Artificial Intelligence]: Applications and Expert Systems, 1.3.5 [Computer Graphics]: Computational Geometry and Object Modelling
Acknowledgements

I would like to begin by thanking my family for the constant support they provided even when I was incommunicado for weeks at a time. Thanks for making sure I was never computer-less and for helping me buy programming books in younger years which nurtured my love of computer science.

Furthermore, I would like to thank Rami for his persistent friendship, Rasha for being a fantastic house mate and friend and never complaining when I was thoughtless and for sharing times of honours frustration, Lara for being the best listener and selfless person I know, and the Honours crew for the good times in the labs. Bobby deserves my thanks for always making sure the honours fridge was filled with diabetes inducing goodness. Bondy and Quenten were always fun to talk to about computer science and other geeky matters . . . . Furthermore, I would like to thank Bondy for his competitive nature which always spurred me to ”do better than Bondy”.

Finally, my supervisors Dr Luigi Barone and Dr Lyndon While have been more patient with me than my feeble writing skills can describe. They always guided me in the right direction, and their ”Good Cop, Bad Cop” routine was a great source of entertainment and was fairly effective at making me get things done. Luigi deserves extra credit for helping me with my thesis above and beyond the call of duty.

. . . and the rest of the dominoes will fall like a house of cards. Checkmate. — Zapp Brannigan, DOOP General.
## Contents

Abstract ii  
Acknowledgements iii  

1 Introduction 1  

2 Literature Review 3  
2.1 Evolutionary Algorithms 3  
2.2 Multi-objective EAs 4  
2.3 The Map Labelling Problem 6  
2.3.1 Introduction 6  
2.3.2 Map Labelling and Cartography papers 7  
2.3.3 Representation 7  
2.3.4 Simulated Annealing Approaches 8  
2.3.5 Genetic Algorithm Approaches 10  
2.3.6 Problems with SA and GA map labellers 11  
2.3.7 Evolutionary Algorithm Approaches 12  
2.3.8 User Hints during Optimisation 14  

3 Materials and Methods 16  
3.1 Modelling the Problem 16  
3.1.1 Representation 16  
3.1.2 Objectives 19  
3.2 Algorithms and Implementation 22  
3.2.1 Representation 22  
3.2.2 MOEA to solve the MLP 23  
3.3 Memetic Algorithm 25
4 Experimental Results

4.1 Environment ................................................. 26
4.2 Hypervolume ............................................... 26
4.3 Comparing running time by fitness evaluations ................. 27
4.4 Experiment One: Determining the Population Size .......... 27
4.5 Experiment Two: Determining the Recombination Probability .. 29
4.6 Convergence using optimal parameters for Guam ............ 30
  4.6.1 Results ................................................... 30
4.7 Memetic Algorithm Experiment ............................... 37
  4.7.1 Visual Experiment: A comparison between Preuß and our
       algorithms ................................................ 37
  4.7.2 Examples of final maps exhibiting objective compromises . 39

5 Conclusion .................................................. 44

A Original Honours Proposal .................................. 46
List of Figures

2.1 Position preferences for labels under a discrete label position system 8

3.1 An illegal label position possible under conventional polar coordinates ........................................ 17
3.2 Polar coordinate system vs Modified polar coordinate system ...................................................... 17
3.3 Minimum attachment distance for labels under the modified polar coordinate system .................. 18
3.4 Example of a label to label conflict .................................................. 19
3.5 Example of a label clarity problem .................................................. 20

4.1 Hypervolume achieved for varying population sizes. .................. 28
4.2 Hypervolume achieved for varying recombination probabilities. .. 29
4.3 Convergence of Conflicts vs Clarity .................................................. 31
4.4 Convergence of Conflicts vs Aesthetics .................................................. 32
4.5 Convergence of Clarity vs Aesthetics .................................................. 33
4.6 Convergence of Clarity vs Font Size .................................................. 34
4.7 Convergence of Conflicts vs Font Size .................................................. 35
4.8 Convergence of Font Size vs Aesthetics .................................................. 36
4.9 Hypervolume when memetic algorithm is used at various intervals. 38
4.10 Map of Guam labelled with MOEA .................................................. 39
4.11 Map Labelled with Preuß’s algorithm .................................................. 40
4.12 Map of Northern Mariana Island labelled with MOEA exhibiting high clarity .................................................. 41
4.13 Map of Northern Mariana Island labelled with MOEA exhibiting high aesthetic qualities. .................................................. 42
4.14 Map of Delaware labelled with MOEA .................................................. 43
CHAPTER 1

Introduction

Most people have relied on a map at one time or another. Whether we use them to navigate or to gain a better understanding of the world around us, cartography has helped us in innumerable ways. Without maps, navigation would prove incredibly difficult.

In an age when computers are able to automate most routine tasks, map making has so far defied total automation. One would think that it would be as simple as feeding a computer a list of coordinates and their corresponding names and then leaving a computer to do the rest. Unfortunately the process is not that simple and is a complex combinatorial problem. The problem encountered when attempting to fully automate map production is known as the Map Labelling Problem (MLP).

The MLP refers to the difficulty encountered when placing textual labels for features or sites on a map, such that the labels do not overlap and are legible to a reader. The Map Labelling Bibliography [1] alone lists 292 papers attempting to address issues posed by the map labelling problem.

Solutions to the map labelling problem also lend themselves to use in several areas outside of cartography. These include the following:

- Drafting and architecture – technical drawings including labels containing names, measurements and other values.
- Graph diagrams – the labelling of graph diagrams such as visualisation of networks, UML diagrams, state diagrams and others.
- Medical image analysis – the inclusion and automation of labelling anatomical and biological features in medical imagery.

Although the task of labelling a map may seem trivial, this is hardly the case. In most real-world cases map labelling is performed by hand. This is done from scratch or by using an algorithm to get an initial sub-optimal solution that is
then improved upon by a cartographer. The human brain is better suited to fix aesthetic issues and detect ambiguities in labels than any algorithm available today. As a consequence, the production of maps remains a costly and time consuming affair. Cook and Jones [6] report that labelling a map accounts for up to 50% of overall map production costs.

Throughout this work we will address many of the issues posed by the MLP such as label overlaps, ambiguity and aesthetic problems. Several existing techniques for solving the map labelling problem will be described along with any shortcomings they exhibit. Most existing algorithms do not consider important factors that contribute to the quality of produced maps. These factors include measures of how legible or ambiguous labels are, font sizes for labelled maps, and aesthetic qualities.

These shortcomings limit the usefulness of map labelling algorithms for real world map production. Though minimising the number of label overlaps is a primary concern of labelling algorithms, there is more to producing high quality maps than this measure. Current algorithms attempting to incorporate other quality objectives to the creation of maps, treat these objectives as secondary objectives optimising for the number of conflicts primarily.

As map production is a process that usually requires a human expert to place labels to maximise their usability, these ‘secondary’ objectives are not given high priority in map labelling algorithms. The production of maps is a subjective process for which the requirements are often not known ahead of time. For example, the font size for a map is usually not known ahead of time and may require several attempts at labelling before the appropriate font size is discovered. Additionally, a labelling algorithm usually does not know the aesthetic qualities or clarity constraints required to make a map usable ahead of the labelling process. Current labelling algorithms simply do not consider these subjective qualities and the appropriate compromises required to create a production quality map.

In Chapter 2 we introduce the map labelling problem and the various problems it poses to the production of maps. We introduce several algorithms intended to solve a restricted model of the problem. In Chapter 3 we introduce our Multi-objective Evolutionary Algorithm (MOEA) used to create a set of map labellings, each of which are better than each other for one or more of the quality objectives. This process is superior to other algorithms in that it allows a user to select the optimal labelling for their subjective quality compromises. In Chapter 4 we discuss experiments that are intended to show the operation of our multi-objective map labeller. This process includes the determination of parameters for our evolutionary algorithm that give good results for the tested maps. As map labelling is a subjective process, several example maps are included.
CHAPTER 2

Literature Review

2.1 Evolutionary Algorithms

*Evolutionary Algorithms* (EAs) are techniques used for the optimisation of complex problems. Darwin’s theory of evolution is the inspiration for such a technique \[7\]. Just as Darwin’s model of evolution applies to natural processes, evolutionary algorithms analogously apply to optimisation problems.

