Diplomarbeit

Solving Map Labeling Problems by Means of Evolution Strategies

Mike Preuß

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

19. Februar 1998

Betreuer:

Prof. Dr.-Ing. Hans-Paul Schwefel
Dipl.-Inform. Frank Kursawe

Contents

Chapter 1

The Problem: Map Labeling

1.1 A Step into Cartography

During the last years the computer science has found its way into the business of cartography. It may be surprising that it was common to do the whole design and layout process by hand up to the 80ies. The reason for this may have been the cartographers' attitude towards their own profession, seeing themselves in a field between crafts and art [14] on the one hand, and the complexity of some subtasks of map creation on the other hand.

Meanwhile, computer hardware has been vastly improved, and the financial incentive grew to perform some of the most time consuming tasks (an upper bound of three days is estimated in [26]) of these highly qualified (and most certainly adequately payed) specialists electronically. Some problems, however, turned out to be very hard for automation. One of these is the *Map Labeling Problem*[1], which is the task to arrange some amount of textual information (each belonging to a certain map feature) on the map. Another one is the problem of automatic generalization that has to be done if the data of a map is used to create a map on a smaller scale.

While the cartographers try to integrate the new technology into their work, it takes of course effect on cartography itself. That ought to be discussed a bit more in detail.

1.1.1 Electronic Cartography: Maps on Demand

From the viewpoint of a computer scientist a map can be reduced to a background picture with some points, lines and labels painted onto it. These foreground drawings represent additional data taken from a cartographic database which includes

[1] The string 'Map Labeling Problem' will be abbreviated as MLP later on.

the location and the attributes of every feature. The databases, completed by a collection of spatial operators, are nowadays referred to as *GIS*[2] and are the base technology for computer aided mapping. Actually, maps generated with a GIS software may consist exactly of these two components: a digitized picture (for example a satellite image) and a subset of data of a cartographic database [6]. On the other hand, it is also possible to store the needed graphical description of the surface in the GIS, too. That enables the generation of freely scaled maps on demand.

In my opinion, the use of electronic map generation systems has at least two important effects on cartography:

- A new degree of globalization is reached: national authorities for standardization – for example the *US Geological Survey* or the *Arbeitsgemeinschaft der Vermessungsverwaltungen*[3] in Germany – are collecting map feature data in their databases. Exchange and check against the other databases is now possible, standardized portioning of maps is no more needed. Instant access – via Internet, for example – can be granted to a vast and growing number of people.

- The manufacturing process of a map – even if it is printed and sold in the usual way – will be accelerated by far. Everything that can be done faster by software than by hand will be done if great quality losses can be avoided. As there is no reason why computers should not become faster in the future, this process is limited only by the extent of tasks that can not be done automatically (remember Amdahl's Law[4]).

The speedup of map making renders great possibilities but may also have disadvantages. Eduard Imhof, a famous cartographer from Switzerland who had great influence on the development of map design in the 60ies and 70ies, already stated in 1962 [13]:

> *"Maschinen und Geräte sind höchst nützlich und wünschenswert, soweit sie die Kartenerstellung beschleunigen und verbilligen, doch sind sie von sehr zweifelhaftem Werte, wenn sie nicht auch die Qualität der Karten mindestens zu bewahren suchen."* [5]

[2] Geographic Information System

[3] The AdV establishes a national database named *ATKIS*: Amtliches Topographisch-Kartographisches Informationssystem.

[4] The law was invented in [1] and lays down an upper bound for the maximal speedup that can be reached by parallelization with respect to the parts of an algorithm that have to be computed serially. Transferred to automatization of map-making within here, the computable tasks can be assumed to be processed almost immediately, the others remain to be done by hand as before. So the amount of time used is (in theory) only limited by the interactive parts.

[5] Machines and devices are very useful and desirable, as far as they speedup and cheapen map manufacturing, but their value is very dubious if they do not strive for preservation of map quality.

Nowadays, cartography develops with the help of new algorithms fulfilling tasks that had to be done manually before. Therefore, cartographers and computer scientists have to co-operate and are both responsible for the preservation of map quality.

1.1.2 Foundations of Map Design

What do we have to know about map creation before concentrating on the labeling itself? It should be useful to have a look at the different types of maps presented to us in everyday life to find distinctions and similarities between them.

- At first, there are topographical maps which usually provide the main part of each atlas and are used as the background of more specialized maps, for example weather-charts. They show the land coverage and man-made items that have some areal dimension like towns and roads. Therefore, road maps can be put in this class, too.
- Thematic maps have a more specific scope on any kind of spatial information as the weather condition, density of population or just country borders. They are often shown in newspapers or on television to elucidate military conflicts or economical differences between countries.
- Technical maps are much more abstract than a topographic map. Although they have an underlying spatial structure, they are used for position identification of the different measurements only. The main emphasis lies on the presentation of the acquired data. There is of course no clear distinction between technical and thematic maps, even topographic maps can be arranged into the group of thematic maps as a subset. The classification mostly refers to the intended map use. However, technical maps often fulfill conditions maps of other types do not:
  - There is only one kind of textual information: data boxes with equal width and height, they are usually placed with one edge on the point where the data was acquired.
  - The background image is being ignored while placing the labels; there are no map features apart from other labels that have to be preserved.
- Astronomical maps may look quite similar to topographical maps, depending on the area of the sky that is being mapped. However, there are two notable differences:
  1. The geometric object that is projected onto a plane is no sphere but an almost unbounded three-dimensional space that is looked at from within. That means there can be two or more objects at the same position on the map although they have a distance of many billion miles in reality.

2. There are almost no line features, only point or area objects.

The contents of a map does not only depend on the map type or the area that is being mapped, but on the chosen scale as well. Especially people with a driving license will know this effect from road maps. Therefore, I want to demonstrate the relation of scale to generalization by a close look on topographic maps. To avoid a detailed classification of scales (which can be found in any textbook on cartography, for example [34]), I classify the maps according to their intended use:

- Scales of about 1 : 5,000 upto 1 : 75,000 are used for more or less exact maps of a very small area. Objects measuring only a square meter can be shown on the map, size and shape do not have to be adjusted. Maps of this type are used to lay down the borders of landed property or to back up municipal survey and construction projects.
- Scales of about 1 : 100,000 upto 1 : 800,000 are used to produce road maps (for example). They include one or more towns and the rural area lying inbetween them. Generalization is difficult at these scales because there is often a lack of space to include all the contents of larger scale maps of the same area. Some are simply so small that they could not be perceived if they would be put in at their original size. A road would result in a very long, but very thin line, too thin to print in many cases. The cartographer has to decide on the objects that should be in the map and has to resolve conflicts if two important features share the same place. The chosen objects have to be adjusted in size and shape now: they must be visible on the map but should be distinguishable from others (that limits their size) and besides they have to look similar to their real counterparts.
- Scales equal to 1 : 1 million or smaller are mainly used for atlas maps. They show wide areas of the earth, from countries upto whole continents. At this small scale no details of the ground coverage can be perceived, the shape of every item included is freely adjusted to correspond with the others and all features that do not have major importance are totally left out.

As stated before, the problem of generalization is very difficult for automated map design systems and is not generally solved as two my knowledge.

Another important attribute of a map, some rules for the shape and positioning of textual information are derived from, is the projection that has been used to generate it. In most cases the mapped area is a segment of a sphere in reality[6].

At this point, we encounter an implicit difficulty of cartography in general: an excact plane map of a sphere is not possible, there are always losses of accuracy.

[6]The earth is not excactly a sphere, but that may serve as a sufficient model for most applications.

Only one of three properties can be preserved without distortion: either angles, length proportions or area sizes. Hence, the projection of a map is chosen with respect to its purpose. If this turns out to be the navigation of a ship or airplane, the angles must match the original ones. On road maps, length ratios should be undistorted, and atlas maps showing countries or continents should permit area comparisons and therefore preserve area proportions.

The history of projections reaches back to Ptolemaios and most probably Archimedes, always striving for better compromises for the three contrary aims stated, and nowadays, there are hundreds of different methods[7]. Here is an example showing the effect of two different projections on maps of Arabia and Greenland (figure 1.1).

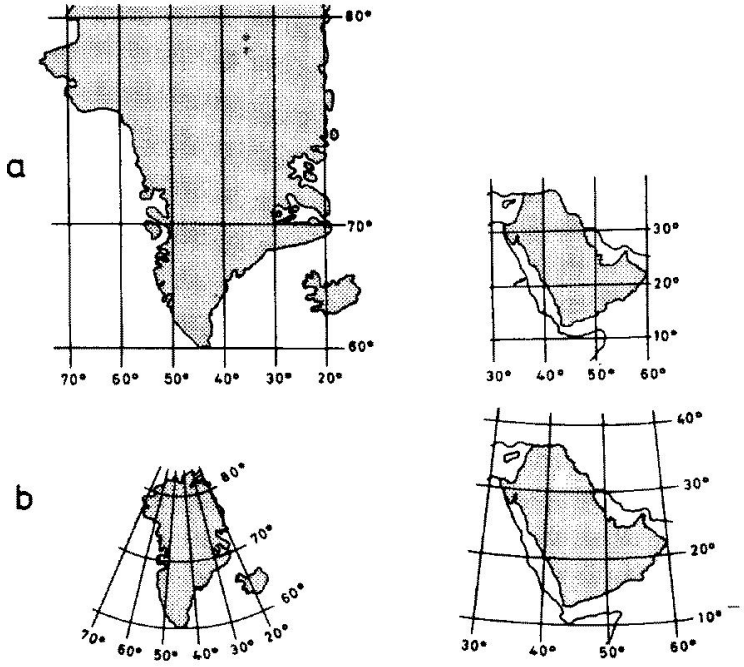

Figure 1.1: The effect of two projections (a, b) on maps of Greenland and Arabia. Area sizes are preserved correctly in (b), whereas (a) is a standard Mercator projection. Taken from [34], by permission.

With respect to map labeling, a rule has to be observed that corresponds directly to the type of projection chosen: Labels have to be arranged in parallel to the degrees of latitude that may have the shape of a line for cylindric projections or the one of an ellipse for conic or azimuthal projections. See [13] for details.

[7] A few of them that have some practical importance can be found in [11], with their mathematical background elucidated.

It is of course easier to deal with rectangular labels only than with non-specified geometric figures, but there is no restriction that hinders the application of the same methods to them. So it may be enough to solve map labeling problems with axis-parallel rectangles at first, leaving the more general task for further research.

1.1.3 Basics of Map Label Arrangement

As the section 1.1.1 showed, it is important to have some knowledge about cartographic labeling preferences before starting to label any map. An article that is cited in every publication that deals with automated map labeling is [13] again. Imhof summarizes the rules that have to be observed during name placement and later work suggests only minor changes. He states six general requirements for a good label placement I reflect here in a summarized form:

- Readability: chosen typeface, letter size and color on the one hand and positioning of the text on the other should support perception.
- Definite attachment: it should be clear which object belongs to a text.
- Avoidance of overlaps: the other map elements should not be covered by a label.
- Spatial integration: the label, looked on as a graphical shape without textual information, should help to clarify the spatial context of the designated object.
- Site Identification: the chosen font should be a hint to the type of the labeled object.
- Overall aesthetics: the labels should not be spread over the map symetrically, but name clusters do also look disagreeable. This takes effect on generalization as well as on the placement.

Cartographers arrange the named features in three groups (see [13]), ordered by their topographical dimension:

point features: cities, summits, but also area features on small scales

linear features: rivers, streets, borders

area features: mountains, islands, countries, lakes

Whereas the notations for point and linear features are arranged aside the object, they are written into the described object in case of area features (except for the case

the area feature is sized too small to place the label inside its boundaries; then, it is treated like a point feature).

Depending on the length of a linear feature, it may be necessary to label it two or more times what seems to evoke a higher degree of complexity on the labeling. On the other hand, the label of a river or road can be easily moved outside an area that is densely filled with other labels. The label of an area feature can be placed nearly anywhere inside the feature. That can not be done with a point feature label, it has to stay near its feature because otherwise the attachment of label and feature can not be identified uniquely. Solving conflicts between many point feature labels may therefore be much harder than dealing with conflicts of line or area features. An algorithm that labels point features correctly can be expected to work on line and area features, too. For that reason I concentrate on point features in this work.

Different authors evaluate the order of preferred point label positions and present several possibilities. To begin with the most cited one, I refer to [13] once again:

Imhof expresses his opinion about good and bad placement solutions within a set of merely fuzzy rules. The position that is to choose if the object is far-off other map elements is on the right side of the site and raised a little bit (he does not state how much this little bit is) against it. The label should not be aligned with the site because that looks unfavourable. If other objects cover this best place, the label has to be moved – eventually moved to another direction. At first it should be put to the left side, then over the top and at last under the bottom. The latter distinction is done because the bottom of a names skyline is much more regular than the top. Few characters descend under the font baseline but many ascend above the height of a 'n' or 'o' what is the standard height for lower case letters (see [20]).

In contrast to Imhofs viewpoint as a cartographer, Pinhas Yoeli searched for labeling rules that could be transferred directly into an algorithm (see [37]). Bounded by technical restrictions[8] he operated on a grid with each point feature exactly covering one square grid unit. The label has to be set to one of ten specified positions, the first the most preferable. Two of these positions have been invented due to the fact that centering of a label with an even number of letters is not possible relative to the grid space of the feature (positions 7 and 8) which is always one and therefore odd.

The last two positions are not strict but meet the condition that the label must be adjacent to the feature. It can be moved along the top or the bottom of the designated site, but its surrounding rectangle has to touch the one of the point feature.

Yoeli also designed positioning rules for large point features and areas, but as these map items are not used in this work, I do not describe them in detail. Instead of this, I would like to express my opinion about the relationship of his article to the one of Imhof: The two priorization models are not equal but very similar. The

[8] He did not have access to another but a grid-oriented output device.

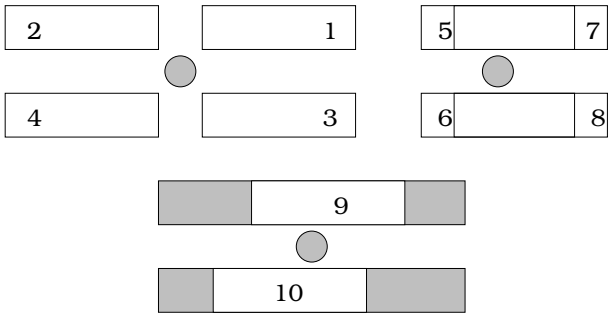

Figure 1.2: The priority order proposed by Pinhas Yoeli. The filled boxes around the lowest circle mark the range of allowed positions.

difference is that Yoeli has put the cartographic rules in concrete terms to be used in a placement algorithm.[9]

The importance of Yoeli's work is pointed out by Chyan Victor Wu and Barbara P. Buttenfield in [36]. They put his priorization model in position of being the ancestor of many algorithms for automated label placement (for example [9]) and try to answer the questions of suitability and practical importance of the model for automated map name placement. The road maps evaluated during this study do not comply with Yoeli's model but follow rules that seem to depend on the particular publisher and the purpose the map had been designed for. Another reason for this may be the absence of any distinction of different feature types in the model and therefore the non-observance of the many rules for special cases that have been proposed by Eduard Imhof. As a consequence, Wu and Buttenfield request an algorithm that is able to realize special placement situations and reacts to them while also taking the aesthetic balance into account.

1.2 Algorithms by Others

The idea of solving map labeling problems with an Evolutionary Algorithm has been originated by Hans-Paul Schwefel as he encountered an article about the map labeling algorithm by Alexander Wolff and Frank Wagner in a popular German newspaper [26]. Therefore, I started my search for other methods in this field with the diploma thesis of Alexander Wolff [35]. Following his references, I found a

[9] In my opinion there are no essential differences between Imhof's fuzzy rules and Yoeli's practical priorization model, although some are stated in [36]. The original version of Imhofs paper ([13]) that was printed in German does not allow the conclusion that

> "Imhof prefers the position directly under the symbol rather than over the symbol."

(cited from [36], page 11).

very good introductory paper [4], dealing with a comparison of many opimization algorithms applicable to map labeling that has been published by a research group from Harvard. Thereby they estimated the value of an own labeling method based on Simulated Annealing – a technique that has much in common with Evolutionary Algorithms – by a comparison with others.

From the many different approaches mentioned in the accessible publications I chose three that seemed to be of particular importance for the development of the desired labeling algorithm. They are presented in chronological order now.

1.2.1 The Energy Minimization Algorithm of Stephen Hirsch

The most striking thing about this method (it has been published in [9]) is the use of a continuous placement model opposed to most other suggestions that prefer a set of four or more possible positions for a label. Hirsch moves the names around a point feature on a circle with a defined radius. New positions are computed using a repulsion model that reminds of a simulation of equally charged particles. If a label closes in the bounding rectangle or circle of any other object, a repulsive force is induced, driving them into opposite directions. A label that is overlapped by more than one other object moves into the direction that is given by the vectorial addition of all conflicting objects (sites cannot be moved). After the label motion has been computed, all label positions are changed at the same time. The system is now hopefully moving into a state of minimal forces and thus minimal kinetic energy, as it can be recognized to happen in nature.

This first overlap avoidance method is not always capable of leading a label out of a conflict area. Especially areas of high feature density often bring up deadlocks; each label has to cross the other one to get to an uncovered area. Therefore, a second kind of movement is used from time to time: at once, the labels are put into the position the computed directional vector is pointing to.

In consequence, the algorithm is a specialized version of a gradient search with additional heuristics. Like other gradient search methods it is heavily in danger of getting stuck in a local optimum.

1.2.2 The 2-SAT Algorithm by Frank Wagner and Alexander Wolff

Frank Wagner came across a labeling problem that had been formulated purely mathematically. With Michael Formann he showed its $\mathcal{NP}$-hardness and developed a very fast approximation algorithm that has a time consumption of about $\Omega(n \log n)$ [32].

