On the design and analysis of competent GAs

Steven van Dijk Dirk Thierens Mark de Berg*

28th March 2002

Abstract

We study two recent theoretical models—a population-sizing model and a convergence model—and examine their assumptions to gain insights about the conditions under which GAs work well. We use these insights to formulate several design rules to develop competent GAs for practical problems. We then use these rules to design a GA that solves the map-labeling problem, an NP-hard problem of real-world significance. Finally, we test whether the fact that our GA followed the design rules inspired by the theoretical models results in a scale-up behavior as predicted by these models. Experiments show that this is indeed the case.

1 Introduction

Genetic algorithms have been applied to solve an impressively wide range of problems. Although they have proven to be very flexible, many successful GAs are found by making educated guesses for the representation, parameters and operators. As a result, GA design is sometimes seen as a black art, not as an engineering task with a solid theoretical basis. In part, this is caused by the complexity of the behavior of the algorithm, which is difficult to model in its entirety. Several different theoretical approaches are being pursued, such as the facet-wise composition of partial models (Goldberg, 1999), the exact analytical models based on Markov-chains (Vose, 1999; Rowe, 2001), formal proofs of GA behavior (Jansen and Wegener, 2001), models based on statistical mechanics (Shapiro, Prügel-Bennett, and Rattray, 1994), approaches bases on the estimation of distributions (Mühlenbein and Paass, 1996; Pelikan, Goldberg, and Lobo, 2002), exact schema theorems (Poli, 2000), and coarse-grained analysis of building-block evolution (Stephens and Waelbroeck, 1999).

These approaches offer valuable insights in various important aspects of genetic algorithms, but do not all translate easily to practical design rules. Indeed, practitioners dealing with real-life problems may feel that current theoretical results are too far removed from the practical realities to be of any use.

In this paper, we study this gap between theory and practice, following the approach of facet-wise design as advocated by Goldberg (1999). Specifically, we examine two recently developed models, and see what they can teach us about successful GA design. We translate these lessons in practical design rules for building a competent GA. We use these rules to design a GA for a practical problem that is NP-hard, namely the map-labeling problem: placing the names of map elements on a cartographic map. Next,

*Institute of Information and Computing Sciences, Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands. Email: {steven, Dirk.Thierens, markdb}@cs.uu.nl

we turn around and examine whether the results of the GA match with the predictions of the models. This allows us to evaluate the practical usability and significance of the models.

A GA can be called *competent* if it is able to find good solutions for the problem at hand in reasonable time. "Good" can be defined as finding solutions with quality that is a certain percentage (say, 97%) of the optimum. "Reasonable" means that the algorithm should *scale up* well (Thierens, 1999). The scale-up behavior of an algorithm is the relation between input size l and the amount of computational effort W required to find a solution with the requested quality. The amount of computational effort the GA spends is the product of the number of fitness evaluations E and the time needed to perform a single fitness evaluation e_{fit}: $W = E \cdot e_{fit}$. The optimal number of fitness evaluations E is also the product of two factors: the critical population size and the number of generations it takes to converge: $E = n^* \cdot t^*$, where n^* is the critical population size needed to obtain a solution of a certain quality, and t^* is the number of generations until convergence when the GA uses a population that is sized large enough ($n \geq n^*$). Both factors (n^* and t^*) therefore determine the scale-up behavior of the number of fitness evaluations spent by the GA.

In this paper, we discuss two models from literature—a population-sizing model (Goldberg, Deb, and Clark, 1992; Harik, Cantú-Paz, Goldberg, and Miller, 1999) and a convergence model (Mühlenbein and Schlierkamp-Voosen, 1993; Thierens and Goldberg, 1994a; Bäck, 1995; Miller and Goldberg, 1996)—that together allow the prediction of the scale-up behavior of the GA. Both models predict a scale-up of the square root of l, yielding a prediction of a linear scale-up for the number of fitness evaluations. Unfortunately, these models have only been tested for artificial problems of *bounded difficulty*.[1] It is implicitly assumed that the concept of bounded difficulty is relevant for many practical problems, too. In this paper we will strengthen this claim by showing how good solutions can be found for the map-labeling problem by treating it as a problem of bounded difficulty.