In EAs, a population of solutions (individuals) for an optimisation problem is created, each with its own set of parameters known as its genotype (usually represented by a string). At each stage of the algorithm, individuals are perturbed and then judged via a *fitness function* that numerically evaluates them. The weaker solutions from this population are rejected while better solutions are allowed to survive and contribute to the next generation of solutions. This process results in a *selection pressure* that pushes the population towards better solutions as judged by the fitness function.

As in evolution, two techniques are used together to create new solutions from parent solutions: *recombination* and *mutation*. Recombination combines portions of the genotype from two or more parent solutions to create a child that shares traits with each of its parents. Recombination encourages sharing of beneficial traits with other individuals in order to reduce the amount of work required to find good solutions. Mutation allows the creation of new attributes by randomly modifying a solution’s genotype so that new regions of the search space can be explored.

The procedure for a generic EA is as follows:

1. Initialise the solution population.

2. Evaluate the population using the fitness function for the problem.

3. Generate an offspring population from the parent population using mutation and recombination operators.
4. Evaluate the offspring population using the fitness function.

5. Combine the offspring and parent population into a single population and sort solutions by their fitness.

6. Select a new parent population from the better solutions from the population sorted in 5. Relatively worse solutions are allowed to die off to make room for the new child solutions.

7. Repeat steps 3–6 until convergence criteria have been satisfied.

EAs work well when applied to hard problems such as those with large search spaces and non-intuitive solutions. They have been shown to yield excellent results for a wide range of problems including combinatorial optimisation processes such as the travelling salesman problem [10], chip design [25], geometric shape design [2] and network layout problems [3].

2.2 Multi-objective EAs

Multi-objective EAs (MOEAs) are used to optimise a solution with multiple objectives that cannot be clearly ordered by importance. Rather than have a single solution with a ‘best’ fitness value for a population, MOEAs generate a front of diverse solutions, each of which is better than another in one or more objectives. A front of optimal solutions in which an improvement in one objective necessarily worsens at least one of the other objectives is called the pareto-optimal front.

An example of an application where MOEAs excel can be found in Barone et al [2] with their multi-objective EA for the evolution of rock crushers. Rock crushers are used to extract raw materials from rocks and are subject to various tunable parameters. Barone et al.’s MOEA attempts to simultaneously optimise two objectives: maximising the capacity of the rock crushers, and minimising the size of the crushed product. Each of these objectives is worthwhile in its own right and cannot be clearly ordered. As the optimisation of each of these objectives may require a compromise in the other objective, a set of trade-off solutions must be maintained. Each of these solutions are not clearly better than another, and may each be useful under certain conditions.

A key concept in the operation of MOEAs is that of domination. In an MOEA, an individual is said to dominate another solution if it is at least as good in all objectives and better in one or more objectives. In the case of the rock crusher
MOEA, if an individual has a larger capacity and creates a smaller product than another individual, then it dominates the other solution.

Using this definition of domination, we are able to introduce a method for determining the approximation of the pareto-optimal front. A front represents a set of solutions, each of which is a trade-off in one or more objectives when compared to any other solution in its front. Under the definition of non-dominated ranking, no solution in a front is dominated by any other solution in that front. Each front is assigned a rank, which denotes that front’s level of non-domination. For example, the front with rank 0, otherwise known as the pareto-optimal front, contains individuals which are not dominated by solutions in any other rank. It is these ranks that are used by the EA to sort candidate solutions in step 5 of the algorithm described earlier.

One specific example of a MOEA is NGSA-II by Deb et al. [8]. NGSA-II operates in a similar way to the generic EA described above, with the exception of step five. Rather than assessing individuals purely on one fitness value in this operation, they are instead ranked by sorting into non-dominated fronts. It combines this ranking function with a crowdedness measure, a measure used to choose equally ranked solutions that maximise the coverage of the objective space. This crowdedness measure is used on fronts which have grown too large to be fully included in a new population.

In the case where a front is too large to fit into the surviving population, it is necessary to determine which solutions from that front should be allowed to survive. As the aim of an MOEA is to allow diverse solutions to be found, any technique aimed at discriminating between equally ranked solutions should select solutions which provide the best coverage of the solution space. One technique for achieving this is the Crowded-Comparison Operator used in NGSA-II [8]. This technique uses a distance measure which calculates the ‘distance’ between solutions over multi-dimensional space. By choosing solutions from the pareto front which have the largest distance, a diverse spread of solutions is ensured.
The *Crowded-Comparison Operator* technique operates as follows:

1. Select the front that requires pruning and set a distance measure for all solutions to zero.
2. Sort individuals by objective $m$.
3. Select boundary individuals with best and worst scores for $m$.
4. Increase distance measure for individual $i$ by the difference between individuals $i - 1$ and $i + 1$ for objective $m$, normalised for the range of values for objective $m$:
   \[
   \frac{\text{Ind}_{m}^{i+1} - \text{Ind}_{m}^{i-1}}{\text{maxfitness}_m - \text{minfitness}_m}
   \]
5. Repeat step 4 for each non-boundary individual $i$.
6. Repeat steps 2 – 5 for each objective $m$.
7. Sort all individuals by overall distance values and select those with the greatest distance.

### 2.3 The Map Labelling Problem

#### 2.3.1 Introduction

The Map Labelling Problem (MLP) has long been a problem for researchers attempting to automate map production. Currently, as reported by Cook and Jones [6], labelling a map accounts for up to 50% of overall map production costs. Imagine a map with a number of sites (or features) on it. Now consider labelling each of these sites with a textual label containing the name of each site. In order for the map to be read easily, the relationship between each label and its associated site should be unambiguous, the label should be easily discernible from other labels and the map should be aesthetically pleasing. Using a naive heuristic (approximation) to label sites tends to lead to label overlaps and maps that are difficult to read.

The MLP is difficult to solve, as finding the optimal combination of label positions for real world maps is infeasible using brute force techniques. Wolff [24] proves that the Map Labelling problem is NP-Hard to even approximate the solution with quality guarantee better than 50%. Work to computationally solve this problem has thus revolved around approximating the best labelling.
2.3.2 Map Labelling and Cartography papers

A seminal paper by Imhof [13] outlined several requirements for good map labelling. These have formed the basis for all further work in the area.

These requirements are:

- **Legible** — labels must have legible font sizes and be positioned in such a way that they are easily read.

- **Unambiguous** — each label must clearly identify a single site and not be confused with another label or site.

- **Overlap avoidance** — a label should not overlap any other label or site.

- **Aesthetics** — labels should not be overly clustered or distract from map features.

Cartographers must attempt to satisfy these subjective requirements to achieve high quality solutions. Accordingly, all attempts at automated map labellers must choose a combination of these requirements to form their optimisation criteria.

Previous approaches tackle this optimisation process using a variety of methods. Raidl [17] uses Genetic Algorithms (GAs, similar to EAs) combined with heuristic techniques to find good solutions. He finds that this gives better results compared to another approach by Christensen et al. [4] that uses simulated annealing as its optimisation technique. A similar genetic algorithm is proposed by Dijk et al. [22]. Preuß [16] demonstrates a working Evolutionary Strategy (ES) for solving the map labelling problem and presents a comparison with other techniques. Nascimento and Eades [9] view user interaction as essential to the map labelling process. They present a framework to allow suggestions provided by a human operator to guide the search to better solutions. These are discussed in the remainder of the chapter.

2.3.3 Representation

Most previous attempts to solve the MLP problem use a model based on Imhof’s [13] qualitative measures for rating a labelling solution. In order to model the MLP, we must first develop a representation for labellings and formulate a means of assessing the quality of these solutions.

We define a site $S_i$ as represented by a Cartesian coordinate $(x, y)$ corresponding to its position on a map. Although the position of the label relative to a site
could also be represented as a real valued \((x, y)\) position, most of the previous attempts to solve the MLP follow Imhof’s discrete representation of label positions. Imhof lists eight possible label positions and their comparative desirability (preference rating shown in Figure 2.1), which are used to rate a particular label position when measuring the quality of a labelling. Label preference values are weighted lower than conflict values. This is to ensure that a labelling algorithm will never choose a series of labels with preferential positions over the elimination of a label conflict.