The prerequisites of this algorithm called **A** are:

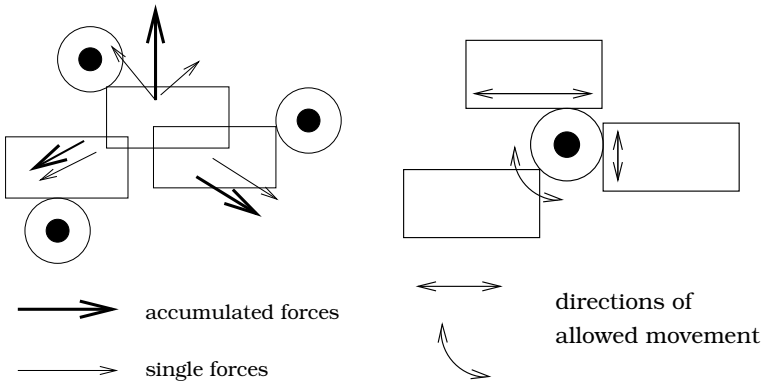

Figure 1.3: The forces resulting from conflicts (left) and the label moves allowed in the algorithm of Stephen Hirsch (right).

- Each label is an axis parallel rectangle and has the same size as the others.
- The label is placed in one of four possible positions: one edge has to meet the labeled feature.
- Cartographic preferences can not be taken into consideration, all four possible positions must have equal scores.

Since the optimization is started without a default box size, it is the task of the algorithm to find a valid labeling – there should be no overlap – with label measurements increased as far as possible. **A** was found to guarantee a label size of at least 50 percent of the optimum what proved to be useless for real world applications – the original labeling problem had been invented by the municipal authorities of Munich who wanted to manufacture technical maps of their ground water drillholes.

Therefore, Wagner developed algorithm **B** with the help of Alexander Wolff which dealt with the problem of chosing two of three or four possible label positions by use of a heuristic. The remaining task is equivalent to a 2-SAT logical decision problem that can be solved efficiently. It has been documented ([35] and [33]) that **B** is able to find a labeling close to the optimum if all of the constraints stated above are met. But, in my opinion, it will be difficult to apply this method to more complicated real world problems with labels of different sizes and more cartographic requirements.

The algorithm is reachable via Internet at the URL

```
http://www.inf.tu-berlin.de/map-labeling/
```

and may be used for comparison with other labeling tools because it provides possibilities of data exchange.

1.2.3 The Simulated Annealing Algorithm of Jon Christensen, Joe Marks and Stuart Shieber

Another very different approach has been proposed by an American team of researchers who tested the suitability of the simulated annealing algorithm for the MLP (first publication [3], presented to the public with [4]). They did so after proving that the simple variants and also many of the much more complicated problem instances of map labeling that can be generated by increasing the positionings' degree of freedom are not solvable efficiently if $\mathcal{P} \neq \mathcal{NP}$ (see [18]). The proof even holds if the discrete set of positions that can be used for one label is extended to an unbounded number (as it has been used by Hirsch in [9]).[10]

A great advantage of the simulated annealing method is the ability to jump out of a local minimum. This is a property it shares with all the other stochastic optimization techniques[11]. The quality of a generated solution is only determined by evaluating a fitness function that returns a numerical value. So, if the aims or the priorities of these aims change, it suffices to adapt the fitness function. This has already been done by the group. In the meantime, line and area features have been added to the maps and are handled by combining the values of eleven different measures. These are added after multiplication with a particular weight factor that is very high for overlap measures and lower for positioning ratings [5].

The placement of the point feature labels is done by chosing one of a set of nineteen valid positions, each one tagged with a penalty value from the interval $[0, 1]$. A new labeling is conceived by changing only one label position and evaluating the fitness function. If its value was improved, the labeling is accepted, otherwise it is rejected with a probability that is growing during the optimization run.

In my opinion, this algorithm is the only one of the three presented here that can be integrated into a computer system that is used to design and manufacture real world maps.

1.3 Marking a Destination Point

Since the basic cartographic knowledge that has to be taken into consideration and the algorithmic approaches that have already been tried to solve Map Labeling Problems have been presented, it is about time to summarize the aims of this thesis now.

[10] This is of importance for the modeling used in this thesis, too.

[11] Evolutionary Algorithms also belong to this category.

The most important task is to decide on the practicability of a method based on an Evolutionary Algorithm for solving a MLP and its benefit for use in map manufacturing later on. This method has to be:

general: Although the problem has to be restricted so it can be coped with in a limited amount of time, the designed algorithm itself must not be bound to these restrictions.

extensible: It must be possible to integrate further rules derived from cartographic experience into the algorithm without changing its internal structure.

However, some prerequisites have to be made to limit the amount of work that arises from the manifold possibilities of different map types and map elements.

- No other than point features are used. Line and area features have to be added if the map labeling algorithm has turned out to be practicable. The evolution of the simulated annealing algorithm described above (see 1.2.3) from [4] to [5] proves this to be possible.
- No site can have more than one label. There are of course features that are usually shown with more than one text field. For example, the altitude and the name of a mountain are both put into a map, the former with higher priority. This must be considered during algorithm design but does not have to be implemented.
- Used text sizes are not subject to optimization but are regarded as input that has to be provided from outside the algorithm.

Furthermore, efficiency is no primary task while implementing any code and is only dealt with to estimate the overall time consumption for different problem sizes. Though it is good to have a rough impression about the time consumed by the optimization, raw cpu-second measurements are of questionable value because hardware becomes faster and faster these days. Even the implementation of a method using another language or tool can speedup or slow down any algorithm enormously.

Chapter 2

Modeling the Problem

2.1 An algebraic Model for Quality Judgement

Based on the requirements for a good label placement that have been identified in section 1.1.3, I try to generate a mathematical system now that enables a quality verification of computed solutions. Some of these aims cannot be subject to an optimization, however. In which way could the remaining goals be expressed to have them handled by an algorithm?

- Readability can only be improved as far as the positioning is concerned. Several penalty models or priority recommendations have been applied to integrate this aim into labeling algorithms. Typeface, letter size and font color have to be provided by the person operating the optimization who should have cartographic knowledge about the suitable settings.
- Definite attachment means that the label should be closer to its site than any other label. Perceptability is supported if all labels without placement conflicts are put at the same position relative to the site. Besides that, sophisticated use of different colors may support fulfilling this task as it does for readability, referring to investigations of Henry Beller [2] and others. However, this is not subject to optimization but to application of cartographic experience.
- Overlaps have to be tested for any generated solution. The labels must not cover other labels, sites or borders. For usability in practice, it is also important to check for overlaps of other map components like relief shadings.
- Spatial integration is left out for reasons of simplification.
- Choice of the different label fonts for site identification can only be provided

from outside the algorithm, it just has to use the settings during the evaluation of the quality of a labeling.

- It is very difficult to judge about the overall placement impression automatically. Some statistical measures can be gained from the name distribution on a map, but the derivation of an aesthetic criterion from this data is at least questionable. Therefore, it may be omitted.

With this distinction of optimization tasks and cartographic settings supplied from outside the algorithm, it should be possible to build a mathematical model of the MLP.

2.1.1 Basic Entities and Measurements

Entities that have to be taken into account from the abstract viewpoint of an algorithm designer are: sites, labels and bounds. Another useful entity that may serve as a basic unit while constructing more complicated objects is a point. It is defined as a position on a two dimensional plain area with a cartesian coordinate system that can have values from $\mathbb{R}^2$. Points are mostly indicated by the standard letter p_i, the subscript holding an identification number, their two constituents are named x_i and y_i, corresponding to the appropriate cartesian axis. A point can be member of none, one or more objects (the latter case means that the objects overlap). The distance of a point $p_i(x_i, y_i)$ to another $p_j(x_j, y_j)$ is as usual

$$d_p(p_i, p_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{2.1}$$

with d_p indexed by p as a distinctive mark to other distance functions.

As a base for the construction of objects later on, I use restricted point sets P_i upon which a function $c(P_i, p_j)$ is defined that decides whether a point p_j is contained in the set.

$$c(P_i, p_j) := \begin{cases} -1 & : \quad p_j \in P_i \\ 1 & : \quad p_j \notin P_i \end{cases} \quad , \quad P_i \quad \text{is restricted set of points,} \quad p_j \quad \text{is point.} \tag{2.2}$$

A restricted set of points P_i should have a measurable distance to a single point, too. It is set to:

$$d_s(P_i, p_j) := min\{d_p(p_g, p_j) * c(P_i, p_j) \mid p_g \in P_i\} \quad , \tag{2.3}$$

wherein min denotes the infimum of a set it is included in itself.

A point set that is a simply connected and restricted region is called an object O_i. Its limitations ensure the existence of a border which is formed by the set of all points meeting the condition that every ε environment around them must contain points inside and outside O_i. The set of border points of an object O_i is referred to as $B(O_i)$. As the object is restricted, so is the border. Every object has a definite area it covers on the plane. This measurement is generated by the function $a(O_i)$.

The objects are intended to turn out to be map features later, so it is necessary to define a specialized distance function on them that is able to measure overlaps, too.

Overlap and distance of two objects can be expressed with a single function that generates positive values if the objects do not touch but negative values in case of overlap. To simplify the formula, we require $a(O_i) \leq a(O_j)$. This condition removes the possibility of O_j being fully contained in O_i but not vice versa. As a consequence, there has to be at least one point in $B(O_i)$ (border set of O_i) that is also in O_j if the two objects overlap. If $a(O_i) > a(O_j)$ is true on the contrary, $d_o(O_i, O_j)$ is conceived by swapping the objects. The distance function can now be defined as:

$$d_o(O_i, O_j) := \begin{cases} min\{d_s(O_j, p_g), \quad p_g \in B(O_i)\} & : \quad a(O_i) \leq a(O_j) \\ d_o(O_j, O_i) & : \quad a(O_i) > a(O_j) \end{cases} \tag{2.4}$$

This definition can be applied to obtain the distance an object has to itself what is equal to the distance it has to another object that contains excactly the same set of points. The result measures the greatest distance of two points lying on the border and is smaller than zero to indicate an overlap.

If all map elements are circles or rectangles with bounds parallel to the cartesian coordinate axes, the distance measurement can be simplified without loss of quality. This can be done by reducing the points of the bigger object checked against the border points of the smaller object from the full border set to the ones having the same x or y coordinate value (they are touched by the same horizontal or vertical line), if any exist. Under condition $a(O_i) \leq a(O_j)$ (in case of $a(O_i) > a(O_j)$ i and j have to be swapped), the distance measurement can be computed using equation 2.5 instead of 2.4.

$$d_{po}(O_i, O_j) := \begin{cases} min\{d_p(p_k B(O_j), p_g) * c(O_j, p_g)\} & : \quad \exists \quad p_g \in B, p_k \in B(O_j) \mid \\ \qquad\qquad (x_k = x_g) \vee (y_k = y_g) & \\ min\{d_s(B(O_j), p_g), p_g \in B(O_i)\} & : \quad else \end{cases} \tag{2.5}$$

The map itselft consists of an object that represents its spacial expansion $\mathcal{M}$ and contains two additional sets of objects, a set of m sites $S = \{s_i \mid 1 \leq i \leq m\}$ which are not moveable and a set set of r labels $L = \{l_j \mid 1 \leq j \leq r\}$ which are. To enable identification of one or more labels that belong to a feature, a function b is defined

that yields the numerical value one if one of the operands is a label of the other, zero otherwise.

$$b(O_i, O_j) := \begin{cases} 1 & : \quad (O_i \; is \; label \; of \; O_j) \vee (O_j \; is \; label \; of \; O_i) \\ 0 & : \quad \neg(O_i \; is \; label \; of \; O_j) \wedge \neg(O_j \; is \; label \; of \; O_i) \end{cases} \tag{2.6}$$

Apart from monitoring conflicts between labels, the map object should always contain all site and label objects. Therefore, the distance function must be modified slightly to recognize conflicts with the map border:

$$d_m(O_i, \mathcal{M}_j) = max\{d_s(B(\mathcal{M}_j), p_g) * c(\mathcal{M}_j, p_g), p_g \in B(O_i)\} \quad . \tag{2.7}$$

The function d_m generates negative values if the object O_i is fully included in the map object $\mathcal{M}_j$, positive ones if it is not.

2.1.2 Testing Different Map Requirements

As stated above, there are some conditions a good labeling has to meet which can be tested directly by automatic evaluation of mathematical equations. These will be formulated now.

Overlaps

Labels are the only objects of a map that are moveable during the labeling process. So every conflict test that makes sense must include at least one label. The objects that can cover the desired area are: sites, other labels and map borders. By the use of d_o (or d_{po} in case of the restricted problem) for all possible combinations of labels with other labels or sites and d_m for the map borders, all overlaps can be detected since they are indicated by negative function results.

The distance requirements can even be made more demanding if the objects contained in a map are expected to have a minimal spacing greater than zero (that would ensure the avoidance of overlaps) but equal to any value that seems to be appropriate from the point of view of a cartographer. I assume that it is not necessary to guarantee a minimal distance to the map borders as far as they are not crossed. A difficulty that may arise if some space next to the border has to kept free of labels is that the features being located in the forbidden area can not be labeled properly.

Different features may have different minimal distances because the symbol sizes used to indicate their positions are usually chosen according to the importance of each feature. Font measures and typefaces are adequately set (see [20]) and make

up a great variety of label sizes on most maps. Therefore, it is recommendable to express the distance requirements by functions rather than by constant values. These are named $\Delta_s(O_i)$ for other sites and $\Delta_l(O_i)$ for other labels tested. If O_i is itself a site, the value of $\Delta_s(O_i)$ is meaningless because the features can not be moved on the map, any conflict between them has to be accepted.

The rules that have to be obeyed for any movement of the label l_t of a map M with a set of labels L and a set of sites S to pass the test for overlap are thus:

- The label must keep a distance $\Delta_s(l_t)$ to any site s it is not label of.
- The label must keep a distance $\Delta_l(l_t)$ to any other label l.
- For every site $s_v \in S$ the label is not attached to, its minimal distance $\Delta_l(s_v)$ must be kept.
- For every other label $l_u \in L, u \neq t$, its minimal distance $\Delta_l(l_u)$ must be kept.

Definite Attachment

The requirements that guarantee a certain perception which label is attached to which feature are twofold. First, the connected label must be put into an distance interval that is bounded by a minimum and a maximum distance $\delta_{min}(s_i)$ and $\delta_{max}(s_i)$, depending on the site s_i. This condition also guards against an overlap of the site and its own label. Note that these values can vary for different sites, therefore they are also given as functions.

Second, the other labels have to be kept away from this site farther than the attached one. This can be achieved by setting $\Delta_l(s_i)$ to a value greater than $\delta_{max}(s_i)$.

Positioning relative to the site

The cartographic preferences for relative positioning of the label around its site can be expressed with a quality function $q(s_i, d, \theta)$ that returns a penalty from the interval $[0, 1]$, small for good positions (d, θ) and near to one for bad ones. It takes a site s_i, a distance d and an angle θ as argument. The angle of a pair of objects relative to the x axis is computed by the function $\theta(O_i, O_j)$. This can be done, for example, by searching two points $p_1 \in O_i$ and $p_2 \in O_j$ with minimal distance of all possible combinations and to determine the angle of a straight line connecting these points to the x axis.

While constructing $q(s_i, d, \theta)$, it should be considered that the minimum and maximum distances of the attached label are depending on s_i. If a unique minimum of the quality function exists, q does also imply an optimum distance $\delta_{opt}(s_i)$ that has to be within $[\delta_{min}(s_i), \delta_{max}(s_i)]$.

Overall Placement

The aesthetic demand for avoidance of label clusters *and* of equally distributed labels can not easily be transferred into a rule for quality decisions. These two aims are opposite to each other and can not be fulfilled at the same time. Name clusters whose contents are moved outside to improve their perceptibility are of course more equally distributed than before. However, this conflict may be weaker than it seems to be. I think that Imhofs intention while inventing this rule has been to warn cartographers of two extreme possibilities. So it may suffice to realize very strong clustering or very symmetrical maps. Everything between these two labeling solutions is to be accepted.

The name clusters of a map can be detected if the local density of objects is measured wherever a site resides. This can be done by taking the size and the distance of the neighbour objects (which can be sites or labels) into account. Previously defined distance functions 2.4 or 2.5 can be used to obtain meaningful numerical values even in case of conflicting objects. The local feature density function proposed now measures the object density at the location of a site s_i:

$$fd_l(s_i) := \sum_{1\le k\le r} \frac{a(l_k)}{(d_o(s_i,l_k)+1-d_o(s_i,s_i))^2} + \sum_{1\le j\le m} \frac{a(s_j)}{(d_o(s_i,s_j)+1-d_o(s_i,s_i))^2} \quad . \tag{2.8}$$

d_o can be replaced with d_{po} if the requirements for the equation 2.5 are fulfilled. As the monitoring of movements is the primary aim while using this function, the denominators of the sum terms which refer to the object distances are squared. The additional term $1 - d_o(s_i, s_i)$ is used to push the distance values upwards into the interval $[1, \infty[$ because $d_o(s_i, s_i)$ is exactly the minimal outcome of $d_o(s_i, l_k)$. If the numbers used to indicate map positions are scaled down to values much smaller than one, they should be rescaled before they are inserted into the equation. Otherwise the received value will almost not change although the labels are moved.

The mean value of the local densities of all sites may be appropriate to look for clusters in the map. It is now referred to as local feature density of a map M_i. Since the distances are squared in equation 2.8, the obtained value will increase dramatically if clusters arise. These labelings should be avoided.

$$fd_l(M_i) := \frac{\sum_{k=1}^{m} fd_l(s_k)}{m}, \quad s_k \in S \quad \text{of} \quad M_i \quad . \tag{2.9}$$

The standard deviation of all local densities measures the symmetry of a labeling. Its value is small if the symmetry is strong, large if the map density varies much.

$$sym(M_i) := \sqrt{\frac{1}{m}\sum_{k=1}^{m}(fd_l(s_k) - fd_l(M_i))^2}, \quad s_k \in S \quad \text{of} \quad M_i \quad . \tag{2.10}$$

At this point I must admit that the composition of the chosen local density function is somewhat arbitrary. On the other hand, the results are intended to be compared with the ones of other maps and do not need to have a meaning by themselves.