The map-labeling problem comes from automated cartography and is defined as follows. Given is a map of cities and their names. Each name has to be placed on the map next to the city. The *label* of a city is the rectangular bounding box of its name when printed in a certain font and font size. The label can be placed with the city in one of the four corners—in other words, the label can be placed in either the top-right, top-left, bottom-right or bottom-left position. Find a *labeling* (a position for each label) such that the number of non-intersecting labels is maximized. A solution for a map of 1000 points is shown in Figure 1. This problem instance has been shown to be NP-hard (Formann and Wagner, 1991; Marks and Shieber, 1991). The full map-labeling problem is more complex and contains many additional cartographic constraints and additional feature types (for rivers and areas). A more thorough treatment of the map-labeling problem is beyond the scope of this paper. The interested reader is referred to Van Dijk's thesis (2001).

We will develop a GA that finds solutions for instances of the map-labeling problem, based on the issues that were raised by the theoretical models. Note that in the case of the map-labeling GA on maps of fixed density, the term e_{fit} is easy to bound: in the fitness function each label on the map is checked for an intersection in constant time (by checking for an intersection with the labels of a bounded number of neighboring cities). Therefore, the total time needed for a single fitness evaluation is $e_{fit} = O(l)$.

[1] We will say more about problems of bounded difficulty in Section 2.

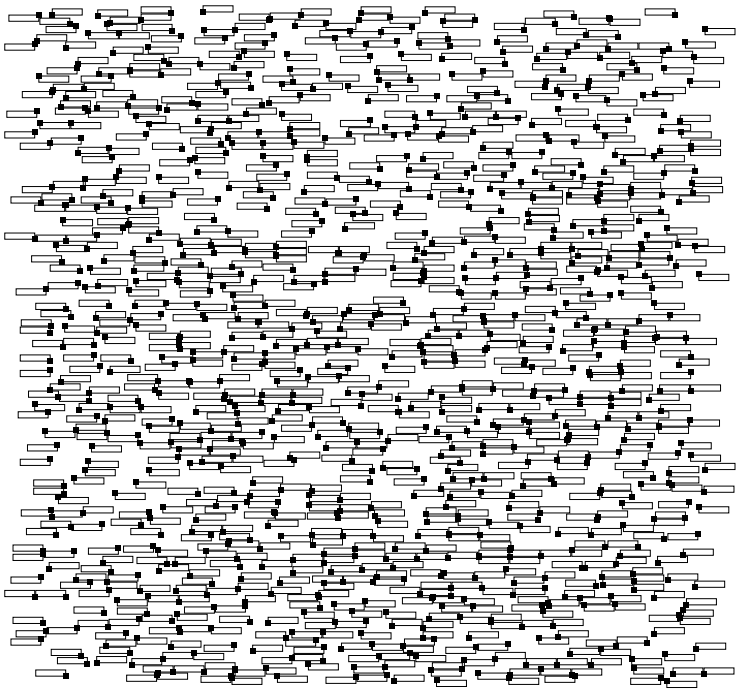

Figure 1: An example of a labeled map.

Combining this with the linear scale-up for the number of evaluations gives a quadratic scale-up for the amount of computational effort: $W = O(l^2)$.

Next, we will use the GA to evaluate the usability of the models, by experimentally testing whether the predictions actually hold up. This will demonstrate whether the models are interesting in a practical setting too. Indeed, the experimental results match the predictions.

Our main contributions are as follows. Firstly, we demonstrate the practical usefulness of two recent theoretical models and the concept of bounded difficulty. We extract the underlying assumptions of the models to gain insights about the conditions under which a GA performs well. Secondly, we show how these insights translate in several practical design rules. Using the rules, we design a GA that solves the map-labeling problem. We use the GA to experimentally evaluate the practical usability of the models under real-world conditions.

This article is structured as follows. Figure 2 shows the relations between theory and practice we will explore in this paper. In Section 2, we will briefly describe the two models from literature. Next, in Section 3, we formulate the design rules and use them to design an efficient GA for the map-labeling problem. The GA is not able to satisfy the underlying assumptions of the models as well as the artificial problems on which the models were originally tested. However, since we were guided by the theoretical principles in the design phase, we expect no serious deviations from the assumptions and the predictions to hold. Section 4 is devoted to a verification of this expectation. Firstly, we systematically check the model assumptions to see how much the GA deviates from them. Secondly, we run the GA on dense, randomly-generated maps and find the experimental critical population size and number of generations until conver-