![Figure 2.1: The possible positions a label may take around a given site. The numerical values correspond to the preference rating given by Imhof. The solutions are rated so that sites will have labels arranged in similar locations if possible which improves the readability of the map.](image)

2.3.4 Simulated Annealing Approaches

Simulated Annealing (SA), first discovered by Kirkpatrick et al. [14], is a technique often used for optimisation problems. During optimisation SA randomly perturbs its current solution. If the new solution is deemed worse by the function being minimised or maximised, it accepts the algorithm accepts new solution as the current solution with a probability. The acceptance of worse solutions is to allow the algorithm to escape local minima. To prevent the degradation of the algorithm to bad solutions later in the process, an SA algorithm includes a temperature variable that represents the probability that a worse solution will be accepted as the next solution. This temperature variable is slowly reduced until the probability that a worse solution is accepted becomes relatively small and the algorithm converges to an answer.

Map Labelling SA by Christensen et al. Christensen et al. [4] use SA as their optimisation technique for solving the MLP. Their algorithm quickly
achieves decent labellings in most cases when attempting to minimise label conflicts.

In their approach, a scoring function evaluates a map labelling solution to allow comparisons between different solutions. Christensen et al. formulate this value by adding the number of label overlaps to a rating for each label, based on the position of the label given by:

\[
f(M_p) = \sum_{i=1}^{n} \left( c_i + \frac{(r_i - 1)}{N} \right), \quad c_i \in \{0, 1\}, \quad r_i \in \{1...N\}
\]

where \( M_p \) is map labelling \( p \), \( c_i \) is a binary value equalling one when a conflict occurs for label \( i \) and zero otherwise, \( N \) is the range of label preferences (in this case 8 different preferences are used), and \( r_i \) is the preference value for that labelling, as shown in Figure 2.1.

The optimisation of this scoring function usually leads to high quality map labellings compared to heuristic and gradient descent methods. Generally, the algorithm quickly yields a good result and is one of the first techniques that produced decent results when minimising for label conflicts on large maps. However, there is more involved in labelling maps than minimising label conflicts and improving label preferences. As Imhof [13] discusses label ambiguity should also be minimised and aesthetic qualities of the map should be improved as much as possible. Christensen et al. do not attempt to optimise these qualities in the map and only improve the label conflicts. This leaves much to be desired when applying their algorithm to real world map labelling tasks. We will be attempting to improve on this technique by simultaneously optimising all of Imhof’s quality objectives and should therefore be able to achieve better results than this labelling technique.
2.3.5 Genetic Algorithm Approaches

GAs are often used to find approximate solutions to NP-hard problems. They were introduced by Holland [12] in 1975 and have lately been applied to the MLP. GAs for the MLP are generally able to achieve slightly better results than SAs at the price of computation time. The first GAs for labelling maps were introduced by Raidl [17] and Dijk et al. [22].

Raidl’s Genetic Algorithm for Map Labelling  Günter Raidl [17] uses a GA for global optimisation of map labellings by minimising label to label overlaps. Raidl uses a standard GA combined with heuristic population initialisation as well as preprocessing in order to find good labelling solutions in a reasonable amount of time.

In Raidl’s approach, a conflict table is generated at the initialisation stage of the algorithm. A conflict level $c_{i,j}$ and a conflict references array $P_{i,j}(i = 1..n, j = 1..n)$ make up this table. The conflict level $c_{i,j}$ is the number of label positions a site $i$, with label in position $j$, overlaps. If the label overlaps a point site itself, $c_{i,j}$ is set to $\infty$ as the position is impossible. The algorithm can now safely prune the search space using this table as many label positions will never lead to optimal solutions. Using this knowledge, the GA does not move labels to hopeless positions that will not lead to solutions that improve a solutions fitness. The conflict table also allows for efficient evaluation of solutions during the GA’s operation, as all possible label conflicts have already been determined.

Raidl’s heuristic initialisation of the GA’s starting population allows faster convergence of the population. Heuristic initialisation forgoes random initialisation of the population in favour of a simple algorithm aimed at creating better initial solutions. Although heuristic initialisation may cause premature convergence to bad local optima, Raidl tunes it to promote population diversity. To maximise population diversity, this heuristic initialisation generates a solution by choosing two label positions randomly for each site and selecting the position with the smallest conflict level. This method ensures an element of randomness, while guiding the initial population towards a reduced number of conflicts.

Experimentally, Raidl’s GA achieves slightly better quality results compared to the simulated annealing used by Christensen et al. [4]. The GA is faster than the SA approach for problems of less than 250 sites, but slower for larger problem sizes. The better performance for smaller problem sets is realised by the GA through the use of the conflict table and heuristic initialisation. As the algorithm performs slowly on large problem sets, it may not be feasible to run in certain circumstances.
Map Labelling GA by Dijk et al. A similar approach that applies a GA to the MLP was proposed by Dijk et al. [22], where a GA is described that uses elitist recombination. Elitist recombination is recombination that operates by randomly pairing two parent solutions and creating two new offspring from them. These offspring compete with their parent solutions for survival into the next generation. Rather than performing mutation on individuals, as used by Raidl [17], Dijk et al. use a local optimiser to improve candidate solutions. This is a procedure that operates only on a subsection of the solution with no regard for the rest of the solution and must have the following properties:

- It should attempt to improve a solution and never worsen it.
- If possible, it should promote diversity by randomly choosing between local optima.
- As this procedure will be applied to a huge number of map subsections, it must be fast.

Notable differences between the GA of Dijk et al. and Raidl are found in population initialisation, the use of a conflict table, and the use of local optimisers rather than mutation. Aside from these, the GA described by Raidl operates similarly to the one described above. Performance is described as better than the SA map labelling algorithm described by Christensen et al. [4].

2.3.6 Problems with SA and GA map labellers

Although SA and GA map labellers have proved useful when used to minimise label to label conflicts, they have not been used to optimise for other objectives that relate to the overall quality of a labelled map. Even though label to label conflicts are an important measure of map quality, Imhof [13] discusses a number of important considerations for producing maps of high quality. Neither the SA or GAs described previously attempt to optimise for the font size, clarity or aesthetics of map labels, each of which directly relate to how easily a map can be used.

Furthermore, the representation used by these map labellers forms an artificially restricted search space. Only 8 label positions may be chosen for each site, although in reality the number of good positions is often much higher. By using a continuous representation, or extending the discrete model to a larger number of label positions, better labellings may be found. Further work is required to better solve this problem.
2.3.7 Evolutionary Algorithm Approaches

**Preuß’s EA for Map Labelling** EAs were first explored as possible solutions to the map labelling problem by Preuß [16]. Preuß describes an evolutionary algorithm to solve the MLP that yields decent results factoring in a number of different objectives.

Under Preuß’s EA, the fitness function incorporates more than label to label conflicts when evaluating the fitness of a candidate solution. The fitness function factors in a variety of label overlap conflicts, a label to site distance measure, and position preferences. The fitness function includes a series of weights that are chosen based on the relative importance of each factor.

These weights are (listed in order of importance by Preuß):

- \( w_{ls} \) — overlaps between a site and label,
- \( w_{ll} \) — label to label overlaps,
- \( w_{lb} \) — label to map edge overlaps,
- \( w_a \) — distance between sites and labels, and
- \( w_p \) — position preferences.

Preuß’s fitness function is calculated as:

\[
\text{fit}(M_i) = w_{ls} \times no + w_{ll} \times nlc + w_{lb} \times nec + w_a \times sd + w_p \times slp
\]

where \( M_i \) represents map labelling \( i \), \( no \) is the number of site/label overlaps in \( M_i \), \( nlc \) is the number of label/label overlaps, \( nec \) is the number of label/edge overlaps, \( sd \) is the summation of the distances from each feature to its label and \( slp \) is the summation of each labels preference position (based on Imhof’s rules).

Unlike most other methods for solving the MLP, which use discrete label positions, Preuß represents label positions with a modified polar coordinate system. These positions consist of an angle and a radius denoting the distance from the site to the label’s closest point. Polar coordinates allow for an infinite number of label positions, as opposed to the 8 fixed label positions used in previous approaches and thus better solutions may be found by the algorithm. However, this additional diversity comes at a cost of a larger search space, which may become too large if not carefully implemented.