2.2 A Composed Fitness Function

The fitness function that is needed for optimization runs later on can be constructed of the named constituents, except for the overall placement criterion which is to fuzzy to include without getting some experience first. The remaining conditions have to be ordered by priority which can be done by multiplying their returned values with different weight factors.

w_{ls} determines the importance of an overlap a label has with a site.

w_{ll} is the adequate value for label to label conflicts.

w_{lb} expresses the relevance of label to map border conflicts.

w_a is the attachment weight that binds the labels to their own sites.

w_p corresponds to the importance of the positioning preferences.

The weights have already been sorted by priority, highest first. It is obvious that the overlap weights must be higher than the positioning weight: a map without any overlap but with some labels that do not comply with an aesthetic criterion may be accepted though being ugly. If the labels are placed nicely on the other hand, but this forces an overlap to arise, the map will be rejected by every cartographer.

By combining the equations that have been built for overlap, attachment and placement control, a fitness function can be constructed that offers a great variety of possible changes. Besided the use of different weight or distance values, even cartographic rules that have been not taken into account may be integrated, due to its modularity. To simplify handling and comparison of results, the fitness function is bound to the requirement that the best result is zero in any case. Furthermore, a lower value indicates a better solution than a higher one (minimization). Thus, negative values of the constituent terms have to be cut off. Therefore, the function

$$cut(x) := \begin{cases} x & : \quad x \geq 0 \\ 0 & : \quad x < 0 \end{cases}, x \in \mathbb{R} \tag{2.11}$$

is used. The fitness of a whole map $M_i = (\mathcal{M}, S, L)$ can be determined by evaluating $fit(M_i)$, which is built up in the following way.

$$\begin{aligned} fit(M_i) = \; & w_{ls} \sum_{j=1}^{r} \sum_{k=1}^{m} cut((\Delta_s(l_j) - d_o(l_j, s_k))(1 - b(s_k, l_j)) \\ & + w_{ls} \sum_{j=1}^{r} \sum_{k=1}^{m} cut((\Delta_l(s_k) - d_o(l_j, s_k))(1 - b(s_k, l_j)) \\ & + w_{ll} \sum_{j=1}^{r} \sum_{1 \le k \le n, k \ne j} cut(\Delta_l(l_j) - d_o(l_j, l_k)) \\ & + w_{lb} \sum_{j=1}^{r} cut(d_m(l_j, \mathcal{M}_i)) \\ & + w_a \sum_{j=1}^{r} \sum_{k=1}^{m} cut((\delta_{min}(s_k) - d_o(s_k, l_j))b(s_k, l_j)) \\ & + w_a \sum_{j=1}^{r} \sum_{k=1}^{m} cut((d_o(s_k, l_j) - \delta_{max}(s_k))b(s_k, l_j)) \\ & + w_p \sum_{j=1}^{r} \sum_{k=1}^{m} q(s_k, d_o(s_k, l_j), \theta(s_k, l_j))b(s_k, l_j) \end{aligned} \tag{2.12}$$

During the computation of $fit(M_i)$, several parts of it can be simplified if the attached labels of a site can be retrieved efficiently by an algorithm without enumerating all possibilities. Furthermore, the calculation of the two double sums using w_{ls} and the two multiplied by w_a, respectively, each can be done at the same time. Apart from d_o, d_{po} can be used for measuring label or site distances if the MLP has been restricted to axis parallel (isothetic) objects.

2.3 Estimation of Problem Complexity

It is unquestionable that the integration of more features into a map will make the MLP become more difficult – otherwhise it could not have been proven to be $\mathcal{NP}$-complete.

Another important property of the map is its feature density. As font sizes and therefore label boxes grow, a labeling without overlap can become impossible (that follows from experience with the algorithm presented in 1.2.2). So the density is measured by comparing the sum of the areas all sites and labels cover with the area they have to be placed on. It is called *global* in contrast to the local feature density that has been defined in 2.1.2.

$$fd_g(M_i) := \frac{\sum_{j=1}^{m} a(s_j) + \sum_{k=1}^{r} a(l_k)}{a(\mathcal{M})} \tag{2.13}$$

Hence, a measurement composed of these two map attributes may help to estimate the level of difficulty to determine a valid labeling. The key to this would be to evaluate the distribution of potential conflicts. Any area that can be labeled without regard to the rest of the map can be separated and optimized on its own. But

if the global feature density grows due to a change of the used font sizes, the areas without potential covering become smaller until it is not possible any more to separate them.

Some maps, however, contain great areas without any site and hence any label. These reduce the value of the computed global density and should be removed from the map before computing it to get a proper value. But this induces the problem of recognizing spacial map structures to the algorithm that may be of similar complexity than the MLP itself. Especially island maps destroy the value of this measure because they are filled with nothing but sea to a high extent.

Due to all the difficulties that arise while striving for a measure that can be used to make a forecast on the complexity of a concrete MLP, a hypothesis is left out for now. However, it may be possible to find a rule examining the data of several optimization runs.

To compare the different optimization runs and to obtain a rule for difficulty estimation of a problem, a quality criterion for the results of an optimization is needed. It should measure the relative convergence rate without respect to a concrete problem instance. I assume that the fitness function is constructed in a way that better solutions result in smaller function values and zero is the optimum.

$$conv(fit_s, fit_e, s, e) := \frac{log(fit_s) - log(fit_e)}{e - s} \tag{2.14}$$

The values that have to be known are a starting point s, given by the number of a fitness function call that yields fit_s, and an endpoint of measurement, indicated by e what is the number of the fitness function call that returned fit_e.

The use of a logarithmic scale is intended to match the typical progress appearance of an Evolutionary Algorithm (LIT). From this point of view the result of 2.14 reveals more information than a linear definition would have done because it counts the decades traversed from s to e divided by the number of fitness function calls needed in doing so.

Chapter 3

Getting the Problem Solved

This chapter makes the heart of the project come into life: a computer program that runs an Evolution Strategy[1] on a MLP. It begins with a basic introduction on *Evolutionary Algorithms* what has been established by now as a generalizing term including several optimization methods which are imitating the natural evolution.

3.1 Evolutionary Algorithm Primer

L' imagination se lassera plutôt concevoir
que la nature de fournir.

Blaise Pascal

Trying to translate Pascal's words I would like to put it like this: Imagination is exhausted more easily than nature. What has this statement got to do with Evolutionary Algorithms? You will probably anticipate that evolution is related to nature somehow. In fact, evolution is seen as an essential attribute of life on earth nowadays. It is the possibility to adapt to a very complicated, moreover even dynamic environment and to build up more and more sophisticated systems we call creatures. If we look on survival in this planetary environment as a problem, nature makes the best use of the existing resources and "discovers" solutions we are often surprised of. [2]

Why are we? Because we would not have thought of them as being possible or because they are so far away from normality that we could not even have imagined

[1] Further on, Evolution Strategy is abbreviated to ES.

[2] For example, there exist some species of bacteria living near so called "black smokers", volcanic spots in the deep sea. The environment they are adapted to appears quite infernal: temperatures vary around 100 degrees of Celsius, the water contains a horrible portion of acid, and the pressure would crush any mammal instantly.

them. This is an important point and one reason for the success of Evolutionary Algorithms.

Although we know no will or person who controls the natural evolution, it can be understood as an optimization that is still in progress today. So if it was possible to transfer this process into a computational environment and to let it solve technical or mathematical problems also, we could possibly get solutions our imagination withholds from us.

During sixties and seventies, a number of researchers (Fogel [17], Ingo Rechenberg [15] and Hans-Paul Schwefel [7], and John H. Holland [10]) had the idea to transform the mechanisms of evolution into optimization algorithms that could cope with numerical problems.

To understand the functionality of these algorithms we have to identify the basic principles of evolution which are recombination, mutation and selection, according to the theory of Charles Darwin.

The recombination has got two important aspects: first it is a way to preserve life from total disintegration that happens to all kinds of living beings we know after they have been touched by death. Second, the children are no clones of their parents, they are both similar and different. Apart from other mechanisms that take place during the formation of a new individual, recombination mixes up the genetic code of the parents to create a new combination of the existing constituents. Even if there was no possibility to change the basic parts, numerous different mixtures could arise this way. So recombination builds up new individuals and composes them as a – probably unique – genetic mixture of the parents.

Mutations could be described as tolerated accidents while working on genetic information. The molecules building our genetic foundation are very large and complicated as shown by popular pictures of the DNA. So they are of course subject to errors that can occur during the copying and assembling processes which are necessary to build new cells. Most of these errors are caught by repair mechanisms, but some are not. In this case an alteration of the genetic code has happened that could most probably not have been achieved by recombination. Mutation is therefore the possibility to create new genetic code that has not existed before.

Selection cannot be detected in nature as easily as recombination and mutation, but has at least the same level of importance. It is the mechanism that makes evolution a dynamic process because it ensures that some genetic code is disappearing what is a very important condition for development. When being invented as a theoretical concept in the last century, the term ”survival of the fittest” became widely known. That does not mean that only the individuals with the highest physical strength or best health survive, but the ones that can cope well with the difficulties of their local environment.

Long enduring isolation is an exceptional state in nature but gives us a chance to investigate this adaption of life to a very limited region as Darwin did at the islands of Galapagos. Competition is the usual case and "selects" the successful individuals what means the successful species in the long run.

The Evolutionary Algorithms should solve problems that can be expressed by mathematical formulas, but they use the same three principles. Their "world" is the search space of a given problem and the individuals are instances of problem solutions, defined by genetic code consisting of the parameter values that indicate the individuals' position in the search space. The only condition that must be fulfilled by the problem is that the quality of a generated solution can be measured and expressed by a number that is called "fitness".

In a few words, the algorithms do the following:

1. A start population of problem instances is generated.
2. The generated individuals are evaluated, i. e. their fitness value is calculated.
3. The chosen termination criterion is checked (does the best generated fitness value meet the expectations?).
4. Recombination is done: the offspring of the actual individuals is generated by mixing their genetic code.
5. The genetic code of the offspring is mutated.
6. The offspring is evaluated: the fitness values are computed.
7. A predefined number of individuals is selected with regard to their fitness.
8. The selected individuals are the new population. The loop is started again at step 2.

The best individual should be stored during the optimization as its result. The mutation and the recombination normally use pseudo-random numbers so the optimization is stochastic and non-deterministic.

Due to the fact that the title of this thesis only mentions Evolution Strategies which are only one of the many existing variants of Evolutionary Algorithms, further writing is dealing with them only. For information about the other existing variants you may have a look into the following references, ordered by increasing minuteness of detail: [30], [28], [29].

3.2 Preparing a Suitable Evolution Strategy

Two questions have to be answered before a detailed design for an optimization tool can be done:

1. What do the parameters look like that can be varied by the optimization?
2. What kind of Evolution Strategy should be used? And furthermore, which of the several proposed possibilities to alter the basic Evolution Strategy are appropriate?

An ES works on individuals with a genetic structure that consists of usually continuous parameters needed to evaluate the fitness function. For the MLP, this function has already been formulated in section 2.2. It induces the use of a map as an individual, and the positions of the labels relative to their sites as the parameters, representing one labeling instantiation. Some information about the locations of the sites, the label texts and the properties of the maps border must be stored, too, but is not subject to the optimization because it cannot be changed.

3.2.1 The coordinate system used for label positioning

By now, a general decision has to be taken on the representation of the label positions. Solutions that have already been tested by others are a discrete, ordered set, as it has been used in [5], or a continuous variable, indicating the angle of the connecting straight line as proposed by Hirsch [9]. Although the complexity of the optimization task is increased by this, I chose a continuous representation similar to a polar coordinate system around every site for the following reasons:

- As the label can be moved in two directions instead of just one, some conflicts may be solvable that would have produced a deadlock otherwise.
- The placement offers the possibility to generate a labeling without overlap by violating the attachment rules to a small extend. This may be decided to be admissible by persons with cartographic experience, depending on the afflicted features context.
- The basic idea of moving a label around the feature can be extended to line features, as well. These are equivalent to point features with an extremely stretched symbol, in this respect. Quality judgement on generated positions would have to be adapted because cartographic preferences are different for line features.
- The ES, described by Schwefel and Rudolph in [27], uses continuous parameters. Nowadays, various approaches are made to handle integer [23] or mixed parameters (Schütz in [19]), too. The use of a discrete position set should therefore be no problem. Nevertheless, these variants mostly evoke additional work and/or difficulty because some kind of mapping, guarding against boundaries, or invention of specialized random number generators has to be done. An ES for integer programming should be used for solving

MLPs if the "pure", continuous positioning model presented here proves to be worthless.

Chosing a polar coordinate system leads to a description of each labels position with an angle and a distance value. Hence, the number of parameters for an individual is equal to the number of labels multiplied by two. But how can the location of the label on the cartesian coordinate system of the map be derived?

In the following, I assume that every site has a circular shape and every label can be handled by a rectangle surrounding its character boxes. This is no general restriction but simplifies the development of an adequate measure. For other than circular sites, it has to be modified to preserve the meaning of angle and distance parameters. However, the decomposition of the labels rectangle into its character boxes does not seem to be appropriate while dealing with the positioning, apart from overlap testing. If done, it may provoke the violation of cartographic rules. For example, label and site should not look as if they were arranged on the same line. This condition cannot be tested properly if the positioning is done with regard to the size of the last characters surrounding box only[3].

Since a label has got some areal expansion, its position can not be detected by simply locating a substitute point with distance and angle given. The label could be moved over the site in this case although the distance has not been changed. If the actual nearest corner of the label is used as attachment point, it can be rotated properly around the site, except for the case it crosses a horizontal or vertical line that traverses the site. This special situation has already been dealt with in [9]. In opposite to the solution presented there, I propose to integrate the four position ranges with a tangential contact of the label with the top, bottom, left or right egde of the site into a modified angle definition.

Therefore, a standard distance has to be used to determine the portion of the angle describing each of the two movement types. I have used the minimal attachment distance δ_{min} for this, but different settings would be possible, too. The circle is cut into four pieces then and the four lines representing the labels height and width are inserted, keeping the length relations. The total length of the obtained rounded rectangle is four times the quarter circle (that is $2 * \pi * \delta_{min}$) plus two times the label width and height, respectively. Every position of circular and tangential movement can be reached now within one full rotation which may have a different length for every label.

The distance is measured taking the appropriate corner in case of a circular and the nearest point of the adjacent label edge in case of a tangential position. This complies with the definition of equation 2.5.

[3]On the other hand, the cartographic rules may be observed with a seperate test for each. Nevertheless, these are not included in the fitness function designed in section 2.2. The quality function $q(s_i, d, \theta)$ is intended to handle all positioning evaluation.

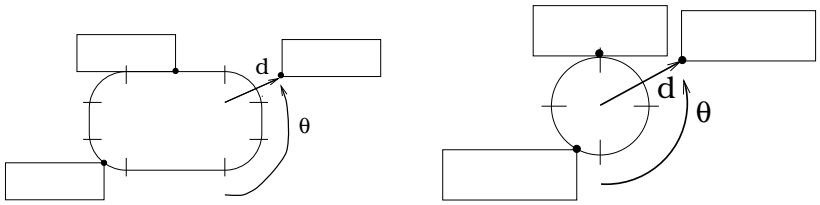

Figure 3.1: Internal representation of the label position using the modified polar coordinate system (left) and the corresponding position on the map (right). Every permitted position can be characterized with a distance d and an angle θ.

3.2.2 Choosing a subset of different ES techniques

The wide-spread variants of different Evolution Strategies based on continuous object variables have been arranged into a system in [27]. Thereof, I chose some operators that seemed to be appropriate for solving a MLP while others have been left out. However, the basic structure of the described ES has not been changed.

At first, the constituents of an individual have to be determined. The object variables are the coordinates of the label positions, two for each label, $n = 2 * r$ with r being the number of labels, for the entire map. The number of mean step sizes n_σ is always set equal to n. A reason for this is the higher degree of freedom allowed during the movement of each label. Some of them can be permitted to be nearly frozen at their position while others jump around their sites to find an uncovered location.

Inclination angles which are used to express correlations of the mean step sizes have been totally left out. The main reason for this is the enormous number of angles that needs to be used for bigger maps. As n_σ always equals n, the resulting $n_\alpha = \frac{n^2}{2} - \frac{n}{2}$ increases much faster than n and crosses the number of $100,000$ for just 224 labels. An alternative approach may be tried with an incomplete set of correlation angles by only considering the local neighbours of each label.

The mutation is done using its simple form described in [30], τ' and τ are constantly set to 0.1 and 0.2, respectively. The recombination operator of [27] is applied with the following restrictions:

- p_r is 1, constantly (recombination is always done).
- ω can only be 1 or 3, that is global intermediary recombination and uniform crossover, respectively.
- γ is not used.

For selection, only the $\overset{\kappa}{>}$ (better than) operator is used. The standard selection op-

erators , and + are contained within $\overset{\kappa}{>}$ for $\kappa = 1$ and $\kappa \to \infty$, respectively.

The termination of an optimization run is checked by three conditions that are much less sophisticated than the ones proposed in [27]:

- A predefined number of fitness function calls $call_{max}$ has been reached.
- The best fitness function value found is smaller than a target value fit_{target}.
- The number of generations processed is equal to a limiting value gen_{max}.

The start conditions $P^{(0)}$, and therewith the start population individuals, are always set up using the method described as **Case a** what is a random placement within given bounds.

This collection of techniques is a rather small subset of the ones that have already been used in practice. If it is found to be incomplete, further methods can be added later on. However, the main expense of the implementation does not arise from the ES but the representation of maps. That is to be dealt with now.

3.3 The Map Labeling Tool

Besides the optimization, there is a number of different subtasks a labeling tool has to fulfill: the map's data is to be read from a database and the map has to be stored internally. Furthermore, a visualization is needed to watch the optimization progress and to compare the provided position rating with the visual impression of the computed outcome. The requirements of each component are summarized now:

Internal map representation: A map, consisting of several features and their labels, has to be stored. Its fitness value is evaluated with methods of conflict recognition and by use of an external positioning quality function. Therefore, conversions from and into the polar coordinate system defined in section 3.2.1 have to be made. Since the locations of the sites and labels are highly dependent on the projection, it has also to be part of this subtask.