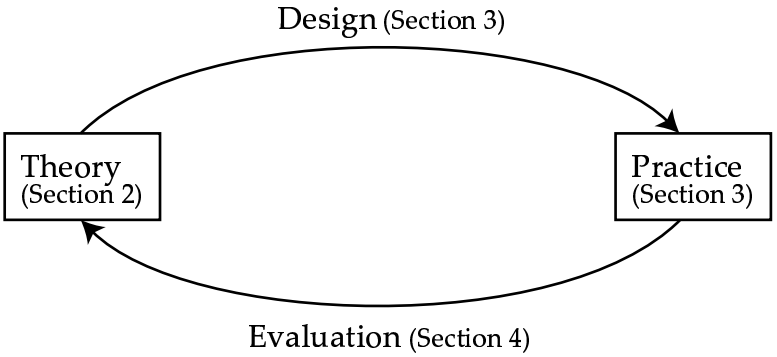

Figure 2: The relation between theory and practice.

gence occurs. These match the predictions of the models. As a result, the total number of fitness evaluations scales linearly with respect to the input size. Some concluding remarks are made in Section 5.

2 The models

Before examining the theoretical models, we have to discuss what is meant in this paper by "problems of bounded difficulty". Such problems can be solved by combining the best schemata of partitions of bounded size (Kargupta, 1995; Harik, 1997). Specifically, in this paper we will consider problems of bounded difficulty that can be solved by a GA whose fitness function is assumed to be additively decomposable, uniformly scaled and (semi-)separable. An *additively decomposable function* (Mühlenbein and Mahnig, 1999) can be expressed as the summation of the contributions of the parts of the solution. More precisely, the function is a summation of partial fitness functions that only depend on a few genes each. For example, given a solution $\mathbf{x} = x_1x_2x_3x_4x_5$, the function $f_{fit}(\mathbf{x}) = f_1(x_1x_2) + f_2(x_3x_4) + f_3(x_5)$ is an ADF. If the different functions $f_i(\cdot)$ all depend on different genes, as in the example above, an ADF is called *separable*. It is also useful to consider the case where a fitness function is "almost" separable, which we will call *semi-separable*: an additively decomposable function is defined as semi-separable if each gene is input to only a small, bounded number of partial fitness functions. For example, $f_{fit}(\mathbf{x}) = f_1(x_1x_2x_3) + f_2(x_2x_3x_4) + f_3(x_4x_5)$ is a semi-separable ADF in which each gene occurs in maximal two partial functions. The ADF is uniformly scaled if the functions $f_i(\cdot)$ have the same distribution of values.

The fitness function used to solve a problem of bounded difficulty can thus be expressed as follows:

$$f_{fit}(\mathbf{x}) = \sum_{i=1}^{m} f_i(x_{i,1}, x_{i,2} \ldots x_{i,k}),$$

where partial functions f_i are defined on at most k genes, with $k \ll l$. Additionally, the functions $f_i(\cdot)$ all depend on different genes and have the same distributions of values.

The bit-counting problem ($f_{fit}(\mathbf{x}) = \sum_{i=1}^{l} x_i$) is the most simple instance in this class of functions. A characteristic example of a problem that involves higher-order relations (defined over multiple genes) is the concatenated trap-function problem. It is defined

as follows. Chromosomes use a binary alphabet and are $l = k \cdot m$ long, where m is the number of trap functions and k is a constant.

We define a *trap function* as follows:

$$f_{trap}(\mathbf{x}_{i+1\ldots i+k}) = \begin{cases} k & \text{if } u(\mathbf{x}_{i+1\ldots i+k}) = k \\ k-1-u(\mathbf{x}_{i+1\ldots i+k}) & \text{otherwise} \end{cases},$$

where $\mathbf{x}_{i\ldots j}$ is shorthand for $x_i x_{i+1} \ldots x_{j-1} x_j$ and $u(\mathbf{x}_{i\ldots j}) = \sum_{t=i}^{j} x_t$.

Trap functions with $k > 3$ have also been called *deceptive function*s (Goldberg, 1989a; Deb and Goldberg, 1994), because information from lower-order partitions (defined over less than k genes) leads away from the optimal schema. In order to find the optimal schema of the partition defined on the genes that are input to the trap function, schemata in the partition should not be disrupted during crossover. In other words, there exists strong linkage between those genes.