Using this fitness function, Preuß forms an evolutionary strategy to solve the MLP. This EA uses recombination that utilises portions of the genotype
from four different individuals to create a new candidate solution. Although Preuß has shown recombination to be useful for map labelling EAs, it is also computationally expensive.

Preuß demonstrates that his evolutionary strategy yields useful results on smaller maps. However, any improvement in map quality achieved by Preuß’s algorithm is likely to be at the expense of CPU-time. This is due to the consideration of several quality objectives separate from the conflict measure, such as aesthetics and distance summation. Unfortunately as Preuß’s algorithm is computationally expensive, he did not demonstrate his algorithm on larger maps. Preuß’s EA is more computationally expensive, often by at least an order of a magnitude, than the other algorithms described earlier.

As noted earlier, Preuß’s fitness function operates with a clear ordering of objectives. When labelling maps the ordering of these objectives may not be clear as these requirements are subjective and context dependent. For example, is a map containing no conflicts better than a map containing one or two conflicts but with a bigger font size?

By combining objectives using predefined weights in the EA’s fitness function there is an implication that these objectives lend themselves to ordering. In many cases it may be better to accept a conflict if it allows a labelling solution to be clearer and more readable. One of the labels involved in this conflict could then be removed allowing for a more readable map. Also, several other objectives such as font size and label clustering are not considered by Preuß’s fitness function, thus restricting the quality of the solutions produced by his algorithm.

An alternative approach is to maintain a variety of labelled maps which are better than each other in one or more respects. As the objectives relating to map quality cannot be easily ordered, a user could instead choose from maps produced by the algorithm and use a map that complies to their personal requirements. For example, consider the problem of map generation on online map websites. These websites generally use a configurable zoom level which changes the level of detail at which a map is shown. A map labelled with a fitness function that combines all different objectives into a single fitness value may not function well at all detail levels. Either the algorithm needs to be run independently for each different detail level, or a means of handling multiple objectives must be realised. Another example is generating maps for visually impaired people who may require an increased font size for improved readability. Maps that function well for some may not fulfil the requirements or preferences of others.
2.3.8 User Hints during Optimisation

Nascimento et al. [9] extend general map labelling algorithms by using hints from a user to find better quality labellings. Their ‘user hints’ framework requires a human user to guide an optimisation algorithm towards optima by providing the algorithm with suggestions. This can be beneficial when used on difficult problems as the combination of human cognition and machine processing power may lead to better solutions than possible from either alone. When solving the MLP, problem is represented in a similar fashion to the other labelling algorithms, using a representation consisting of four possible label positions in quadrants surrounding a site. Although far from total automation, Nascimento et al. believe this is a practical solution to the map labelling problem, as a human can discern aesthetic and readability issues more easily than a machine. Despite human interaction, their method reportedly saves much of the monotonous work required to create a high quality map labelling.

A human expert supervises the optimisation process, at each stage guiding the algorithm towards a usable local optima. After generating a possible solution at each iteration of the optimisation algorithm, a user is asked for possible labelling hints. The program incorporates the user’s solution into the optimisation algorithm, which in turn provides a new solution for user input. The program repeats this procedure until it achieves a labelling of sufficient quality.

Nascimento et al. do not describe the particular optimisation technique underlying the user hints framework. Any global optimisation technique, such as simulated annealing or a genetic algorithm, can be used.

While this is an interesting technique and may lead to better results than other methods, it does not address many of the issues posed by fully automated map labelling. For example, this technique is not feasible when used for the online generation of maps, or in geographic information systems such as car GPS systems. Furthermore, maps may still be costly to produce as they still require a human expert. Despite these disadvantages of the user hints method, its use may be beneficial when attempting to reduce overall map production costs when other methods are not good enough. Even so, their application to the MLP suffers from many of the same traits as the others described earlier — it does not consider many of the important objectives that are required when producing high quality maps.

As we’ve already seen, we need to consider a number of objectives in order to produce the ‘right’ map for the ‘right’ job. If possible, we should avoid combining these into a single objective as this requires us to supply the relative weightings of each objective ahead of time, which is something that we can’t accurately do in
every situation. We would instead like to use a multi-objective approach in which objectives such as clarity, label conflicts, font size, and aesthetics are considered independently. While this approach may increase the run-time of the labelling algorithm, it produces a range of solutions that allows users to select an output which conforms to their preferences.
CHAPTER 3

Materials and Methods

Solving the map labelling problem using an MOEA is important if we are concerned with finding maps that serve the interests of a wide variety of possible users. Using an MOEA, we can find a set of possible map labellings, all of which exhibit different properties desirable to a reader. Each of these cannot clearly be shown to be better than the other and each may be useful in different circumstances.

3.1 Modelling the Problem

In order to use an EA to solve the map labelling, an appropriate representation for modifiable parameters must be determined and methods for determining the fitness of a candidate must be formulated.

3.1.1 Representation

We represent a site on a map as a coordinate, translated from longitude and latitude values from GNIS database files [20]. Each site has an area equal to a circle with a radius of four pixels centred on that coordinate. This area is required to be large enough for the site to be easily identified, but not so large that it makes labelling tasks infeasible or the true location of sites difficult to determine.

We represent labels in our algorithm with bounding rectangles. These are defined as the smallest rectangle that surrounds a site’s name at a given font size and typeset. The properties of these bounding rectangles are determined based on the font size, font typeset, and site name (e.g. Perth) using the Java graphics API.

Most of the labelling algorithms described in Chapter 2 allow a label to take a set number of positions around a site. Using preset label positions restricts
the possible solutions that can be found by the algorithm, and thus restricts the quality of possible solutions. As such, we have opted for a representation that allows a greater range of positions to be taken. To facilitate this we use a polar coordinate position representation consisting of an angle and a radius.

Figure 3.1: Under a conventional polar coordinate scheme, it is possible for labels to overlap their own sites as they have area.

As each label has an area, a conventional polar coordinate scheme may lead to illegal label positions. As the polar coordinate scheme only accounts for the position of a single point relative to another point, the position of a label must be determined relative to a static point on a label, e.g. the bottom left corner. Unfortunately, this method causes cases such the one demonstrated in Figure 3.1, in which a label overlaps its own site.

Figure 3.2: Shown on the left: a conventional polar coordinate system using a static point attached to the label, where large changes in radius are required to move the label. Shown on the right: the modified polar coordinate system using minimal attachment distance, where only a change in angle is required to move the label.

Another issue of concern is that small changes in a label’s polar coordinates should cause small shifts in a label’s position during the mutation phase of the EA. However, given a polar coordinate scheme where the position of a label is formed relative to a static point on the label, large changes in radius or angle are often required to achieve relatively small shifts in label position. This problem is demonstrated in Figure 3.2. In this example, the label movement in the left
diagram under a conventional polar coordinate scheme requires a doubling of radius and a small change in angle. Under our modified polar coordinate scheme this movement requires only a change in angle.

Figure 3.3: Many different label positions have the same minimum attachment distance. For example, the three labels at the top of the circle in the example above all have the same distance. These are ‘sliding labels’. Under Hirsch’s [11] modified polar coordinate system, these positions all have different angles with the same radius.

We have overcome these problems by re-defining radius as the distance of the site to the closest point on its associated label, hereafter referred to as the minimal attachment distance. Under this scheme, rather than determining the position of a label using a static attachment point on the label, e.g. the bottom left corner, the point of attachment is determined from the angle. This is the same representation as used by Hirsch’s map labeller [11]. For example, there are a range of positions at the top of the circle shown in Figure 3.3 that have the the same minimum attachment distance.

Our modified definition of angle allows sliding labels at these points. Sliding labels are labels with the same minimum attachment distance and with a different points of attachment (e.g. bottom right corner rather than the bottom left corner). These are also demonstrated in Figure 3.3. As the radius remains the same for these sliding labels, we must spread out the range of angle to allow for these label positions (i.e. angle defines the point of attachment as well as the position of the attachment point). This definition of angle allows the label to slide around the circumference of a circle with a given radius.
The modified polar representation also allows radius to be used as a meaningful variable in the calculation of objectives that we will describe in Section 3.1.2. For example, when attempting to determine how clearly a label is able to associated with its corresponding site, we require the distance from the site to the closest point on its label. Under a conventional polar coordinate scheme, radius would be meaningless in this case.