Data import: The main emphasis lies on the processing of external data files the used maps can be built of. Furthermore, parameter files have to be interpreted.

Visualization: Maps and the external positioning quality function $q(s_i, d, \theta)$ should be shown with as much internal data as possible (conflicts, geographic bounds, local positioning quality).

Optimization: The different variants stated in section 3.2.2 have to be implemented on top of a basic ES. Computed results should be written into log files.

Analysis: Mean values and convergence rates of a statistically significant number of optimization runs have to be accumulated and transformed into input data for visualization software.

3.3.1 Preliminaries

Two questions are worthwhile to be considered before one starts designing the tool:

1. Which data sources can be utilized to produce maps that can be optimized?
2. What kind of computer language is to be used for the implementation?[4]

Searching for data that is freely accessible and easy to convert into a map, I encountered only one suitable source: the GNIS[5] database. It contains a file for each state of the U.S.A. and an additional one for every pacific or carribean territory that is administered by its government at the moment. The GNIS is a program of the *U.S. Geological Survey* to standardize the names of all features found on maps within the entire country. File processing of this archives is easy since they use the *ASCII* code and provide information about the names, geographical positions, feature class, altitude, county and cartographic source for every entry in a tabular form.

Decision on the implementation language has been far more influenced by personal preferences. I chose the new "Internet-language" *JAVA*™ which has been invented by *SUN*[6] for the following reasons:

- It is object-oriented. This can be of advantage during the design phase and may help to re-use code that has already been written.
- The *JAVA* syntax is very similar to the one of *C* what simplifies the change of languages if a newcomer has got some experience with object-oriented concepts.
- Portability is nearly 100% for all the platforms a *JAVA*-interpreter exists on. Programs can therefore be downloaded and started automatically by an *Internet-Browser* if they use an interface that grants only restricted access to system components due to security requirements.
- Many good ideas of other languages have been used while constructing *JAVA*, some of them are: automated disallocation (garbage collection) of objects,

[4] It is important to know about the language before design starts, or one could design something that can not be implemented otherwise.

[5] Geographic Names Information System, see [31]. Located at the URL: `http://mapping.usgs.gov/www/gnis/`.

[6] Released on January 25th, 1996, in its first operational revision (1.0), following a series of pre-release beta test versions.

sophisticated error handling, multi-layer modularity, strong typing, runtime metadata and persistence. For a detailed comparison of object-oriented languages other than *JAVA*, see [25].

- If the program comments have been written with regard to *JAVA*'s comment syntax, a documentation written in *HTML* can be generated automatically which reflects all invented classes and methods and connects them to each other.

On the other hand, there are some disadvantages, too:

- The performance while doing numerical computations is estimated to be only 5 - 10 % of the one of an implementation in *C*. After all, *JAVA* is an interpreted language. To build a fast optimization, the numerical parts of a program should be translated into another language.
- Only a few useful development tools exist until now, and most of them are quite expensive (in contrast to many *C*-tools which are in the public domain).

3.3.2 Fine-grained Design

The most important classes of the map labeling tool are presented in figure 3.2 which only shows constituents of the *map representation* and the *optimization* domain. These two parts are connected by the interface `Evolvable`[7]. Therefore, they have no internal knowledge of each other and could be replaced by another problem or optimization algorithm, respectively.

Map Representation

Two basic entities make up an internal representation: `Mob` and `Map`. The first is the ancestor of all features and labels and defines their usual behaviour, the latter stores a set of `Mob`s and some additional properties which determines a whole map.

Mob and its subclasses

Every `Mob` corresponds with an entry of an `ImportanceTable`, according to its importance level. There, the standard values of a sites symbol radius (in case it is circular) and color, text color, typeface and size and the distance requirements Δ_s, Δ_l, δ_{min}, δ_{max} and δ_{opt} are stored. The latter is not used during conflict measuring

[7] In *JAVA*, interfaces are abstract classes that define the behavior of other, concrete classes which can declare themselves to be the implementation. Nevertheless, a class can implement many interfaces, what is the only way to model multiple inheritance.

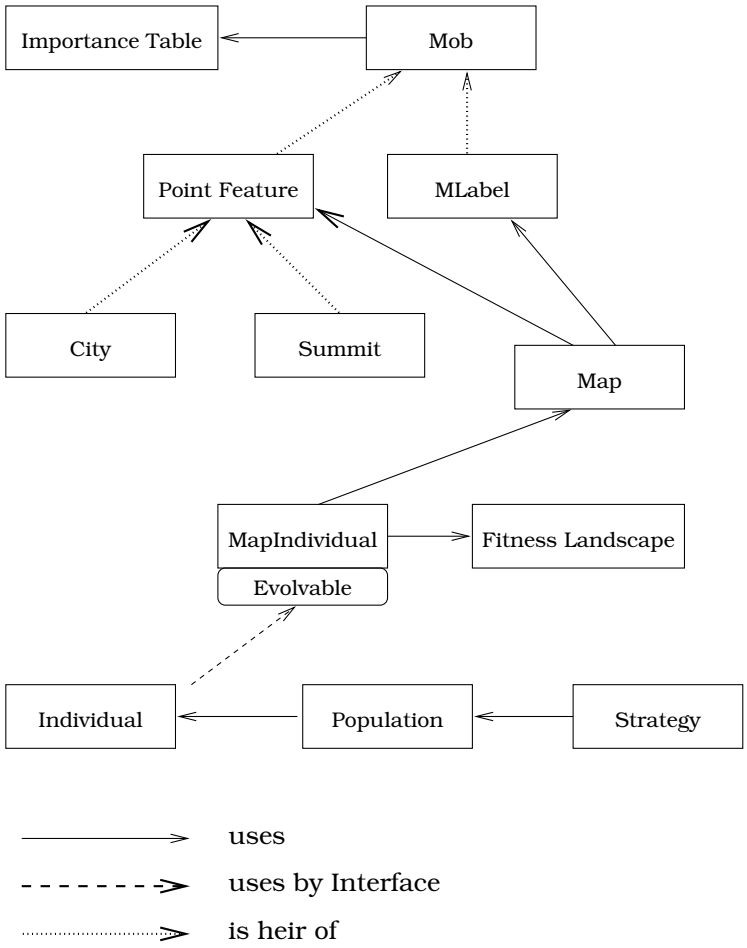

Figure 3.2: The most important classes of the Map Labeling Tool. The lower three classes Individual, Population and Strategy actually implement the ES.

but to make this hidden attribute explicit. It should comply with the optimal distance used in the relative positioning quality function $q(s_i, d, \theta)$. Font attributes, as they have been recommended in [20], can be layed down here, ordered by importance levels. An example is given in table 3.1.

Since the values are copied (not referenced) from the table entry at construction time of a `Mob`, they can be overridden due to cartographic preferences later on. Furthermore, all distance measurings are done with methods of `Mob` and all conflict weights w_{ls}, w_{ll}, w_{lb}, w_a, w_p are stored within its variables.

The `MLabel` (= Mob Label) class of which all labels are instances stores the overlaps that arise while inserting or moving any `Mob`. These are put into conflict lists, one for each label. Therefore, the part of the fitness function that is derived from the

Font name and size	text color	symbol color	radius	δ_{min}	δ_{opt}	δ_{max}	Δ_s	Δ_l
Times Roman 7	black	dark red	2	1	2	5	5	2
Times Roman 8	black	dark red	2.1	1.05	2.1	5.25	5.25	2.1
Times Roman 9	black	dark red	2.21	1.1	2.21	5.51	5.51	2.21
Times Roman 11	black	dark red	2.32	1.16	2.32	5.79	5.79	2.32
Times Roman 13	black	dark red	2.43	1.22	2.43	6.08	6.08	2.43

Table 3.1: An `ImportanceTable` defines standard values for every `MObs` properties, ordered by level of importance (abridged version). All sizes are in pt.

conflict evaluation of a single label can be retrieved without further computation. However, this method turned out to be an efficiency problem later on.

One of the best kept secrets of the design has to be revealed now: how is the relative positioning quality function $q(s_i, d, \theta)$ being modeled? Due to its complex nature if many of the rules described by Imhof are applied on the one hand and the need for a function that can cope with the defined polar coordinate system on the other hand, the use of an additional fitness landscape with the examined site in its center seems appropriate.

Since the landscape should be the same for all label-site pairs, some kind of matching is needed, mapping the angle component of the labels relative coordinates to a normalized angle the landscape description uses. This can be done by setting each of the eight parts of a rounded rectangle to the constant length 1. The normalized angle is simply the number of the octant the label actually occupies (0 - 7) plus the fraction of way it has covered in direction of the next octant, moving clockwise.

The distance component is scaled by $\frac{1}{\delta_{min}}$ before looking up the actual height of the landscape at the label's position.

The landscape is an object of the class `FitnessLandscape` and consists of one or more radial symmetric functions. These are defined by the properties "normalized angle", "normalized distance", "maximum height", "maximum radius", and "degree of the underlying parabola".[8] Angle and distance determine the centers location which corresponds to the point $(1, 1)$ of the parabola, whereas the two boundaries stretch the section $x \in [0, 1]$ of the parabola to the deserved height and width.

If the height of the entire landscape at the actual anchor point of a label (which usually equals the location on its border with the smallest distance to the site) is requested, the largest function value of any parabola in reach is returned.

The class `PointFeature` implements the coordinate transformations stated above what makes them accessible to all of its subclasses. With respect to the contents of

[8] Only functions of the form $f(x) = x^y$, $y \in \mathbb{R}_+$ without additional terms are permitted. Therefore, the conditions $f(0) = 0$ and $f(1) = 1$ always hold.

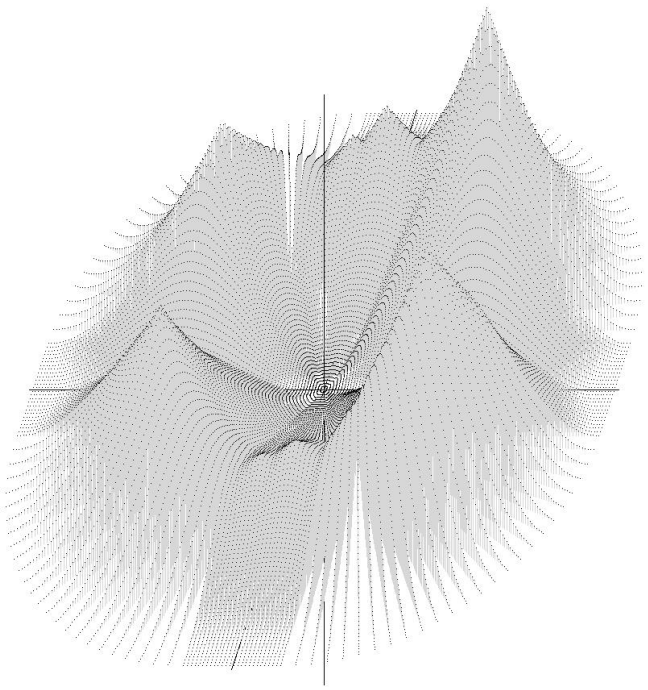

Figure 3.3: The used fitness landscape modeling the rules of Eduard Imhof. Width and height are equal in this example as for a square label. Usually, width is much larger than height what stretches the landscape in x-direction.

the used GNIS files, which also include line and area features, the implementation of two different types of objects was considered worthwhile: summits and populated places. Since all distance computing and coordinate work is derived from the super classes, only the assignment of an appropriate `ImportanceTable` entry to a newly instantiated object remains for implementation. Hence, further classes reflecting other kinds of point features can be added easily.

In opposite to cartographic practice, every summit owns at most one label. Text fields revealing the altitude of a mountain which are often added next to the name box can not be handled due to the restrictions mentioned in section 1.3. Furthermore, they induce a higher complexity into distance and position evaluation:

- Relative positioning may be more difficult. Apart from the fact that only one label can occupy the ideal location, the combination of two text fields may look bad although each of them is positioned the right way. Further cartographic experience should be used to solve this problem.
- Definite assignment has been guaranteed with a simple rule that requires the attached label to have a smaller distance to the site than any other one. How-

ever, the two or more text fields used will mostly have different optimum distances. This requires the rule to be widened.

- It may be of advantage to allow "friendly" labels a smaller distance to themselves than to the ones of other sites. If this holds true, Δ_l cannot be one value of a table but has to be split.

Map and its helper classes
An empty `Map` is constructed by fixing a scale factor and a projection. The latter is handled through an interface also called `Projection` which is implemented by the classes `NoProjection`[9] and `Mercator1Projection`, the standard projection of Mercator with one isometric degree of latitude, taken from [11]. Design allows for adding the deserved `Projection` class at runtime, after reading a parameter file.

The scale factor is applied to the distance requirements and size measurements of all added `Mob`s but has no effect on their position. Changing it expands or reduces all fonts, making the labeling more difficult or easier. The `Map`'s dimensions are not affected since they are only dependent on the projection used and the latitude and longitude bounds provided.

Conflict handling and hence the computation of some numerical value indicating the quality of label arrangement uses an incremental plane sweep algorithm based on the descriptions given in [22] and [16]. Therefore, the constituents of a `Map` are kept ordered by increasing y coordinate value. With the maximum height and required distance known, the range of objects a new `Mob` can affect is found. These are tested for observance of $\Delta_s(o_i)$ and $\Delta_l(o_i)$, or δ_{min} and δ_{max}, respectively[10]. An additional check reveals a possible overlap with the `Map` border. Every time a `Mob` is added to, deleted from or moved within the `Map`, the plane sweep is run to update the conflict lists of the labels which results in a new conflict value for the entire `Map`.

The reason for this incremental approach had been the desired ability to evaluate a slightly changed map. Movement of a single label may remove all existing conflicts and therefore change the fitness dramatically. Nevertheless, this possibiliy was found useless afterwards due to contradiction with the recombination operators, thus the efficiency problem remained.

If the distance check would be computed after insertion of all `Mob`s, it is in $\Theta(NlogN + s)$ (proved in [21] for isothetic rectangles in which the circular symbols can be enclosed for this purpose), whereas the used method has an estimated time consumption of order $O(N^2logN)$. This holds even if the growth of N during the incremental

[9] No coordinate transformation is done, longitude and latitude are interpreted as a point in the Cartesian system.

[10] This distinction refers to different rules for the attached labels of each site.

buildup is considered: search for the appropriate range of objects needs $O(logN)$, testing is in $O(N)$. For all N plane sweep runs, this is roughly $\int NlogNdN = \frac{1}{ln10}\int NlnNdN = \frac{1}{ln10}N^2(\frac{lnN}{2} - \frac{1}{4}) = N^2(\frac{logN}{2} - \frac{1}{4ln10})$, derived using [12].

The class `MapIndividual` implements the interface of the `Map` to the optimization. Besides methods for construction of a master map all other maps are derived from, its main content deals with the evaluation of a map's fitness. Therefore, the conflict value and the relative positioning of each label are taken into account. The latter is computed by using an assigned `FitnessLandscape`.

Data Input

GNIS files contain all named features of a state or administered territory, collected from the processed maps. Since every entry is an *ASCII* string of 240 characters, this is an enormous amount of data for areas densely populated.[11] The larger files include more than 40,000 features of which about one third are the desired summits and populated places. Therefore, some kind of automated selection must take place. The easiest way to obtain a suitable test set of up to 500 point features is to use a latitude / longitude window. Features within this limits are collected and written into an array of `Mob`s which is taken over by the `MapIndividual` class.

Observed data fields are: feature name, feature type, position on earth, and altitude if available.

Visualization

Two different tools emerged from the need to gain knowledge about what does really happen during the optimization:

`LandscapeViewer`: This utility helps in adjusting the `FitnessLandscape` used to model the external relative positioning quality function $q(s_i, d, \theta)$. Basically, it plots any number of three-dimensional points.

`MapViewer`: This application can be used as subtask of the optimization doing on-line visualization, and standalone to replay saved maps. Besides the sites and labels, it also shows conflicts and their evaluated values derived from the conflict lists of each label. Its appearance is shown in figure 3.4.

[11] Some examples: New York $\sim$ 8.8 MB, Florida $\sim$ 7.8 MB, Massachusetts $\sim$ 5 MB, Hawaii $\sim$ 1.7 MB.

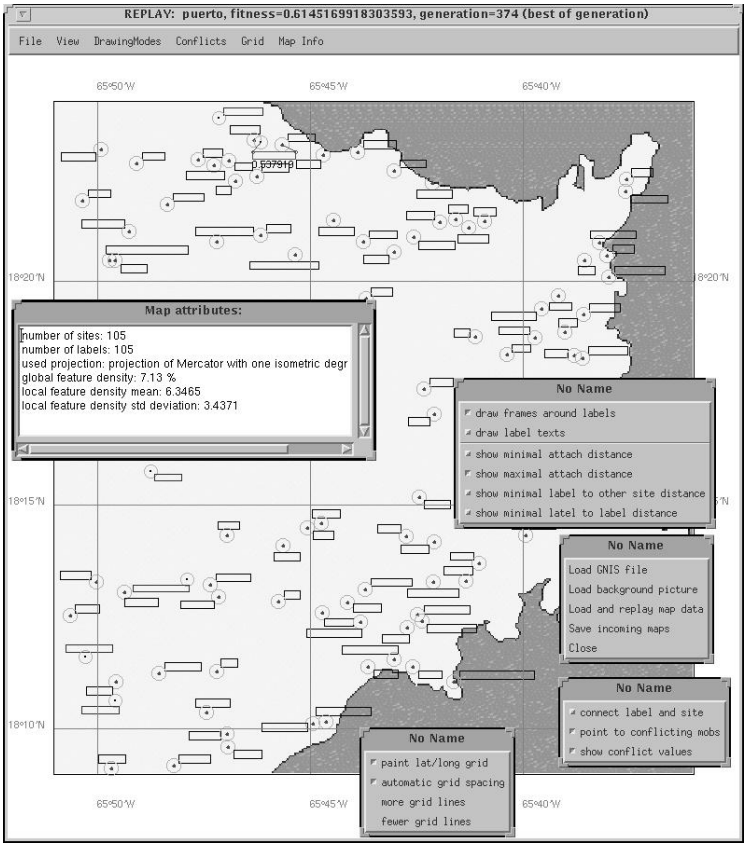

Figure 3.4: The on-line visualization tool `MapViewer` and some of its menue blocks.

Optimization

An instance of the class `Strategy` initializes and supervises the entire optimization and its output. When started, it reads the parameters, creates the first individuals and runs the main loop.

The operators utilized by the ES are deferred to the lowest possible level, that is the `Population` in case of recombination and selection, `Individual` for the mutation.

The optimized problem is provided by a class implementing the interfaces `Evolvable` and `Displayable`. The latter controls the on-line visualization of the proposed problem solutions and is optional. Since the class is linked at runtime, it may be exchanged for another by changing the parameter file. This guarantees reusability of the optimization package.

Analysis

The class `Analyzer` is an application on its own which is used to accumulate the output of many optimization runs and to generate convergence rates as defined in equation 2.14. The written data may be put directly into visualization tools like `gnuplot`.

3.3.3 Implementation

The platform independence stated above has been used for distributed program coding on base of *Solaris*™, *Windows 95*™ and *OS/2*™, due to the availability of different operating systems at different places. Development with the *OS/2*™ *JDK*[12] had to be cancelled in the midst of implementation since it did not comply with the others. At this time I came across the *KAWA*™ tool which is free for students and educational staff and served well during the second half of the implementation.

In general, cross-platform development worked well. Some insignificant differences have been recognized affecting event handling and display update behavior. However, a number of problems arised from the implementation language itself:

- Up to now, there is no way to retrieve exact heights of characters. Furthermore, the font metrics received for a string seem to be distorted, they lack accuracy particularly in case of slanted fonts.
- Several new *JAVA* version have been shipped during the going of this work. Some of the important features[13] came up first with the release 1.1 and convinced me to make use of it.
- While using the batch system of the Chair of Systems Analysis, the *JAVA* application behaved somewhat strange. Although no screen output was generated, it always claimed access to the display. If this got lost, it stopped execution immediately.

Some other improvements of the design were made:

- At first, the `Map` instance was designed to adapt its size to the position of the inserted `Mobs`. Hence, conflicts with the borders had been impossible. Nevertheless, strictly bounded maps may be important to cartographic practice. Therefore, the borders were made immovable later on.

[12] Java Development Kit, including compiler, interpreter, command line debugger, standard library and some tools

[13] as there are persistence and the metadata concept

- Underestimating the storage needs of the internal map representation, I tried to buildup an entire population (which can be up to ∼100 maps) at once. In consequence, the ES allocated dozens of megabytes of memory. The concept has been changed then as to instantiate and evaluate `Map`s in sequence.

Chapter 4

Optimizing Maps

Since the optimization tool and its functionality has been described, some optimization results may be presented. To ensure comparable output from experiments with different parameters, a set of test maps (see table 4.1) is defined. From the vast amount of data available from the GNIS archives, I chose fourteen rather small geographical windows. Although states and territories with fewer feature entries have been preferred in order to simplify file handling, a mixture of inland, coastal and island regions emerged. Coastal regions and inland areas mainly differ concerning the extent of water they include. Geographical windows with a portion of more than 15 % covered by sea are classified coastal, inland else.

To all labelings, a relative positioning quality function $q(s_i, d, \theta)$ modeling Imhof's rules (referring to [13]) has been applied. Therefore, implicit priority order of label positions is as follows:

1. On the right of the site, slightly raised against its center
2. Equal to 1, but lowered against the site's center
3. On the left of the site and raised or lowered against the center by a fixed, small distance (each of the two possibilities gets the same reward)
4. On top of the site, anchored at the left third of the label's bottom edge
5. As 4, but anchored at the right third
6. Below the site, two thirds of the label laying to the right of it
7. As 6, but two thirds of the label laying to the left of the site.

state or territory	map type	longitude range	latitude range
Delaware	inland	$75.515^oW - 75.455^oW$	$39.770^oN - 39.820^oN$
Guam	island	$144.640^oE - 144.780^oE$	$13.230^oN - 13.350^oN$
New Jersey	coastal region	$74.140^oW - 74.000^oW$	$39.980^oN - 40.150^oN$
Northern Mariana Islands	island	$145.580^oE - 145.830^oE$	$14.900^oN - 15.290^oN$
Kansas	inland	$97.500^oW - 97.000^oW$	$37.500^oN - 38.000^oN$
Kaua'i (Hawai'i)	island	$159.490^oW - 159.250^oW$	$22.050^oN - 22.250^oN$
Louisiana	inland	$90.000^oW - 89.65^oW$	$29.770^oN - 30.270^oN$
Massachusetts	coastal region	$70.580^oW - 70.000^oW$	$41.700^oN - 41.875^oN$
Moloka'i (Hawai'i)	island	$157.350^oW - 157.050^oW$	$21.050^oN - 21.250^oN$
Oklahoma	inland	$96.500^oW - 96.000^oW$	$34.000^oN - 34.500^oN$
Republic of Palau	island	$134.400^oE - 134.625^oE$	$7.300^oN - 7.450^oN$
Puerto Rico	island	$65.850^oW - 65.600^oW$	$18.150^oN - 18.400^oN$
Rhode Island	coastal region	$71.750^oW - 71.450^oW$	$41.110^oN - 41.410^oN$
Virgin Islands	island	$65.050^oW - 64.800^oW$	$18.300^oN - 18.400^oN$

Table 4.1: The set of test maps used for nearly all experiments.

4.1 A Typical Outcome

The earliest tests with the ES have been done with even smaller parts of the maps of Kaua'i and the Virgin Islands. As it proved its general ability to solve MLPs, larger maps have been considered as well. The most populated one covered the Eastern part of Massachusetts and included more than 750 sites. Due to computation time constraints, these huge maps have not been investigated at length[1].

Further on, presentation of experiments will generally consist of two parts, a number of figures and a textual record. The range of used figures includes:

maps: Optimization results can be compared directly to an initial labeling. The boxes contain label texts, the sites are indicated by small circles, surrounded by bigger ones, marking the outmost distance allowed: δ_{max}. In most cases, the depicted area is not the only one used within an experiment. Example: figure 4.1.

fitness plots: Progress of the minimization is reported every time a better fitness is reached. Multiple runs are accumulated by computing the mean of the lowest values found by each of them after every function call. The plots use a logarithmic scale according to the predicted convergence behaviour of the ES (see section 2.3). Example: figure 4.3.

convergence plots: The actual convergence rate is calculated every time the best fitness of any run changes, using equation 2.14. It relates the logarithmic fitness advancement with the time needed to perform it. Example: figure 4.4.

[1]Since the optimization is a *JAVA* application, calculating one generation (80 individuals) of the ES takes about 5 minutes on a fast workstation for maps of this size.

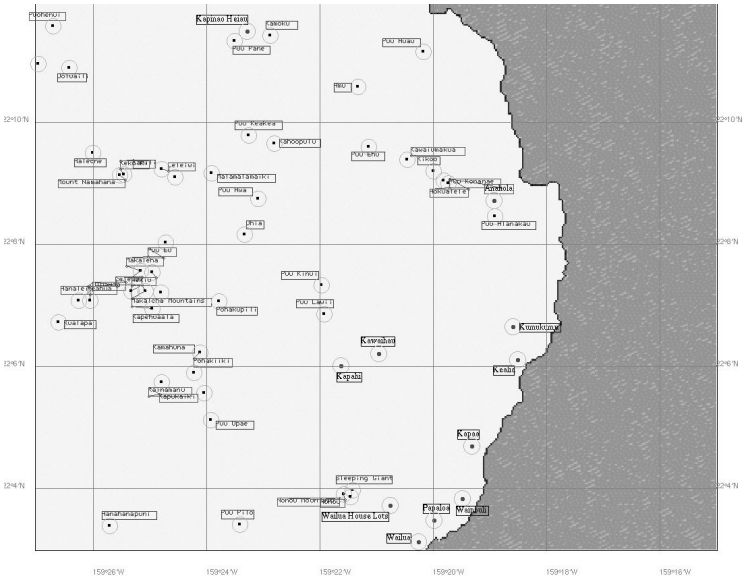

Figure 4.1: Initial state of an optimization run on map Kaua'i.

parameter comparison plots: These figures are needed to summarize multiple runs with different parameters. In these plots, mean values of the resulting convergence rates (measured from the start to the last reported point), grouped by equal parameters, express the overall success of a special setting. The error bars may reveal some information about convergence diversion within the runs. Example: figure 4.8.

The text records consist of three parts: a rough description of the used maps and parameters, a visual analysis of the figures, and a discussion in search for explanations of the obtained results. A text record example follows:

Experiment: Map Kaua'i, no parameter modifications, 3 runs

Analysis: As shown in figure 4.1, the ES starts with a random labeling containing many conflicts. After optimization has finished (figure 4.2), two labels on the left side are left in a state of conflict with some sites. Both fail to comply with the requirements for definite attachment, so they are placed inside the area a site occupies to facilitate identification of its own label. Since placing them by hand appears impossible, a valid labeling may not exist.

The fitness plot is not as smooth as it has been expected but reveals at least two different phases of optimization.

Discussion: If the optimization fails to find a valid labeling, the problem may be unsolvable. If so, application of three techniques can help to make it simpler,

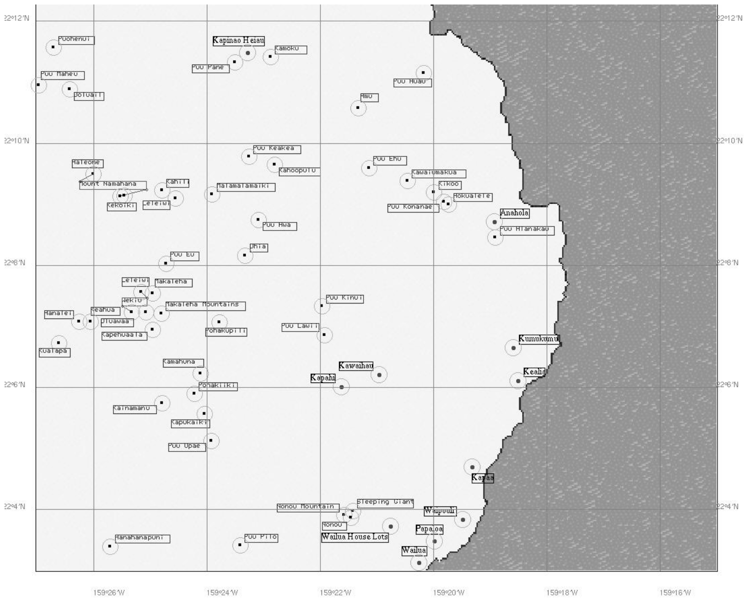

Figure 4.2: The last recorded map instance of the run that begun with figure 4.1.

according to the appearance of the remaining conflicts. These methods are not handled by the algorithm upto now.

- The map is overcrowded with features. Many conflicts remain unsolved and almost no uncovered areas exist. To keep all features on the map, the text size and/or minimum required distances should be reduced.
- The feature names are too long. Therefore, labels get stuck where they would have to cross others to get to an uncovered area. It would be common practice in cartography to separate the name into two lines of text.
- In dense feature clusters, there may exist inner points that are surrounded by others to all directions. As it is impossible to label them without overlap, they should be marked with a symbol that is explained somewhere else on the map. This method is known as point selection.

The observed steps during optimization progress indicate the part of the fitness function that is actually treated. Due to its higher weight, this is conflict removing at first. Every solved conflict appears as particular level while observing the fitness development. After overlaps have been removed or found unresolvable, improvement of the relative positioning remains the only possibility of achieving a better fitness. The line of best fitness becomes much smoother now.

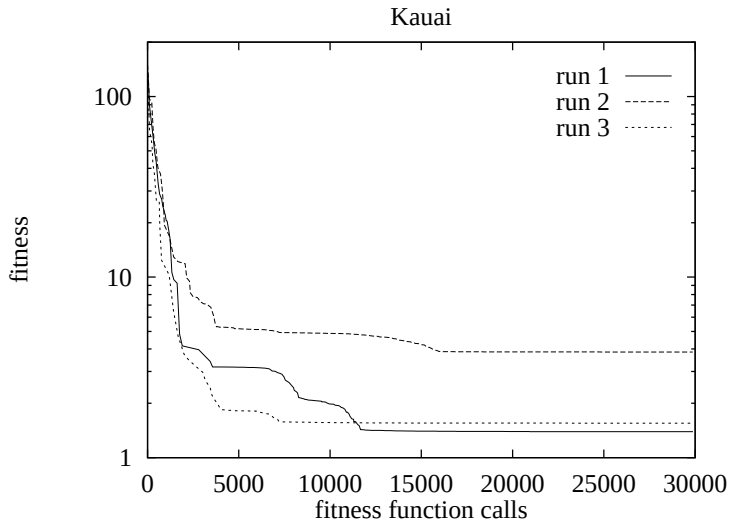

Figure 4.3: Three runs on map Kaua'i, fitness plot.

Note that some labels have not been put to the right of their site although this is the recommended place according to the used relative positioning function (figure 3.3). Since the placement performs much better if all conflicts have been removed during an optimization run, this behavior may arise from the low weight of the quality function $q(s_i, d, \theta)$.[2] Even small label moves from or into covered areas produce a strong impact on the fitness value which may supress all positioning activity. The selection operator of the ES has to fail on this situation because positioning rewards become "visible" only if conflict resolving comes to an end. However, lowering the conflict weight may be no good deal, either. Overlaps could remain as they are balanced with a better positioning reward then. A possible way out of this dilemma is the subject of section 4.3.2.

4.2 On Search for Good Parameters

Since a standard parameter set was used for optimization at first, nothing was known about its suitability to the MLP. Therefore, several experiments have been performed to find good and robust values by variation of one parameter at a time. Unless stated otherwise, the described test set of fourteen maps has been used.

The time needed by an optimization run is mostly determined by the number of calls to the fitness function and hence the construction of maps. Therefore, parame-

[2]The weight of the positioning part is only $\frac{1}{1000}$ of the conflict weight for most runs.

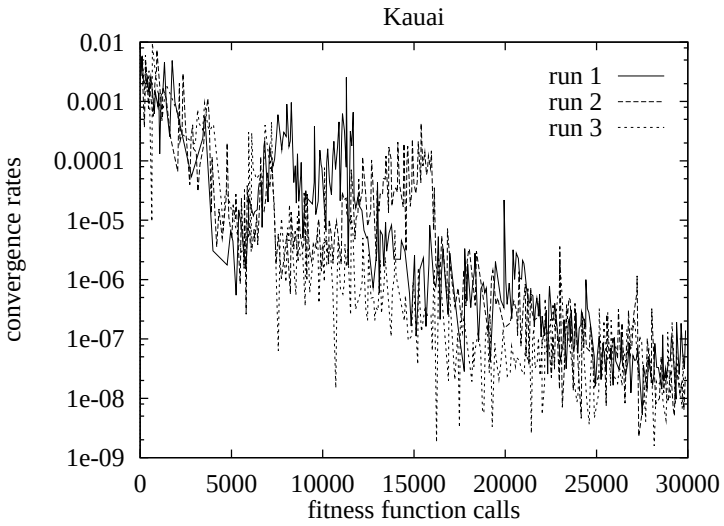

Figure 4.4: Convergence rates of the optimization runs depicted on figure 4.3.

ter sets looked for result in increased convergence rates (defined in equation 2.14) at a constant number of search steps. In this section, a limit of 30,000 fitness function calls has been used for all experiments.

4.2.1 Recombination Type

Two different types of recombination are available from the operator set enumerated in section 3.2.2: global intermediary recombination and uniform crossover. The first creates an individual by computing the mean value of all parents, whereas the latter builds up new combinations of the parents' components. As an individual consists of two different types of variables (object and sigma values), the recombination can be done in four ways. However, those using intermediary recombination on the object variables have been totally left out because their optimization ability during first tests appeared to be very low.

Why does this happen? Intermediary recombination always tends to draw the used components towards a mean value. Hence, the object values are focussed on a small range if generated this way. While doing labeling of maps, preservation of diversity seems more important than a concentrated search. As the object variables of the individuals consist of the label coordinates, the good positions may hopefully be combined if the crossover recombination is used. Nevertheless, both operators have been applied to the sigma values.

Experiment: All 14 maps, intermediary recombination and uniform crossover on

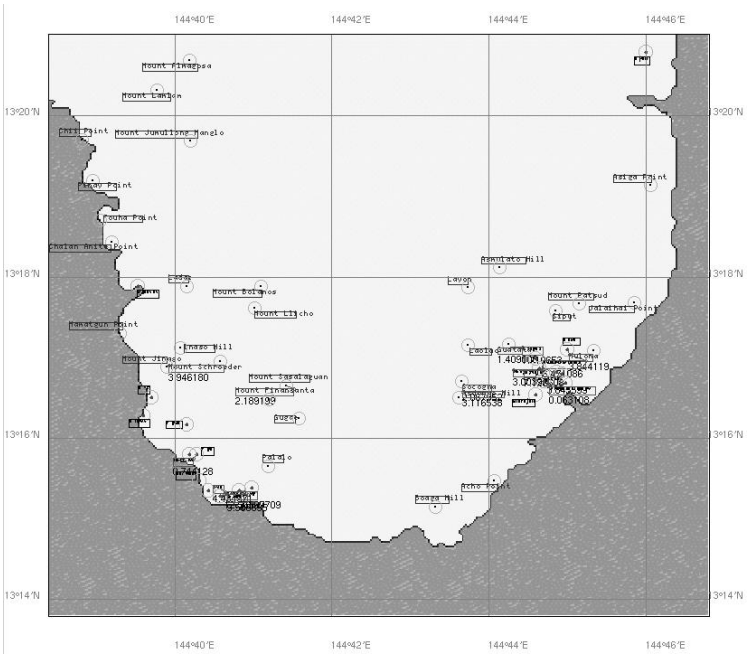

Figure 4.5: Initial state of labeled map of Guam.

the sigma values, total runs: 56

Analysis: The comparison plot reveals a higher mean convergence if uniform crossover is used.