3.1.2 Objectives

Most previous papers mentioned in Chapter 2 attempt to optimise a map's labelling for a single objective, namely the number of conflicts between labels. However, it may be beneficial to simultaneously optimise multiple objectives that relate to the quality of the map produced. For example, it may be worthwhile to maximise a label's font size or increase a map's aesthetic appeal. By incorporating other objectives, a labelling algorithm should be able to produce higher quality maps when considerations other than the number of conflicts are important. We will now discuss the different objectives and their importance to the map labelling process.

Figure 3.4: An example label to label conflict. Labelling algorithms should aim to minimise the number of label to label overlaps.

**Number of conflicts:** All algorithms attempting to solve the MLP aim to minimise the number of overlaps, or conflicts, between labels and other sites. Several types of label conflicts are possible:

- label to label conflicts (weight of 2),
- label to site conflicts (weight of 3), and
- label to map border conflicts (weight of 8).

As a label is defined by a bounding rectangle, a label to label conflict occurs when two bounding rectangles intersect, as shown in Figure 3.4. Similarly, a site and label conflict occurs when a label overlaps a site point. These are considered more highly in the conflict objective as they obscure the site in question. Note that they are also likely to cause a conflict with another label. A label to map
border conflict exists when a label passes beyond the viewable portion of the map.

![Figure 3.5: A portion of a map with bad clarity. It must be easy to determine the association between each site and its label, otherwise it may be difficult to read the map.](Image)

**Clarity:** Although labels may be close to their associated site, it may still be difficult for a reader to associate a given label with the correct site if other labels are also close to the given site. Often when this is the case, the reader must undergo a process of exclusion in order to determine the correct association. An example of a map with bad clarity is shown in Figure 3.5.

The clarity metric aims to balance a pulling force between a label and a site, and an opposing force between the site and the next closest label. For each given site, this measure can be calculated as $\text{clarity}(s) = r_s + r_{sl}^{\frac{1}{2}}$, where $r_s$ is the radius between a site and its own label and $r_l$ is the distance between the same site and the closest other label. Under this definition, as other labels become close to a site there is a high incentive in the fitness function to push the label away from the site. A map with high clarity has labels that are closer to their own sites than any other labels and are thus clearer to the reader.

The overall clarity of a map is determined by a summation of the clarity for each site in the map. Calculating a map’s clarity has a cost of $O(n^2)$, as finding the closest label has a cost of $O(n)$.

**Font Size:** The font size objective determines the size of the labels. A larger font size corresponds to a larger, and hence easier to read label. This is important for visually impaired people who may find smaller font sizes hard to read. However, a larger font may cause additional label conflicts, may make it harder to associate site sites with their given label and negatively affect the aesthetic qualities of the map. Thus, the map labeller must attempt to maximise font size as much as possible without compromising the aforementioned factors. Unlike the other objectives which are determined from each label position, this objective is determined from a single parameter that governs the entire map.
Aesthetic qualities: As Imhof [13] notes, aesthetic qualities of the map should be promoted whenever possible to minimise unsightly label clusters. Two of the most important aesthetic qualities are that the map should not be overly clustered, and should not be overly symmetrical as this will distract from map features. Preuß [16] incorporates these aesthetic qualities into his fitness function by defining the following functions to measure them.

Site Density: Overly clustered labels detract from a map’s readability. Preuß [16] measures clustering by calculating the density around each site. The statistical mean of site densities gives a way of comparing measure of overall map clustering.

Following Preuß [16], the local density at a site is calculated as the sum of the area of all map labels scaled by that label’s proximity to that site. Local density at site \( i \), is calculated as:

\[
sitedensity(s_i) = \sum_{l \in L} \frac{\text{area}(l)}{\text{dist}(s_i, l)^2} + \sum_{s \in S} \frac{\text{area}(s)}{\text{dist}(s_i, s)^2}
\]

where the \( \text{area} \) function calculates the area of a map items (label, \( l \), or site, \( s \)) and the \( \text{dist} \) function calculates the Euclidean distance between the two closest points of two map items.

This density may be used in the calculation of two primary aesthetic indicators. The first is mean density, representing the level of clustering, and the second is the overall symmetry of the map.

Mean Density: The mean density indicates the amount of clustering that a map exhibits. This is calculated by summing the density for each label and dividing by the number of labels. The mean density should be able to indicate the presence of high levels of clustering. Clustering should be avoided as it leads to dense regions which are harder to read and not visually appealing.

\[
\text{meandensity}(M_i) = \sum_{s \in S} \frac{sitedensity(s)}{|S|}
\]

where \( M_i \) is the map labelling \( i \), \( S \) is the set of sites and \( |S| \) is the number of sites being labelled.

Map Symmetry: Symmetry is measured by calculating the standard deviation of local densities. A low standard deviation indicates a high overall symmetry as the map is quite uniformly clustered. This case should be avoided as uniformity impedes the easy identification of sites and their associated labels.
\[
\text{symmetry}(M_i) = \sqrt{\frac{1}{|S|} \sum_{s \in S} (\text{sitedensity}(s) - \text{meandensity}(M_i))^2}
\]

where \( M_i \) is the map being labelled, and \( S \) is the set of sites. Each additional objective typically corresponds to a large increase in running time in an MOEA as the population size would need to be increased to compensate for the larger search space. Due to this, we decided to group these aesthetic measures into one composite aesthetic objective by summing the different metrics together.

The overall aesthetics objective is calculated as:

\[
\text{aesthetics}(M_i) = \text{meandensity}(M_i) - \text{symmetry}(M_i)
\]

where \( M_i \) is the map being labelled. We have subtracted the symmetry measure from the mean density as we are attempting to minimise mean density and maximise the standard deviation of the overall density.

Conflicts, clarity, aesthetics and font size are all important measures of map quality in their own respect. We believe that they cannot be clearly ordered and thus we will use an MOEA to optimise for these objectives simultaneously. By doing so, we believe that we will produce a wider variety of useful maps that can be used in different situations. For example, in many cases it may be worth sacrificing some of a map’s aesthetic appeal if it improves a map’s overall clarity. The compromise for this technique is additional run-time required to compensate for the increase in search space.

### 3.2 Algorithms and Implementation

#### 3.2.1 Representation

In the case of a map labelling EA, each individual solution in the population is represented by a set of label positions, \( L \), that contains the positions of the labels on the map relative to the position of map sites. Each label position consists of a radius and an angle, from which a label bounding box with the correct position can be derived. Each individual also contains a font size variable that represents the font size for that map labelling. Thus, the genotype for an individual is composed of \( 2L + 1 \) parameters: 2 for each label position and one additional value to control the font size. From these variables, all of the objectives discussed above can be measured and used for comparison within an EA.
3.2.2 MOEA to solve the MLP

A primary divergence in operation from the Map Labelling EA proposed by Preuß [16] and this work is the use of a multi-objective EA (MOEA), as described in Chapter 2, to determine which candidates should be maintained each generation rather than using a linear weighted sum fitness function. Preuß [16]’s algorithm converges to a single ‘best’ solution that represents the best fitness that can be achieved when summing all the objectives together. The main problem with this approach is that we have no way of determining the optimal weightings for this fitness function in all cases. In an MOEA, solutions are optimised for all objectives resulting in a ‘front’ of trade-off solutions that are equally as good as each other. By doing so, the MOEA will not compromise one objective over another, ensuring that all objectives are optimised simultaneously.

The MOEA used for map labelling instead keeps these objectives in a series of separate values and sorts candidates into ranks of non-dominated solutions. Recall that a solution is said to dominate another solution when it is at least as good in all objectives and better in at least one objective.

After each set of new individuals are generated by the evolutionary algorithm, they are separated into fronts of non-dominated solutions. These are ranked using the NGSA-II ranking algorithm by Deb et al. [8], as described in Chapter 2, which produces a set of solution fronts in \( O(m(2n)^2) \) time — \( m \) being the number of objectives and \( n \) the population size. Each of the fronts produced consist of a set of solutions which are not dominated by any other solutions in that rank or any lower rank.

The algorithm proceeds as follows:

1. Initialise the population.
2. Sort all solutions into fronts.
3. Add all solutions from the best fronts that are small enough to fit into the population when combined.
4. For the last front that won’t fit into the population, sort the solutions by the crowdedness measure and select the best solutions from the population.
5. Apply recombination and mutation operators to all surviving solutions.
6. Repeat steps 2 – 4 until population the has converged.
**Recombination:** Preuß [16] employs recombination, as described in Chapter 2, in his EA as a method of generating new solutions every generation. Recombination is effective in map labelling EAs, as maps are often reasonably partitioned and often have disparate clusters of sites. Hence, recombination is able to share good labellings for sections of each labelling with other solutions, saving new individuals from solving sections of the map that have already been effectively labelled by current individuals.

With the large population size and high number of objectives, recombination for the MLP can be very expensive. This is because computation of each of the objectives from scratch costs $O(|L|^2)$ time, where $L$ is the set of labels. It is thus computationally cheaper to mutate a solution and calculate the change in the fitness of the solution. This update still has worst case performance of $O(|L|^2)$, however in general only a small proportion of label positions are modified and thus average case performance is $O(c|L|)$, where $c$ is the number of labels that are mutated.

Although it takes significantly more time to use recombination as opposed to pure mutation (a factor of $|L|c$ more expensive), recombination should lead to convergence to better solutions in less generations. We will examine later whether enabling recombination improves the results found by the MOEA and whether good results can be obtained more quickly in comparison to pure mutation operators.

**Mutation:** Mutation of a solution consists of the modification of one or more variables making up the solution. In the case of the map labelling EA, these consist of the font size and label positions. Note that a good solution for one font size, may or may not be a good solution for another font size, so mutation of the font size may be destructive. Label positions may also be mutated. During mutation, the position of the label (as defined by its angle and radius) may be modified. This has the effect of moving a given label to another position, which may reduce one or more of the labelling solution’s objectives.

Mutation of an individual’s font size and label position occurs via mutation by a Gaussian distributed random number. Under this scheme, small mutations in label position or font size occur with higher probability than large moves. Note however, that large mutations are still possible.

After each label position is modified, the change in each of the map objectives is determined and the objectives modified. By only measuring the change in each objective, the cost of modifying each label position is of cost $O(|L|)$, where $L$ is the set of labels, rather than a cost of $O(|L|^2)$ as discussed earlier. It is simple to find the change in each objective from a label mutation by removing
a label and its contribution to each objective and then re-adding the modified label’s contribution to the same objective. This is possible because each site’s contribution to each objective is independent to other sites and thus we can quickly calculate the change in each objective when a label changes position.

When only mutation operators are used, each individual creates and maintains objective scores for each possible font size. As only a fraction of the label positions are modified on average with each mutation, each with associated cost $O(n)$, it is less expensive to maintain a set of objectives values for each font size than to recompute each objective from scratch when a change in font size occurs in a mutated child solution. That is, we cache all objectives for each font size and calculate the change in these objectives after every mutation.

3.3 Memetic Algorithm

*Memetic algorithms* employ local search techniques at various stages throughout the running of an optimisation algorithm to better improve the results found. The idea is to perform local optimisation on the best found solutions without simultaneously causing premature convergence to a suboptimal solution. A similar technique was used by Dijk et al. [22] in their map labelling GA. The memetic algorithm we’ve implemented uses a simple hill climbing algorithm which modifies the angle and radius of each label only while each new label position dominates the last. In order to promote diversity, labels should be optimised in random order so that the same label positions are not optimally arranged in every individual. Two questions arise: does the memetic algorithm produce better overall results than the non-memetic approach (or does it lead to premature convergence on non-optimal solutions) and secondly, how frequently should the local optimisation be run? We will later explore whether a memetic algorithm impedes the progress of a our MOEA, or if it in fact improves its performance.

The memetic algorithm operates as follows:
Shuffle label positions.
for all label positions do
    while new individual dominates old individual do
        decrease position’s radius
        while new solution dominates old individual do
            increase or decrease position’s angle
        end while
    end while
end for

25
CHAPTER 4

Experimental Results

In this chapter, we report on experiments that demonstrate our MOEA solution to the MLP. Our first series of experiments attempt to determine reasonable choices for the parameters that control our MOEA such as population size and recombination probability. Our second series of experiments will visually compare the results of our MOEA to Preuß’s existing map labelling EA [16]. We will also show several maps which exhibit the qualities of each of the different objectives.

4.1 Environment

All experiments were run on a Pentium 4 2.80GHz machine with 512MB RAM. Experiments were run on the RedHat Enterprise Linux WS 3.0 platform using Java2 v1.4. Due to the large memory required by these experiments, tests were run using the Java runtime switch \(-Xmx256M\), which sets the maximum heap size for the JavaVM to 256MB of memory.

4.2 Hypervolume

To calculate reasonable values for the parameters of our MOEA, we first need a means of comparing the results of different runs of the algorithm. Recall that our MOEA produces a front of solutions, so we require a means of comparing the different fronts generated by different runs of the algorithm. To do this, we use the concept of hypervolume. The hypervolume metric [19] measures the overall volume of the objective space dominated by the approximation of the best (final) front found by the optimisation algorithm. We use the hypervolume value as the single measure for evaluation of the final population of our algorithm. As noted by Okabe et al. [19], while hypervolume may not always not be a good measure for performance of MOEA, it is generally better than others. Furthermore, it allows us to compare our MOEA objectively and is a good comparison when
used to compare the final fronts obtained from the same algorithm using different parameters.

A good paper explaining how hypervolume operates as well as a fast algorithm for calculating hypervolume is by While et al. [15]. I have used the WFG algorithm to calculate the hypervolume of the pareto fronts achieved by our MOEA. Although hypervolume is a good measure for the evaluation of a front, we have not used it during the operation of our MOEA as it is expensive to calculate and we generally require a large number of generations to adequately label a map.

The hypervolume measure has been used in this work to determine the optimal MOEA parameters such as the population size and recombination probability for labelling our test maps. After determining these, we have shown that the MOEA converges to a good approximation of the pareto-optimal front of solutions that has good coverage of the objective space.

We will then show various final labelled maps that exhibit the value of the different objectives. This will show that a choice of labelled maps with different qualities is worthwhile and a valuable proposition when labelling maps. Finally, we will visually compare a map of Guam labelled with our MOEA map labelling algorithm with the same map labelled by Preuß’s EA.

4.3 Comparing running time by fitness evaluations

We have implemented a measure of cost associated with recombination and mutation so they can be compared without regard to the elapsed wall clock time, which is not always an accurate indicator of the processor time used, especially on multi-user systems. Each label change costs one ‘evaluation’ (our measure for labelling run time) and each instance of recombination costs \( n \) evaluations. These costs are in line with the running time that will be needed to process a label change and recombination respectively.

4.4 Experiment One: Determining the Population Size

This experiment is intended to determine the population size, given that an increase in population size will cause a decrease in the number of generations that can be achieved in the same run-time. Using a large population also allows for a greater diversity of solutions in the final front and should achieve better results. Tests were run with population sizes ranging from 100–1000 individuals. Experiments were run for 10,000,000 evaluations, using the comparison method
described in Section 4.3. Each test was run ten times and the hypervolume achieved for that population size was then averaged. Results are plotted in Figure 4.1.

![Figure 4.1: Hypervolume achieved after 10,000,000 evaluations for varying population sizes for the map of Guam (51 sites). We can observe a steady increase as population increases to 300 solutions after which only a small increase is observed.](image)

As we can clearly see in Figure 4.1, there is an increase in the Hypervolume obtained by the MOEA as population size increases. We can observe a steady increase in hypervolume as the population size increases from 100 to 300 individuals. After this point, we observe only a small increase in hypervolume as the population size increases.

However, as the cost of ranking individuals into non-dominated fronts increases as the population size increases, increasing the population size comes at a cost of overall running time. Although using a population size of 600 gives only a small improvement in hypervolume over a population of size 300, we have chosen to use a population of size 600 as the increase in run time is minimal and the increase in population relates to a small improvement in the diversity of final solutions.
4.5 Experiment Two: Determining the Recombination Probability

This experiment is intended to determine whether recombination, given its high runtime cost, is beneficial when labelling maps. This experiment consists of running the map labelling MOEA with recombination used at varying probabilities. This technique allows recombination to still be used without incurring the full cost of performing recombination every generation. We will be labelling a map of Guam, a small map with 51 sites, also tested by Preuß [16].