Discussion: That is what has been expected. I assume the slightly better performance of the ES with uniform crossover originates from a greater diversity of the sigma values. The use of uniform crossover is recommended for recombination of object variables and sigma values.

4.2.2 τ

τ is one of two external parameters provided to the ES to control self-adaptation of the sigma values towards the problem's fitness landscape.[3]

Mutation of an object variable x_i basically works by adding a normally distributed random number the variance of which is determined by the appropriate mean step size σ_i (equation 4.1, cited from [30]). As described by the upper line of the equation, σ_i itself undergoes a change before it is applied to x_i. The mutation of σ_i is strongly influenced by τ' and τ. The former is a global factor weighting a random

[3]This is not the `FitnessLandscape` used for positioning evaluation but the multi-dimensional search space the individuals are moving in.

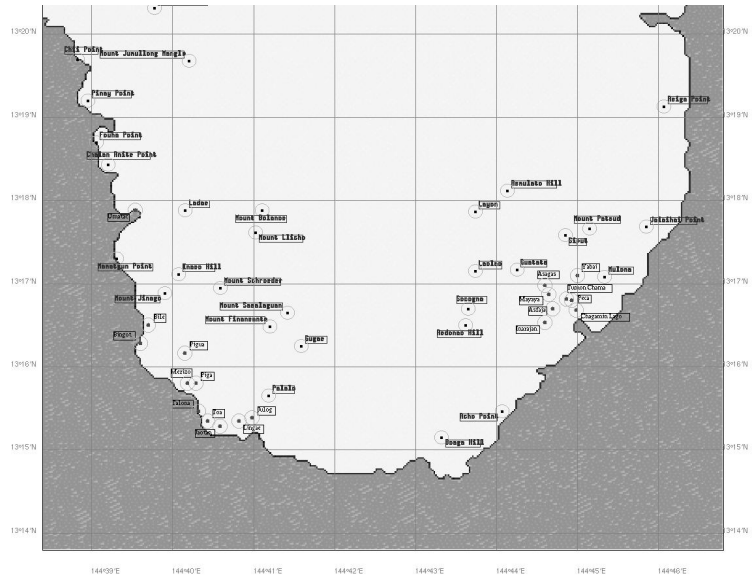

Figure 4.6: Last recorded map of Guam, ending the optimization run started with the map in figure 4.5.

number $N(0,1)$ that is the same for all object variables of the individual whereas the latter indicates the weight of each component's private random number $N_i(0,1)$. Hence, a higher value of τ or τ' means faster motion of the σ_i.

The self-adaptation stated above works by selection of the "better" individuals. Although the mean step sizes do not affect the fitness directly, they control the dynamic behaviour of the object variables. In the long run, individuals whose σ_i are appropriate for the landscape are able to move more efficiently towards its minima and are thus selected whereas the others jump around or hold position in the search space.

Since fast adaptation can lead into local minima, chosing τ and τ' always means a trade-off between convergence speed and probability. Small τ and τ' values may waste time and great ones may get the ES stuck in a good but not optimal position. Despite the recommendations of $\tau' \sim \frac{1}{\sqrt{2n}}$ and $\tau \sim \frac{1}{\sqrt{2\sqrt{n}}}$, proposed in [8], I used constant values inbetween 0.1 and 0.5 for almost all experiments.

$$\begin{aligned} \sigma_i' &= \sigma_i exp(\tau' N(0,1) + \tau N_i(0,1)) \quad , \\ x_i' &= x_i + \sigma_i' N_i(0,1) \end{aligned} \tag{4.1}$$

Experiment: All 14 maps, τ ranges from 0.1 to 0.5 by steps of 0.1, number of total runs: 70

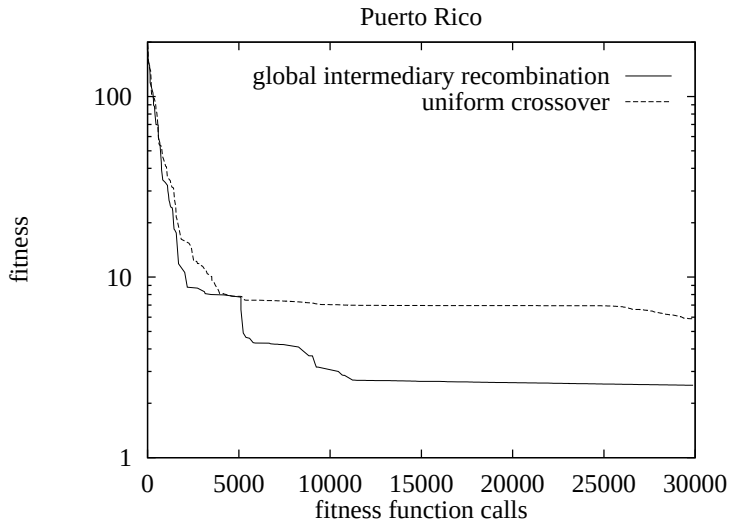

Figure 4.7: Comparison of recombination types while labeling the Puerto Rico map (accumulated fitness of 2 runs each).

Analysis: The mean value of the convergence rates raises from $\tau = 0.1$ to $\tau = 0.4$ and then falls again. The best convergence ever is reached at $\tau = 0.3$ which is also the parameter setting with the smallest deviations which means good convergence behavior is most probably reached.

Discussion: The recommended value of τ is approximately 0.3 .

4.2.3 λ and κ

Usually, one would expect variations of λ and κ being discussed separately. However, parameter experiments revealed some similarity of the results received from varying each of them. This may be due to the different degree of selection pressure they exert on the offspring individuals.

λ denotes the number of offspring individuals that is created while computing one generation. Therefore, a higher ratio of λ to μ (which is the number of surviving individuals) entails a higher selective pressure. For all runs with varying values of λ, κ has been set to 3.

For an ES that uses aging as a compromise between "hard" $,$-selection and "soft" +-selection, κ denotes the maximum number of generations an individual can survive. All tests with different values of κ were run with a $(15, 60)$-ES.

Experiment: 1. All 14 maps, λ ranges from 30 to 100 by steps of 10, number of total runs: 112.

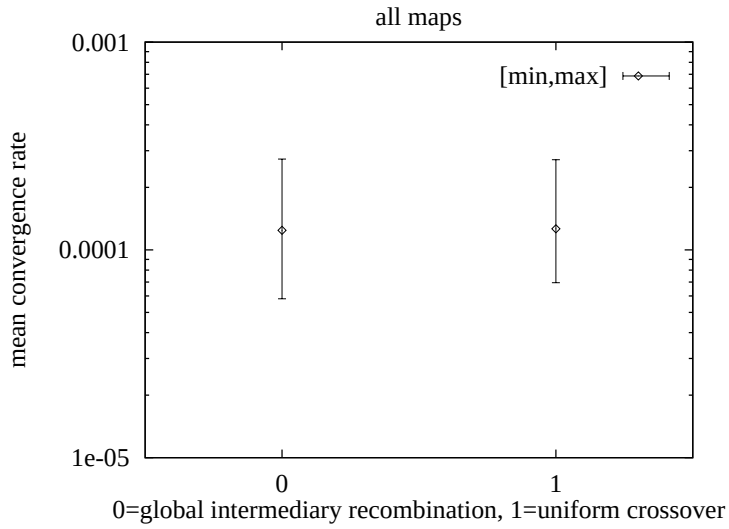

Figure 4.8: Comparison of recombination types for whole test set of maps (accumulated convergence rates of 28 runs each). Although global intermediate recombination performs better on the Puerto Rico map (figure 4.7, the uniform crossover barely wins the race for all 14 maps.

2. The full test set, $\kappa \in \{1, 2, 3, 4, 5, 6\}$, total runs: 84.

Analysis: The effect of different selection pressures on the convergence rates results in a waveform Since the shape of this waveform is quite similar in case of the κ and the λ test, they are both shown in figure 4.15.

Discussion: It is undeer why the two peaks of either plot are both located at a very low $(40/2)$ and a standard value $(90/5)$, but as the second one seems more robust, it should be recommended for further use.

Notice that only one of the two parameters was changed at a time while the other was set to $\kappa = 3$ and $\lambda = 60$, respectively. The behavior of the ES may change for different combinations.

If the set of parents is very stable for the time the individuals are allowed to stay therein, the κ-selection ES is comparable to a ν-strategy with a higher λ. The number of individuals getting the chance to supplant a parent grows with increasing λ and κ. The main difference is the temporal stretching of the selection process in case of the κ-selection.

However, some correlation of the selection pressure arising from equivalent values of κ and λ can be assumed from figure 4.15. Taking this into consideration, I propose the following hypothesis: the selection pressure exerted on the population by a κ selection is proportional to the one produced by an

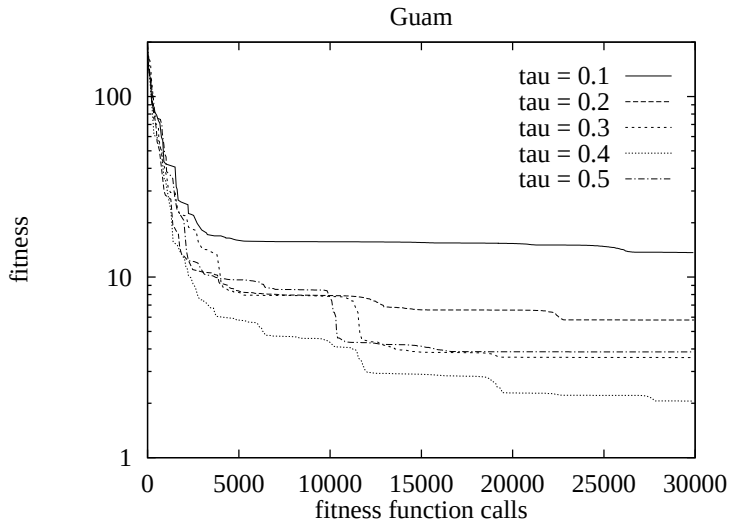

Figure 4.9: Optimization of map Guam, using 5 different τ settings inbetween 0.1 and 0.5.

equivalent ratio of dying to surviving individuals under some circumstances (equation 4.2).

$$\kappa \sim (\frac{\lambda - \mu}{\mu}) \tag{4.2}$$

By pure chance, the values that were remaining constant while the others were varied are equal in terms of 4.2: $3 = \frac{60-15}{15}$. Nevertheless, this hypothesis has to be investigated further, using a larger amount of optimization runs.

4.2.4 ρ

ρ is the number of parents used to build up a new individual during recombination. Therefore, it measures the speed of "communication" by which good components are spread out into the entire population.

Experiment: All 14 maps, $\rho \in \{1, 2, 3, 4, 5, 6\}$, number of total runs: 84.

Analysis: The rise of the mean convergence rates seems to reach its climax at $\rho = 4$. Unfortunately, no labelings have been done with $\rho > 6$. Nevertheless, decreasing convergence rates for greater parent numbers seem to be more likely than increasing ones, according to figure 4.17.

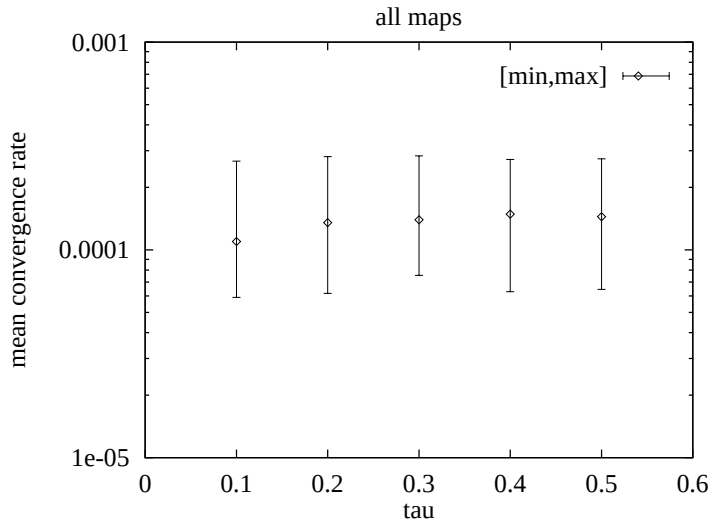

Figure 4.10: Accumulated convergence rates for the entire set of test maps (14 runs per value of τ).

Discussion: If ρ is set to 1, recombination is eliminated. Hence, mutation has to do all optimization. Probably, the best individual has to adapt the σ_i on its own and removes conflict after conflict, producing a fitness development looking like a stairway (figure 4.16).

Since $\rho = 1$ means total isolation and $\rho = \mu$ is immediate and unlimited communication, a compromise between these two extremes may serve well. By this trade-off the disadvantages of either side should be avoided, resulting in weak locality (though no topology is imposed on the individuals) and slow communication. This finding complies with a similar one documented in [24].

If $\rho \geq \sqrt{\mu}$ holds, information (good label positions) of all parent individuals may be gathered within two generations. As this appears to be advantageous, a recommended setting of $\rho = \sqrt{\mu}$ is concluded.

4.3 Adjusting Fitness Weights and Minimum Distances

The weights used to measure different components of the fitness function may have enormous influence on the ES's ability to find good labelings. Since they indicate the most "profitable" changes of the object variables, optimization is driven to consider the referring subtasks first. For a MLP, removing any overlap has been considered the primary target. Therefore, utilized conflict weights are approximately $1,000$ times higher than the one of relative positioning which forces the ES to forget about aesthetics as long as conflicts can be resolved. However, weight balance of

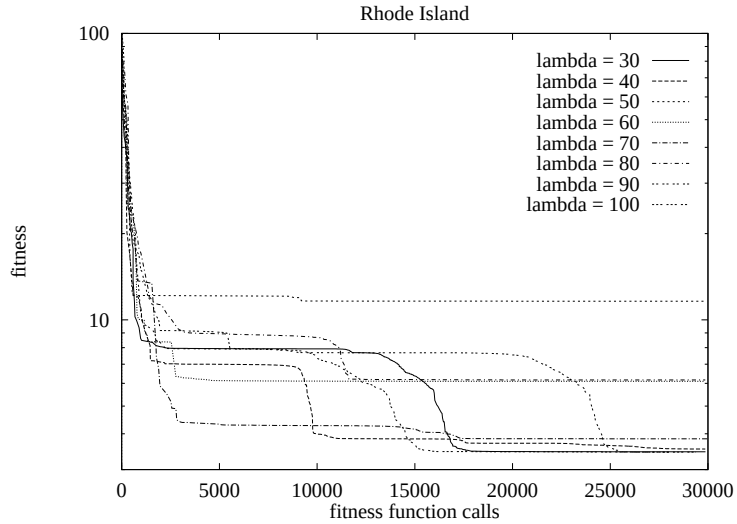

Figure 4.11: Fitness while labeling the Rhode Island map with different settings of λ. Center values with $\lambda \in \{50, 60, 70\}$ do not climb down to the range of "good" results.

these two components has been subject to further investigations reported in section 4.3.2.

Fixing minimum distances is a difficult task because it requires a trade-off between readability and problem complexity. If many sites are packed densely and thus occupy the area of maximal attach distance of each other, the optimization may fail in finding a valid labeling because none exists. This is understood easily if two sites with identical geographic positions should be labeled by hand, given $\delta_{max} < \Delta_l$[4]. Each site draws near its own label but pushes away the foreign one, resulting in an unsolvable deadlock. Therefore, Δ_l has been adjusted to be equal to δ_{max}.

4.3.1 Recovering Lost Labels

At first, the weight used for conflicts of labels with borders has been set equal to the one for label-label overlap. Hence, some sites laying aside a border lost their labels. If these have been located near to many other sites, those labels pressed them off the map sheet. Since crossing a label is quite "expensive" in terms of the fitness function, they had to stay outside as long as the border remained "guarded" by another label.

Therefore, the border-conflict weight has been dramatically increased to a value much higher than the other weight factors. This forces the labels to be kept inside

[4]This requirement has been laid down in section 2.1.2

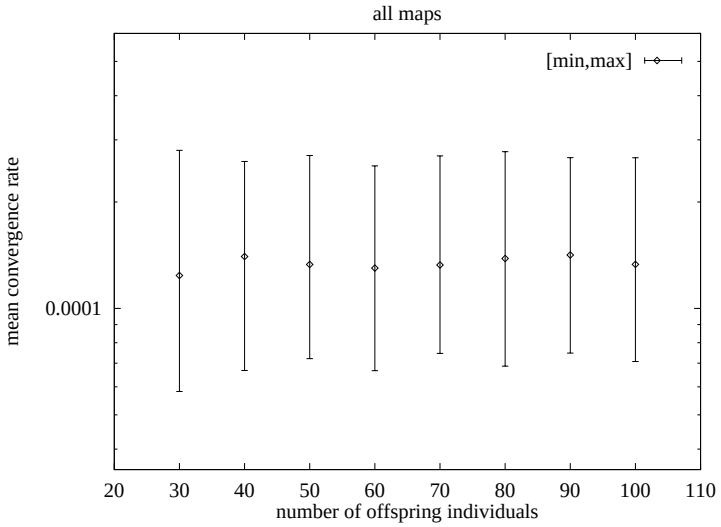

Figure 4.12: Mean convergence rates of the entire set of test maps labeled, varying λ. The two settings $\lambda = 40$ and $\lambda = 90$ appear to be most advantageous.

the map's borders and removes this type of conflict after a few generations.

4.3.2 Balance of Positioning and Overlap Avoidance

The aim of this section has been to find another weight w_p of the external relative positioning quality function $q(s_i, d, \theta)$ which had been integrated into the fitness function to represent aesthetic criteria. Therefore, optimization runs with w_p ranging from 0.0002 (that is $\frac{1}{10,000}$ of the label-site conflict weight w_{ls}) to 20 have been carried out.[5] The results cannot be presented using an ordinary comparison plot. As, for great w_p, the fitness of the start individuals changes dramatically, the convergence rate defined in equation 2.14 is distorted.