Tests were run with a population of 600 individuals, as obtained from the first experiment, for 10,000,000 evaluations as described in Section 4.3. Each test was run ten times and the hypervolume achieved for that recombination probability was then averaged. Results are plotted in Figure 4.2.

![Figure 4.2: Hypervolume achieved after 10,000,000 evaluations at various recombination probabilities with a population size of 600 individuals.](image)

Figure 4.2 shows that as the probability of recombination increases, the hypervolume also increases. As we are trying to maximise hypervolume, we can
observe that a higher recombination probability is beneficial for map labelling using our MOEA. Even though recombination is expensive, (recall that this requires us to recalculate all objectives from scratch), it is clear that the sharing of good parameters is beneficial to the operation of the EA.

This process was also repeated with larger maps and the same result was observed. Unfortunately, as larger maps take many more generations to converge, multiple runs were not carried out. Even so, the overall trend is towards using recombination with a high probability as results achieved generally show a large improvement when using recombination.

4.6 Convergence using optimal parameters for Guam

This experiment is intended to show that all objectives are simultaneously minimised under our MOEA. These tests will be carried out using the population size and recombination probability derived from the first two experiments. We used a recombination probability of $0.9$ with a total population size of 600. A recombination probability of $1.0$ was not used, as it did not always lead to solutions with zero conflicts, even though the average hypervolume achieved with these parameters was slightly better than $0.9$.

We cannot easily plot all four dimensions (objectives) in one graph to show convergence, as it would be hard to interpret. Instead we will show a series of plots displaying each pair of objectives, i.e. conflicts vs clarity, conflicts vs aesthetics, etc. This process results in six graphs that demonstrate that all objectives are simultaneously minimised.

4.6.1 Results

Figure 4.3 shows the movement of the pareto front for conflicts and clarity. Note that each new front completely dominates the last at each successive generation. As no crossover occurs between fronts, there are no solutions within each successive front which dominates a solutions from a later front. Thus each new front moves achieves a better pareto front than the last. It is clear that the MOEA is able to simultaneously minimise the clarity and conflicts objectives. Note that the compromise required to make small gains in clarity using solutions from the final pareto front requires a significant increase in the number of conflicts. Thus for Guam, most users will chose a map with 0 or 2 conflicts and a clarity of 110–120. This compromise covers a large proportion of usable labelled maps when considering clarity and conflict objectives.
Figure 4.3: Convergence of Conflicts vs Clarity
Figure 4.4: Convergence of Conflicts vs Aesthetics
Figure 4.4 shows the movement of the pareto front for conflicts and aesthetics. Each new front completely dominates the last for generations 0–300. After generation 300, the MOEA has found a front that generally minimises both objectives. At this point the MOEA appears to search for an optimal spread of solutions for these objectives. We believe that after 700 generations the MOEA has found a pareto front that is a good approximation for the pareto optimal front for the conflict and aesthetics objectives.

Figure 4.5: Convergence of Clarity vs Aesthetics

Figure 4.5 shows the movement of the pareto front for clarity and aesthetics. Note that each new front completely dominates the last at each successive generation. It is also notable that the MOEA finds solutions with relatively good aesthetic qualities quickly. However, these solutions do not have a particularly high clarity and thus most optimisation occurs with the clarity objective. As a result, we can observe a large decrease in the clarity objective over time, while aesthetics is further minimised to a lesser degree. I believe that this is because clarity is quickly minimised as solutions dispense of their initial random label positions and find better positions. These positions are quickly shared throughout the population through the MOEA’s use of recombination.
Figure 4.6: Convergence of Clarity vs Font Size
Figure 4.6 shows the movement of the pareto front for clarity and font size. Once again each new front completely dominates the last as the number of generations increases. We can observe that clarity is reduced in each successive generation while still minimising font size (recall that the font size axis is inverted so that we are minimising both objectives). Overall, we can see that clarity is minimised without any noticeable compromise to the minimisation of the font size objective.

Figure 4.7: Convergence of Conflicts vs Font Size

Figure 4.7 shows the movement of the pareto front for conflicts and font size. Note that each new front completely dominates the last. Also observe that the number of conflicts is reduced in each successive generation while still minimising font size (recall that the font size axis is inverted such that we are minimising both objectives). After generation 100, all pareto fronts contain only the optimal solution for both objectives, 0 font size (i.e. 12pt font) and 0 conflicts. Overall, we can see that clarity is minimised with a high success without any compromise to the font sizes obtained by the algorithm.

Figure 4.8 shows the movement of the pareto front for aesthetics and font size. The aesthetic objective is minimised in each successive generation while
Figure 4.8: Convergence of Font Size vs Aesthetics
still minimising font size. While the front for generation 300 appears to cross later generations’ fronts for font size 1, this is merely an illusion caused by the linear fitting lines between the plotted points. Overall, we can see that aesthetics is minimised with success, without any compromise to the font sizes obtained by the algorithm.

4.7 Memetic Algorithm Experiment

This experiment is intended to determine if the use of a local search at particular intervals of the MOEA improves the results obtained by the EA or reduces the running time needed to find those solutions.

In order to be useful, memetic algorithms should not decrease the quality of results obtained in any optimisation process. It is thus important to determine whether the use of memetic algorithms during optimisation is ever detrimental to the operation of the EA. To determine this, several experiments will be run with the memetic algorithm used at various intervals during the process. From these results, it should be easy to determine whether the use of a memetic algorithm causes premature convergence or if in fact it allows the EA to obtain superior results.

Figure 4.9 shows the hypervolume achieved by the MOEA when our memetic algorithm is used at varying generation intervals. The algorithm has been tested on the map of Guam with a population size 600 and recombination probability of 1.0. Observe that a decrease in hypervolume is measured when the memetic algorithm is used often, at 1, 10 and 20 generations. However, we can observe an improvement in hypervolume when the memetic algorithm is used every 50 generations. This interval appears to give improved results when compared to other intervals and not using the memetic algorithm. Using the memetic algorithm too often appears to be detrimental to the operation of the MOEA, as in our tests it reduced the hypervolume achieved.

4.7.1 Visual Experiment: A comparison between Preuß and our algorithms

Figure 4.10 and Figure 4.11 demonstrate the map of Guam labelled by Preuß’s map labelling EA [16] and our MOEA. It is easy to see that Preuß has labelled his map with a much smaller font size. When compared to the map labelled by our MOEA, we can see that in several cases the map produced by Preuß’s algorithm is harder to read than the map produced by our MOEA, see Figure 4.10. For
Figure 4.9: Hypervolume for MOEA when memetic algorithm is used at varying intervals.
example, notice Chii Point and Mount Jumullong Manglo which are very close to each other and are harder to read in Preuß’s labelling.

Figure 4.10: Map of Guam labelled with our MOEA. This map contains 0 conflicts, a clarity of 110, aesthetics of 1.1, at font size 10.

4.7.2 Examples of final maps exhibiting objective compromises

A key aim of our MOEA for the MLP is its ability to produce a diverse set of solutions that exhibit compromises between quality objectives. The labelling of the Northern Mariana islands shown in Figure 4.12 has relatively high clarity. Compare this labelling to Figure 4.13 also produced by our MOEA which uses a larger font size and has better aesthetic qualities than Figure 4.12. These maps are simple examples of the compromises exist in the labelling process. A
Figure 4.11: The final map labelling of Guam produced by Preuß’s map labelling EA [16].
Figure 4.12: Map of Northern Mariana Island labelled with MOEA exhibiting high clarity. This is a good example of a labelling that minimises ambiguity in site labels.
Figure 4.13: Map of Northern Mariana Island labelled with MOEA exhibiting high aesthetic qualities. This is a good example of a labelling that maximises the aesthetic qualities in site labels and uses a larger font size than the labelling in Figure 4.12.
cartographer using our MOEA would view a selection of solutions such as these and then choose the solution which they decide best fits their requirements. In these figures, the cartographer may decide that a larger font size and better aesthetic qualities are more important than further optimising clarity. That said, another cartographer with a different set of requirements may decide differently and choose the solution with higher clarity. Our MOEA gives a user this choice, and allows them to ultimately decide which labelling best fits their requirements.

Figure 4.14: A map of Delaware (185 sites) labelled with our MOEA. This map contains 0 label conflicts and has a relatively good clarity which allows most labels, with the notable exception of Madelyn Gardens, to be easily associated with their site by a user.

I have also labelled the map of Delaware, shown with 185 sites in Figure 4.14, to demonstrate that larger maps are feasible with our MOEA map labeller.
CHAPTER 5

Conclusion

This dissertation has examined the problems posed by the MLP. Current map labelling algorithms, though effective when used to minimise label conflicts, are not always effective when labelling maps for real world use. These algorithms generally do not consider additional measures of quality that are important when producing quality maps. Imhof [13] defines several measures for the evaluation of map quality: conflicts, aesthetics, ambiguity (clarity) and legibility (font size). Algorithms that do consider these measures treat them as secondary objectives and do not give them high priority during the optimisation process.

Preuß’s [16] map labelling EA described in Chapter 2 includes measures for all of map labelling requirements defined by Imhof [13] in its fitness function. This is an improvement over the SA by Christensen et al. [4] and GAs by Raidl [18] and van Dijk [21] as its consideration of other quality objectives improves the overall quality of the map produced by the algorithm.

As labelling maps is a very subjective process, the relative importance of different objectives may not be clear prior to the optimisation process. As Preuß presents his EA’s fitness function with a set of static weights, it will make compromises throughout the optimisation process which may not represent the choices that would be made by a cartographer when labelling the map. Preuß’s EA will arrive upon on a single map that minimises his fitness function. This map usually does not represent the subjective compromises that might be made by a human operator.

In this dissertation, we have introduced an MOEA algorithm for map labelling that is able to arrive on a diverse set of labelling solutions. These solutions represent a compromise between one or more quality objectives. Unlike other labelling algorithms, all quality objectives are simultaneously optimised with our MOEA — that is, trade-offs are not made during the optimisation process. After optimisation has completed, a user is able to choose a map that conforms to their specific requirements, whether they are looking for a map with good clarity, large fonts, low number of overlaps or a map that is aesthetically pleasing.
Our algorithm lends itself well to the subjective nature of the MLP as the set of potential maps produced are able to be visually inspected by a cartographer. He or she can then make an informed choice of the best compromise between quality objectives for that given map. As the optimal compromise between objectives changes from map to map, algorithms that evaluate maps based on a predefined weighted sum are not able to adapt to the best compromise solution.

We have shown that our MOEA is able to simultaneously optimise multiple quality objectives and settle on a final pareto front for the multiple objectives. Several final solutions for real world maps were demonstrated to show that the MOEA is able to create usable solutions for real problems. A memetic algorithm, implemented as a hill climbing algorithm used occasionally throughout the optimisation process was shown to achieve modest improvements for map labelling. Finally, we have selected several final labelled maps which demonstrated compromises between quality objectives. These are intended to give the reader a better understanding of the choices given to someone using our MOEA for labelling maps which require a non-obvious compromise between the quality objectives.

We believe that we can report a degree of success when applying MOEAs to the MLP. Unlike competing map labelling techniques, our algorithm optimises more than just the map’s label conflicts. Quality objectives such as map clarity, font size and aesthetics are often very important to a map’s usability — most map labelling algorithms do not even consider these qualities when optimising map labellings. As improving an objective may require compromises in one or more other objectives, our map labelling MOEA produces a front solutions, each of which are better than each other in one or more respects. We have shown that our MOEA successfully optimises these solutions and presents a diverse set of solutions to a user so that they can choose an optimal labelling based on their own subjective set of requirements.
APPENDIX A

Original Honours Proposal

Title: \textit{A Comparison of Optimisation Techniques Applied to the Map Labelling Problem}

Author: Lucas Bradstreet

Supervisors: Dr. Luigi Barone and Dr. Lyndon While

Background

The Map Labelling Problem

The Map Labelling problem refers to the difficulty encountered when arranging labels – each of which refer to a specific feature on a map – such that these features can be uniquely identified by a reader. Due to the high number of degrees of freedom in each label, this is a complex combinatorial problem, which has been proven to be NP-hard [24].

Map labeling makes up a large proportion of map production costs and remains a time consuming and expensive task for cartographers. As a result of its importance to cartography, this problem has been targeted for further research by the ACM Computational Geometry Impact Task Force [9]. An automated solution will not only significantly reduce map production costs, but may also allow for real-time generation of maps on map websites and in Geographic Information Systems (GIS).

A set of requirements for good label placement have been defined by Imhof [13] and Yoeli [26] and are listed below.

- \textit{Legibility} — Labels must have legible font sizes and be positioned in a way that is easily read.
• **Unambiguity** — Each label must clearly identify a single feature and not interfere with any other label or feature.

• **Overlap avoidance** — A label should not overlap any other label or feature.

• **Aesthetics** — Labels should not be overly clustered or distract from map features.

Although the task of labelling a map may seem trivial, this is hardly the case. In most real-world cases map labelling is performed by hand. This is done from scratch or by using an algorithm to get an initial sub-optimal solution that is then improved upon by a cartographer. The human brain is better suited to fix aesthetic issues and detect ambiguities in labels than any algorithm available today. As a consequence, the production of maps remains a costly and time consuming affair.

Solutions to the map labelling problem also lend themselves to use in several areas outside of cartography. These include the following:

• Drafting and architecture – technical drawings including labels containing names, measurements and other values.

• Graph diagrams – the labelling of graph diagrams such as visualisation of networks, UML diagrams, state diagrams and others.

• Medical image analysis – the inclusion and automation of labelling anatomical and biological features in medical imagery.

**Evolutionary Algorithms**

Evolutionary algorithms (EAs) are strategies for function optimisation that are useful when solving problems that cannot be feasibly solved on large problem sets. Darwin’s theory of evolution is the inspiration for such a technique [7]. Just as Darwin’s model of evolution applies to natural processes, evolutionary algorithms analogously apply to optimisation problems.

An outline of how an EA operates is as follows:

1. Initialise the solution population.

2. Evaluate the population using the fitness function for the problem.
3. Generate an offspring population from the parent population using mutation and recombination operators.

4. Evaluate the offspring population.

5. A comparison of each solution’s fitness is made using the algorithm’s fitness function. A number of offspring or parents survive this natural selection and the rest are discarded. The surviving solutions become the new parents for the next generation.

6. Repeat steps 3 – 5 until the convergence criteria has been satisfied.

Aim

The primary objective of this project is to carry out a literature review and comparison of several techniques that can be used to solve the map labelling problem. These would include the implementation and testing of basic heuristic approaches [23], simulated annealing algorithms [5], genetic algorithms [21] and EAs [16] [18] that could give satisfactory solutions for most problem sets.

A further aim of this project is to research a novel technique for solving the map labelling problem using multi-objective evolutionary algorithms. Preuß [16] notes the possibility of a good solution to the map labelling problem using evolutionary techniques. Multi-objective EAs are useful when finding optimal solutions when dealing with multiple conflicting criterion. In the case of map labelling, these include label font size, label orientation, and distance of the label from its corresponding feature. In order to achieve a high quality label solution, none of these criterion should overwhelmingly dominate the solution as this may yield a sub-optimal answer. It is my hope that a correctly implemented multi-objective EA will yield a usable labelling configuration solution in a majority of cases.

Method

Research: Literature review on map labelling and EAs. Research simulated annealing and EA approaches to the problem. March-April.

Implementation: Employ a basic implementation of EAs on a simplified version of the map-labelling problem. Implement other algorithmic solutions including simulated annealing as well as heuristic approaches. March-July.
Testing: Experiment with various data sets, fitness functions and implemented solutions. Analyse their uses and weaknesses and update accordingly. June-August.

Dissertation: Although documentation will be written throughout the life of the project, it is inevitable that a large proportion of my time toward the end of the project will be spent writing up my dissertation. August-September.

Software and Hardware Requirements

This project does not have any special requirements. The Intel based machines in the Computer Science School’s Honours laboratory should be sufficient to implement and test labelling implementations. My research will be done predominantly in the Linux operating system using a high level programming language such as Java or C. Use of the UWA high performance computer lab may be needed if my implementations require parallel processing in order to complete within a practical time frame.
Bibliography