However, manual interpretation reveals no advantage of different w_p. For a wide range of $0.0002 \leq w_p \leq 0.2$, almost no change in the ES's attitude towards the two targets "conflict removement" and "aesthetic placement" can be recognized. If w_p comes close to w_{ls} (which is 2), more and more conflicts remain because the ES concentrates on gaining profit from finding optimal relative placements.

Solving both problems at once appears to be impossible with the used fitness function.

[5] Actually, test settings have been $w_p \in \{2 * 10^{-4}, 2 * 10^{-3}, 2 * 10^{-2}, 2 * 10^{-1}, 2, 20\}$.

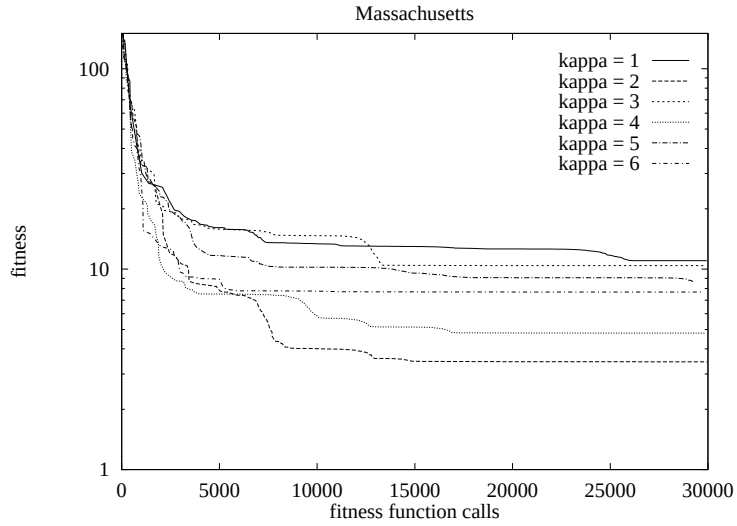

Figure 4.13: Optimization of map Massachusetts with various values of maximum lifetime. Best performance results for $\kappa = 2$ and $\kappa = 4$.

4.4 Investigating Map Complexity

While testing the ES on labelings of different complexity, two directions appear worthwile investigating:

1. How does increasing map size affect the ongoing of optimization?
2. May a rule for complexity estimation be derived from experiments with growing feature sizes (and therefore increasing global feature density)?

Experiments with maps of increasing size revealed no significant change of performance if the added areas are not overcrowded with features.

Since it is difficult to summarize results on feature sizes as the appropriate values depend on the map's attributes, two maps are investigated separately.

Experiment: map of Oklahoma (on which features are arranged very regularly), 12 different feature sizes from $1\ pt = 0.00002$ to $1\ pt = 0.000057062$, number of total runs: 24

Analysis: The last valid labeling lies on the edge of a peak in figure 4.18, convergence rates decline from there on.

Discussion: To predict a feature size that enables a valid labeling, it would be necessary to find the peak on the left of figure 4.18 and then detect its edge. For

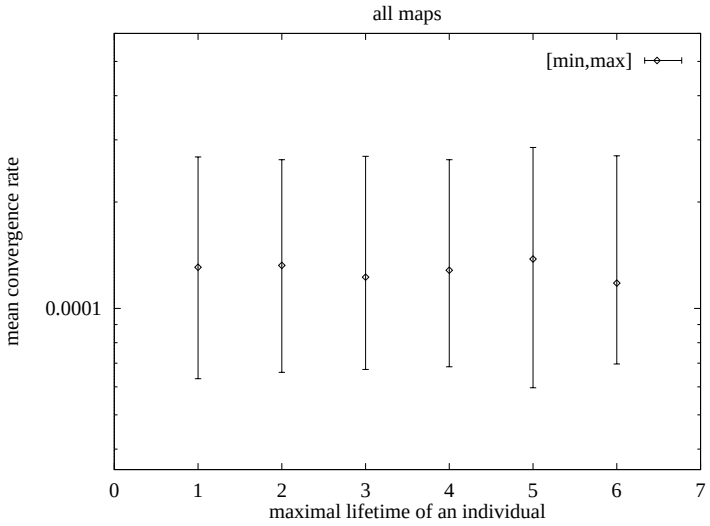

Figure 4.14: Accumulated convergence rates for the entire test set, varying κ. Labeling performs best with $\kappa = 2$ or $\kappa = 5$.

higher feature sizes, the convergence rates are falling fast at first and then reach a level of saturation.

Experiment: map of Kansas, 12 different feature sizes from $1\ pt = 0.00001$ to $1\ pt = 0.0000285309$, total runs: 24

Analysis: The behavior of the ES appears similar to the one exhibited while optimizing maps of Oklahoma, except for the peak on the left side of figure 4.19 which seems to be much broader.

Discussion: The steps found in the interpolation of the mean convergence rates may be originated by the map areas which are densely crowded, as is Kansas City. These feature centers seem to "collapse" if the feature size is increased beyond a certain threshhold, resulting in high penalties for many new conflicts. But in order to reach statistical significance, more optimization runs on different maps have to be carried out.

Additionaly, the development of the global feature density (see equation 2.13 on page 20) has been observed while changes to the feature sizes have been tested. As a rule of thumb, it can be concluded that labeling becomes "very difficult" if the global feature density increases to more than 10%.

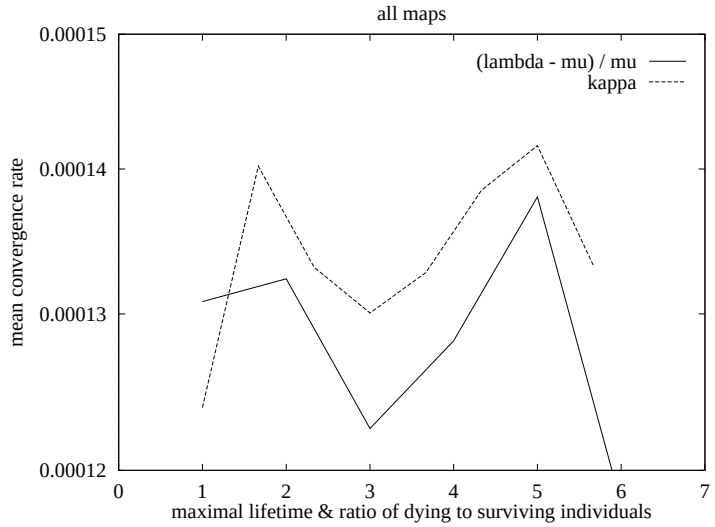

Figure 4.15: The mean convergence rates (figures 4.14 and 4.12) of experiments with different values of κ and λ, plotted into one coordinate system. Results of tests varying λ are transferred to $(\frac{\lambda-\mu}{\mu})$.

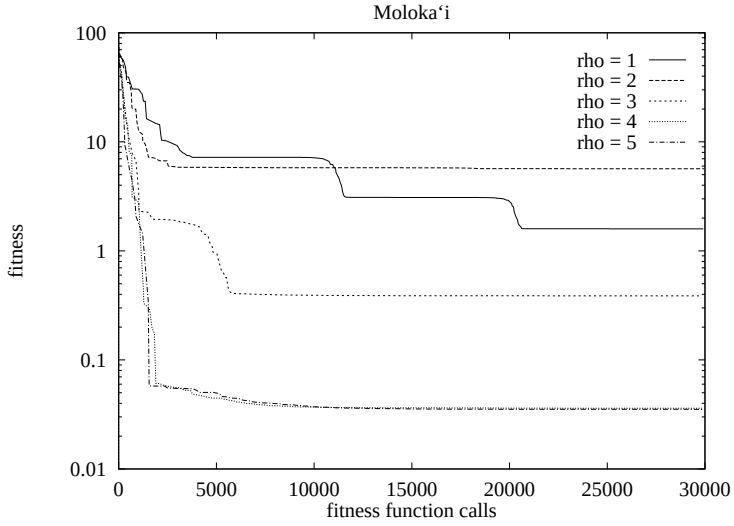

Figure 4.16: Fitness development while varying ρ, the number of parents of each individual. Best performance with $\rho = 4$ and $\rho = 5$.

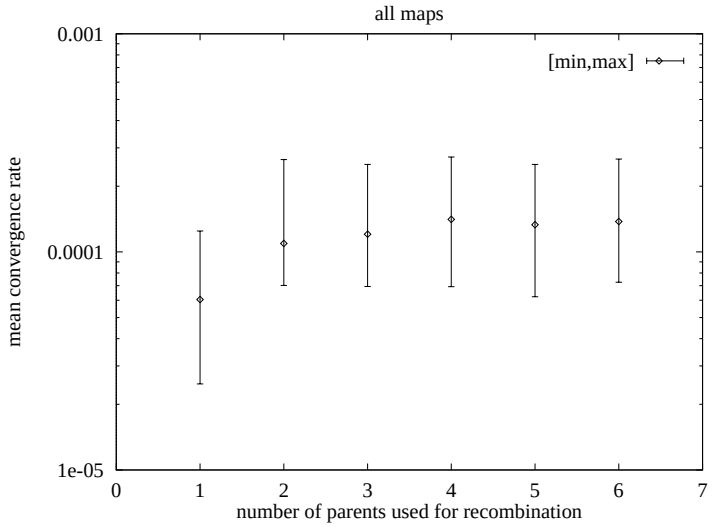

Figure 4.17: Convergence rates of (14 runs each) labelings with changing number of parents used to create an individual (ρ). Good performance results for $\rho \sim 4$.

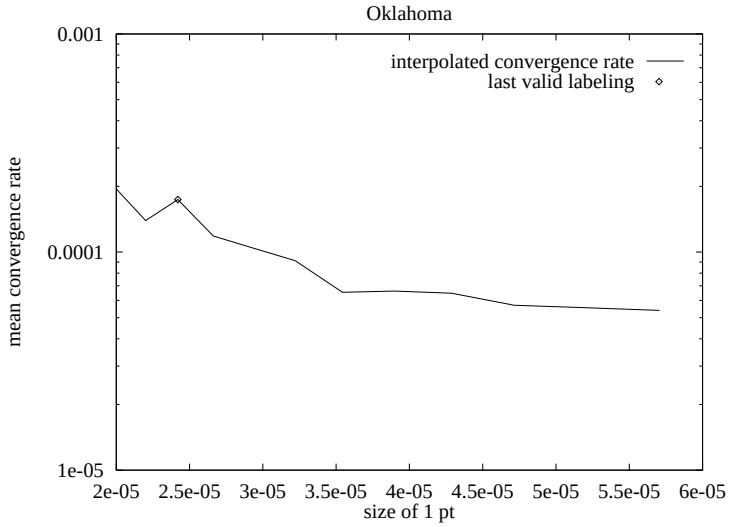

Figure 4.18: Accumulated convergence rates for the map of Oklahoma, varying the feature sizes.

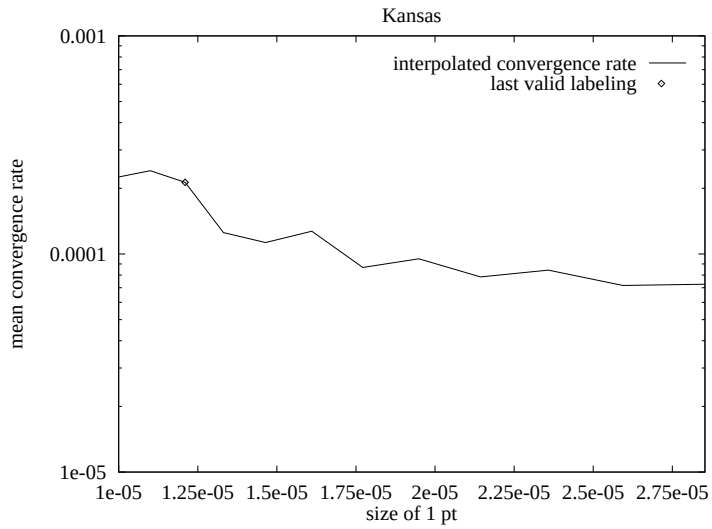

Figure 4.19: Accumulated convergence rates for the map of Kansas, varying the feature sizes.

Chapter 5

Evolving the Algorithm

Since the observed performance of the ES while labeling maps is good but not overwhelming, some changes in the algorithm have been tested, striving for better convergence rates. Without a fair on-line visualization of the optimization, none of these proposed changes would have been possible.

5.1 Variation of the Mutation Rate

This invention has been made to increase the ability of the ES to jump out of deadlocks which may arise if a label gets stuck on the "wrong" side of its point feature and is surrounded by sites to all directions it can travel to. Fitness costs of crossing (this means penalization due to overlap) any site are very high, holding the label at its suboptimal position. If the ES is allowed to do some steps without evaluation, it may be able to "cheat" the fitness function.

Therefore, the commonly used concept of mutation probability p_m has been extended to an assured mutation (if $p_m \geq 0$) and an additional one carried out with a probability of $p_m - 1$.

Experiment: 8 of 14 maps, 20 mutation rates inbetween 0.1 and 2.0, number of total runs: 480

Analysis: The ES performs best if the mutation rate is set to 0.7, $1.2 - 1.3$ or 1.6. Leaving out 25% of all mutation events or adding on another approximate 25% the standard ES mutation rate seems the be of advantage.

Discussion: The success of mutation rates $p_m \in [1.2, 1.3]$ may be due to the permitted "unwatched steps" which may enable crossing of densely crowded areas.

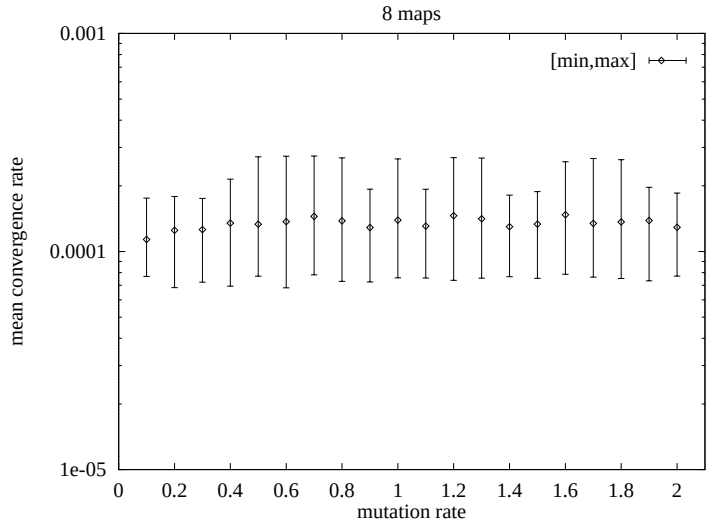

Figure 5.1: Mean convergence rates for different mutation rates, every entry is generated from 24 runs.

5.2 Incremental Map Assembly

An incremental approach to map labeling may be rewarded by two advantages:

1. The amount of calculation used for conflict evaluation decreases if there are fewer features most of the time of the optimization run.

2. Placement by insertion may be easier than by movement.

Therefore, the ES has been modified to allow incomplete individuals. To keep recombination working, the full component block (2 object variables and sigma values for every label) is used as before with some components declared as void. Some semantic structure had to be added, too, because a label position containing a valid distance and an undefined angle (or vice versa) can not be interpreted properly. Hence, a parameter called *buildingBlock* has been invented to tell the ES the number of components that should be treated as one in terms of recombination. To ensure incomplete start individuals, the initialization has been changed to use a probability $0 \leq p_i \leq 1$ for any component instead of $p_i = 1$ for the standard versions of the ES. As the *buildingBlock* is 2 in this application, the likelihood of both components being initialized is p_i^2.

Individiduals (Maps) with less valid components will probably have a reduced number of conflicts. Therefore, the fitness function has been changed to a relative value $\frac{fit}{r}$, with the number of valid labels r. As a reward for more complete maps,

the denominator has been increased[1], reducing the fitness of highly filled maps by far below the one of sparsely populated maps with similar relative fitness. Nevertheless, this approach failed. No complete maps emerged, even if they had been produced with the standard method. As soon as conflicts arised, further insertion of labels stopped.

Within a second approach, I tried to elude this weakness using a repair operator. After recombination has been finished, the operator is run on every new individual, searching for invalid components and repairing them with a probability of p_r. Hence, a valid component of one of the parents is chosen randomly and inserted instead of the void one. As the individuals are repaired permanently, this creates a counterpressure to the fitness losses arising from the insertion of new labels.

However, p_i and p_r need to be balanced carefully. If p_i or p_r are too low, the algorithm may diverge. On the other hand, high values of p_i and p_r reduce the computation time savings earned from the incompleteness of the individuals. Therefore, two experiments have been accomplished to conceive proper values for each parameter.

Experiment: all maps, $p_r \in \{0.2, 0.3, 0.4, 0.5\}$, total number of runs: 56

Analysis: The comparison plot of mean convergence rates reveals no information in this case because the number of labels actually used must come close the number of sites in the end. Therefore, optimization runs on two maps, depicted on figures 5.4 and 5.5, are examined further on.

Discussion: Whereas in figure 5.4, nearly all settings of p_r lead to a full map, some sites remain unlabeled in case of the Northern Mariana Islands map 5.5. The ideal value of p_r seems to depend on the labeled map, but $p_r = 0.2$ suffices to succeed in most cases.

Experiment: all maps, $p_i \in 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$, number of total runs: 168

Analysis: Runs on the maps of the Northern Mariana Islands (figure 5.6) and of Louisiana (figure 5.7) are examined. Differences of the results for various values of p_i appear to be smaller than those received from changing p_r.

Discussion: As the fitness plots look similar for all tested values of p_i, a low initialization probability may be chosen to save time in the conflict evaluation.

Anyway, the value of p_r may be of higher relevance to the ongoing of the optimization.

[1] Several possibilities have been tested, for example: $\frac{fit}{r*10}$ and $\frac{fit}{r^2}$.

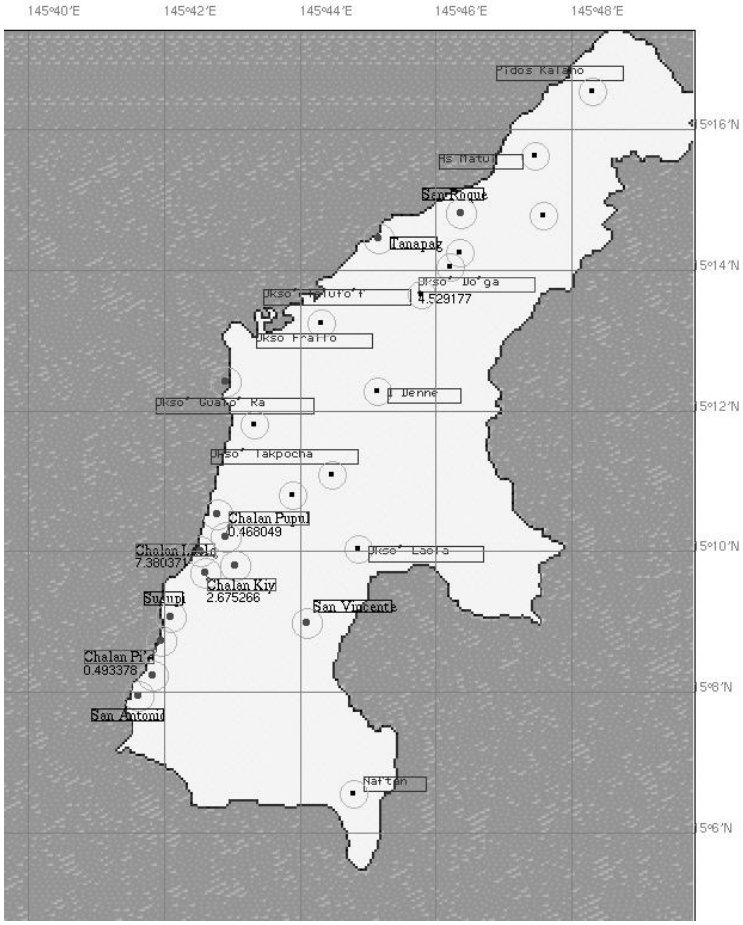

Figure 5.2: Initial state of the map of the Northern Mariana Islands, with p_r set to 0.5.

If the ES fails to find a complete map because none without conflicts exists, the incremental map assembly may be used for labeling tasks with point selection allowed. Due to selection of better maps, the omitted labels will mostly be the ones with the worst conflict values.

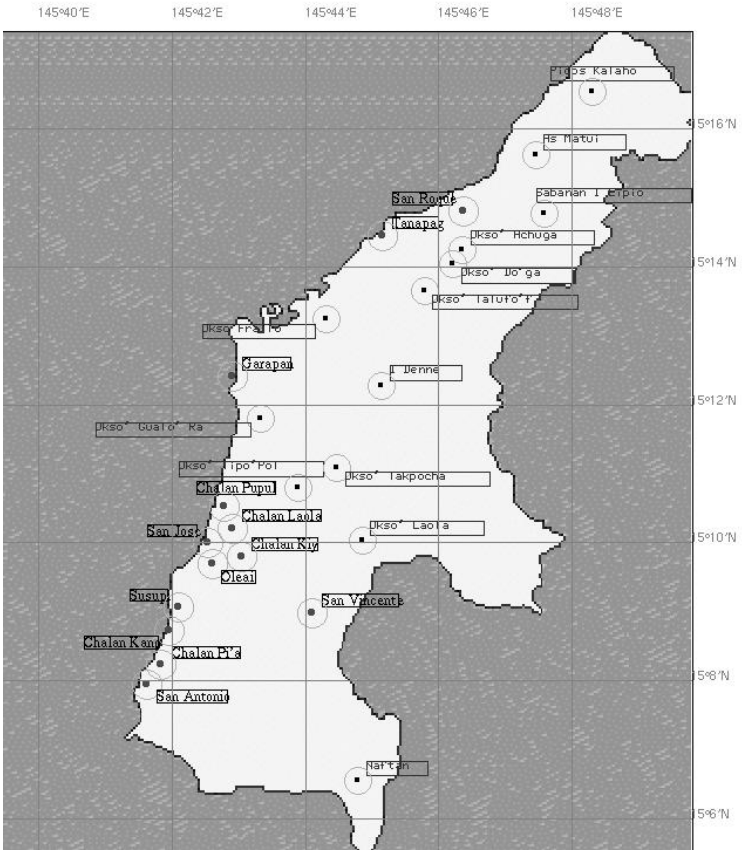

Figure 5.3: Last recorded map of the optimization that begun with the map in figure 5.2.

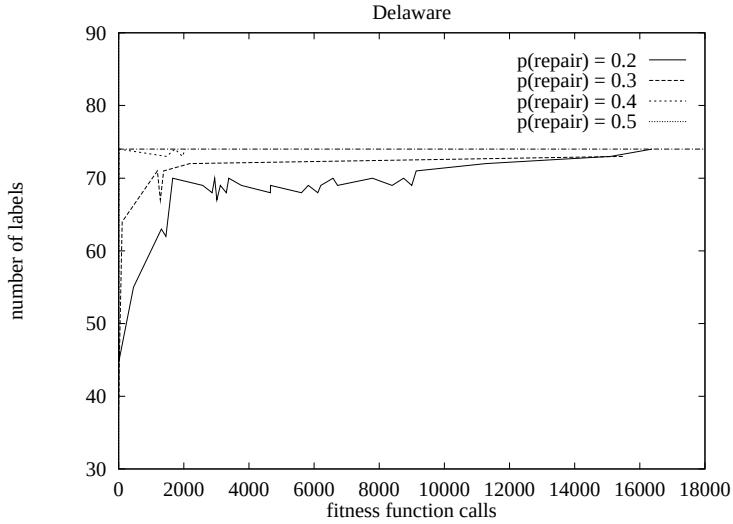

Figure 5.4: Integration of labels into the map of Delaware, testing different repair probabilities. The horizontal line at the upper marks the number of sites. Initialization probability p_i was set to 0.6 constantly.

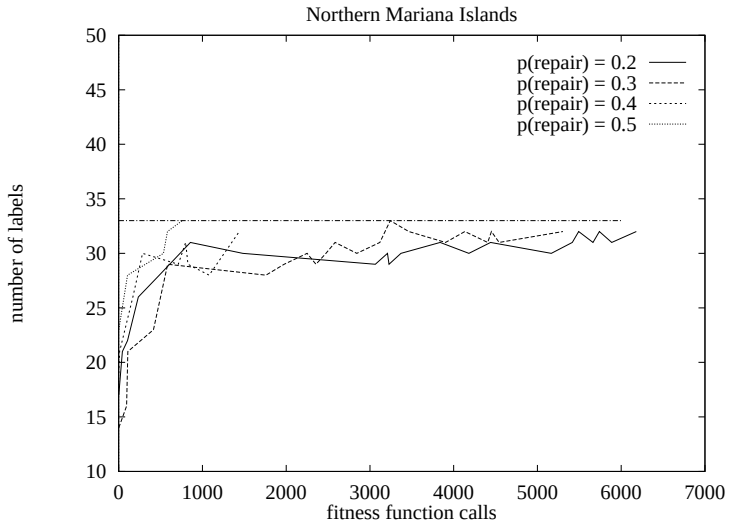

Figure 5.5: Integration of labels into the map of Marianas, testing different repair probabilities p_r. The horizontal line at the upper marks the number of sites. Initialization probability p_i was set to 0.6, constantly. Notice that lines end if a value remains stable until the optimization finishes.

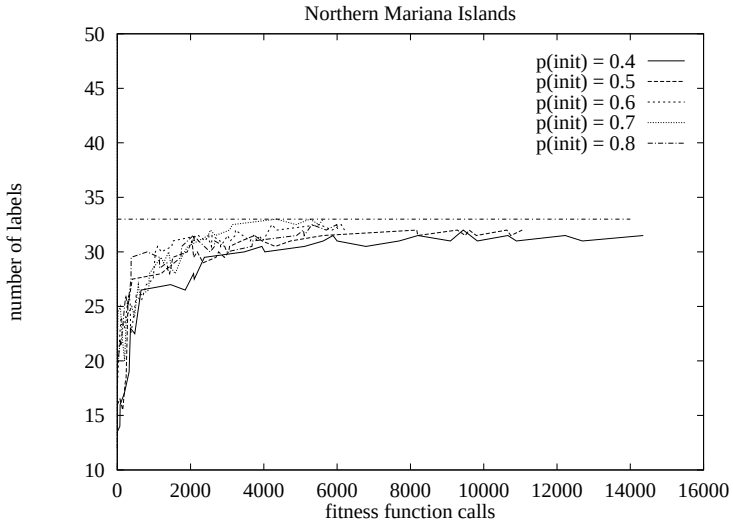

Figure 5.6: Integration of labels into the map of Marianas, varying the probability of initialization of a component while constantly setting p_r to 0.3. The horizontal line at the upper marks the number of sites.

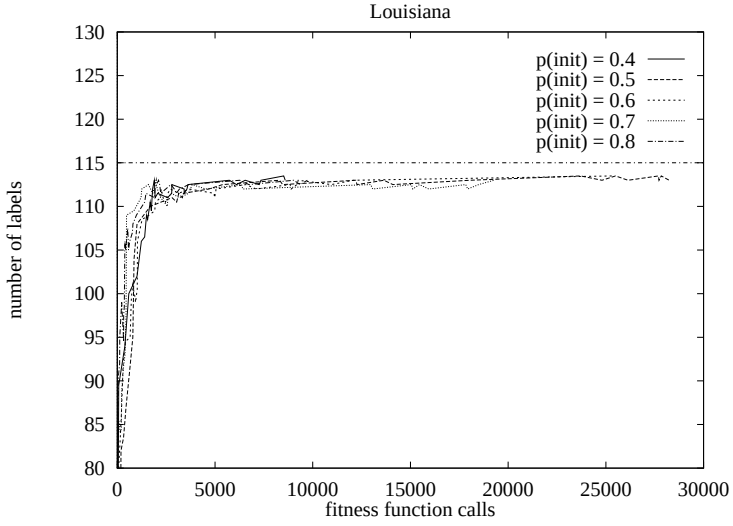

Figure 5.7: Integration of labels into the map of Louisiana, varying the initialization probability of each component. The horizontal line at $y = 115$r marks the number of sites. The repair probability has been constantly set to 0.3.

Chapter 6

Conclusion

The cardinal point this thesis dealt with is the question whether an algorithm based on a ES is capable of solving MLPs. *This can be answered positively.*

Within chapters 4 and 5, a great portion of the optimization runs succeeded in labeling their map (if a valid solution existed at all). Furthermore, some recommendations for robust ES parameters emerged from diverse experiments (table6.1).[1]

However, these conclusions depend on the used set of test maps. Another set may yield different results despite the fact that the used mixture of inland, coastal and island regions embodies a great diversity which adds some reliability to the outcome.

Due to restrictions mentioned in section 1.3, the algorithm can not be used for mapmaking in practice yet, but since the fitness function is modular, extensions only require inventing some additional terms. Integration of new point feature types is very easy, only affecting the map representation, not the algorithm. The algorithm is general as has been demanded — the modeling presented in chapter 2 may serve as foundation for alterations if emphasis on conflict avoidance of point features changes to other fields.

[1] Initialization and termination may be set meeting special needs or referring to [27].

parameter	meaning	recommended value
ω	recombination type	descrete for object variables and sigma values
ρ	number of parents of one individual	$\rho = \sqrt{\mu}$, (4 if $\mu = 15$)
τ	learning rate, used for mutations of σ_i	~ 0.3
sel	selection operator	κ - selection
κ	maximal lifetime of an individual	2 or 5, depending on λ.
μ, λ	parent and offspring numbers	$(15, 90)$, influenced by selection operator.

Table 6.1: Recommendations for ES parameters while solving MLPs.

Weight adjustment and settings of the minimal distances are optimized for testing and should be reconsidered by some person experienced with map-making. If the algorithm continues to find good labelings whenever being provided with test data of different sources, it is probably capable of solving real world problems as well.

Comparison with other methods described in the first chapter had to be omitted but should be carried out in future.

A statement concerning the use of *JAVA* : many problems arose from the "misuse" of this interpreted language for numerical optimization. Hence, the cardinal disadvantage with respect to this aim is its slowness while performing floating point calculation. Despite of these deficiencies, the language is well suited for object-oriented design and all kinds of applications requiring intense graphical interaction (as map-making does).

6.1 Where to go from here

The vast amount of time needed to perform a good labeling is the only reason that renders the application of the described algorithm more difficult. Therefore, it should be compiled with a not yet existing *JAVA*-bytecode compiler once all required functionality has been added. If this does not suffice for the needs of on-line map-making, the incremental approach of section 5.2 may be considered for further investigation. It is also applicable if automated point selection is permitted.

Another way to gain speed is parallelization. Without any changes to the algorithm, performance would be multiplied by parallel computation of the individuals. Other methods become available if changes to its structure are considered. Since the optimization of many small maps is of great advantage for the speed of their conflict evaluation, big ones may be split up if clusters are found easily.

This is a new problem which remains unsolved...

Bibliography

[1] Gene M. Amdahl. Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, AFIPS Press, Reston, Virginia, 1967.

[2] Henry K. Beller. Problems in Visual Search. In *International Yearbook of Cartography*, pages 137–144. Kirschbaum, Bonn Bad Godesberg, 1972.

[3] Jon Christensen, Joe Marks, and Stuart Shieber. Labeling Point Features on Maps and Diagrams. Technical Report TR-25-92, Harvard University, Cambridge, Massachusetts, December 1992.

[4] Jon Christensen, Joe Marks, and Stuart Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[5] Shawn Edmondson, Jon Christensen, Joe Marks, and Stuart Shieber. A General Cartographic Labeling Algorithm. *Cartographica*, (to appear), 1996.

[6] ARC/INFO - The World's GIS. Technical report, Environmental Systems Research Institute, Inc., Redlands, California, March 1995.

[7] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis (Dr.-Ing. Dissertation), Technical University of Berlin, Department of Process Engineering, 1974/75.

[8] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, Basel, 1977.

[9] Stephen A. Hirsch. An Algorithm for Automatic Name Placement Around Point Data. *The American Cartographer*, 9(1):5–17, 1982.

[10] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[11] Josef Hoschek. *Mathematische Grundlagen der Kartographie*. Bibliographisches Institut, Mannheim, 1984.

[12] Il'ja N. Bronštejn and K. A. Semendjajew. *Taschenbuch der Mathematik*. Deutsch, Thun, 1989.

[13] Eduard Imhof. Die Anordnung der Namen in der Karte. In *International Yearbook of Cartography*, pages 93–129. Kirschbaum, Bonn Bad Godesberg, 1962.

[14] Eduard Imhof. *Kartographische Geländedarstellung*. de Gruyter, Berlin, 1965.

[15] Ingo Rechenberg. *Evolutionsstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973. (Dr.-Ing. Dissertation, Technical University of Berlin, Department of Process Engineering, 1971).

[16] Kurt Mehlhorn. *Multi-dimensional Searching and Computational Geometry*. Data Structures and Algorithms 3. Springer, Berlin, 1984.

[17] Lawrence J. Fogel and A. J. Owens and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.

[18] Joe Marks and Stuart Shieber. The Computational Complexity of Cartographic Label Placement. Technical Report TR-25-92, Harvard University, Cambridge, Massachusetts, December 1992.

[19] Martin Schütz. Eine Evolutionsstrategie für gemischt-ganzzahlige Optimierungsprobleme mit variabler Dimension. Master's thesis, Universität Dortmund, September 1994.

[20] Paul Bühler. Schriftformen und Schrifterstellung. In *International Yearbook of Cartography*, pages 153–180. Kirschbaum, Bonn Bad Godesberg, 1961.

[21] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry. An Introduction*. Springer, New York, corr. and expanded 2nd print., 1988.

[22] Ralf Hartmut Güting. *Datenstrukturen und Algorithmen*. Teubner, Stuttgart, 1992.

[23] Günter Rudolph. An Evolutionary Algorithm for Integer Programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving From Nature, 3*, pages 139–148, Springer, Berlin and Heidelberg, 1994.

[24] Günter Rudolph and Joachim Sprave. Significance of Locality and Selection Pressure in the Grand Deluge Evolutionary Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving From Nature, 4*, pages 686–695, Springer, Berlin and Heidelberg, 1996.

[25] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object Oriented Modeling and Design*. Prentice-Hall, Eaglewood Cliffs, 1991.

[26] Vasco Alexander Schmidt. Reine Forschung, praktische Resultate. *DIE ZEIT*, (18):45, April 1995.

[27] Hans-Paul Schwefel and Günter Rudolph. Contemporary Evolution Strategies. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life, Third European Conf. on Artificial Life, Granada, Spain, June 1995*, pages 893–907, Springer, Berlin, 1995.

[28] Joachim Sprave. Evolutionäre Algorithmen zur Parameteroptimierung. *Information Processing Letters*, 43(3):110–117, 1995.

[29] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. PhD thesis (Dr. rer. nat.-Dissertation), Universität Dortmund, Department of Computer Science, September 1994.

[30] Thomas Bäck and Frank Kursawe. Evolutionary Algorithms for Fuzzy Logic: A brief Overview. In *Fifth Int'l Conf. IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems*, volume II, pages 659–664, Paris, July 1994.

[31] U.S. Department of the Interior, U.S. Geological Survey. *Geographic Names Information System–Data Users Guide 6*. Reston, Virginia, 1987.

[32] Frank Wagner. Approximate Map Labeling is in $\Omega(n log n)$. *Information Processing Letters*, 52:161–165, 1994.

[33] Frank Wagner and Alexander Wolff. An Efficient and Effective Approximation Algorithm for the Map Labeling Problem. In *Proceedings of the 3rd Annual European Symposium on Algorithms (ESA 1995)*, pages 420–433, 1995.

[34] Herbert Wilhelmy. *Kartographie in Stichworten*. Hirt, 1990.

[35] Alexander Wolff. Map Labeling. Master's thesis, Freie Universität Berlin, 1995.

[36] Chyan Victor Wu and Barbara P. Buttenfield. Reconsidering Rules for Point-Feature Name Placement. *Cartographica*, 28(1):10–27, spring 1991.

[37] Pinhas Yoeli. The Logic of Automated Map Lettering. *The Cartographic Journal*, 9(2):99–108, December 1972.