The fitness function for the concatenated trap-function problem is a concatenation of m trap functions:

$$f_{fit}(\mathbf{x}) = \sum_{i=0}^{m-1} f_{trap}(\mathbf{x}_{i\cdot k+1\ldots i\cdot k+k}) .$$

Each trap function is defined on k genes and introduces linkage between those genes. The optimal solution can be found by combining the best schema of each partition defined over linked genes. For example, if $k = 4$ and $m = 2$, these partitions are simply FFFF#### and ####FFFF, where "F" denotes a fully specified gene and "#" denotes the "don't care"-character (Goldberg, 1989b). The optimal solution 11111111 can be found by searching the best schema in each partition (namely, the schemata 1111#### and ####11111) and combining them.

The remainder of this section examines two models from the literature. We extract the underlying assumptions and will use them in the next section to formulate several design rules. We start in Subsection 2.1 with the convergence model to find t^*, the number of generations until convergence. It is assumed the population size is adequately sized. Subsection 2.2 will cover the gambler's-ruin model, which deals with the critical population size n^*—that is, the minimal population size needed to find a solution with a certain level of quality.

2.1 Determination of t^*

There have been several studies (Mühlenbein and Schlierkamp-Voosen, 1993; Thierens and Goldberg, 1994a; Bäck, 1995; Miller and Goldberg, 1996) on the convergence characteristics of GAs that solve the bit-counting problem, which is to find a bitstring of length l with the maximal number of 1's. It is a very useful problem to study because its properties (for example the distribution of fitness values in a randomly-generated population) can be calculated exactly. Furthermore, it has building blocks of only one gene, which means that no disruption can occur. Using uniform crossover, almost perfect mixing can be obtained. Mixing is called *perfect* when no correlation between building blocks—which was introduced by selection—remains after crossover.

Mühlenbein and Schlierkamp-Voosen (1993) analyzed the convergence time—the number of generations to obtain convergence to the optimal string—of a GA that solves the bit-counting problem using truncation selection, uniform crossover, no mutation, and assuming a properly-sized population. Uniform crossover is used because no disruption of building blocks can occur for this problem, and it mixes the building blocks

well. Crossover is always applied. The rate of convergence of the GA is primarily determined by the selection pressure of the selection scheme. This can be quantified as the *selection intensity* at generation t (denoted by $I(t)$) as follows (Bulmer, 1980):

$$I(t) = \frac{\mu_{sel} - \mu(t)}{\sigma(t)}, \tag{1}$$

where μ_{sel} is the mean fitness of the selected individuals, and $\mu(t)$ and $\sigma(t)$ are the mean and the standard deviation of the fitness of the population at generation t, respectively. Since crossover does not change the proportion of 1's—that is, the building blocks—in the population, the mean of the fitness of the new population $\mu(t+1)$ is equal to μ_{sel}. Under the assumption of perfect mixing, the population fitness is binomially distributed, which can be approximated well with a normal distribution, giving values for $\mu(t)$ and $\sigma(t)$. The use of a rank-based selection scheme implies that the selection pressure is constant. Therefore, the selection intensity $I(t)$ is equal for all t and is just denoted as I. The selection intensity I depends on the truncation constant used. For example, if $T = 80\%$, $I = 0.34$.

The following result was obtained (Mühlenbein and Schlierkamp-Voosen, 1993) for t^*, the number of generations until convergence:

$$t^* = \frac{(\frac{1}{2}\pi - c)\sqrt{l}}{I} = O(\sqrt{l}).$$

where l is the length of the chromosomes, and c is a constant depending on the number of building blocks in the initial population.

Thierens and Goldberg (1994a) investigated other selection schemes, such as tournament selection, and the elitist recombination scheme (the latter is discussed in Section 3). All rank-based schemes were found to have $t^* = O(\sqrt{l})$. In contrast, for proportionate selection, $t^* = O(l \log l)$ holds. This suggests that the above result holds for all selection schemes which are rank-based. Bäck (1995) considered (μ, λ)-selection and tournament selection, and used order statistics to generalize the results for different selection intensities.

Miller and Goldberg (1996) extended this research by considering noisy fitness functions. Furthermore, they also applied the model to more complex problem domains than the bit-counting problem. They derived equations for domains in which the mean and the standard deviation of the fitness distribution can be expressed as functions of the proportion of converged building blocks. For the more complex domain of concatenated trap functions an approximation was used.

Miller and Goldberg found that their prediction of the convergence behavior for the concatenated trap function closely matched experimental results. The requirement of separability can be relaxed to semi-separability by modeling the interactions between different partitions[2] as noise. Miller and Goldberg showed that adding small levels of noise to the fitness function added a constant to the number of generations until convergence. Therefore, as long as the linkage between genes from different partitions is weak, the model gives a good approximation.

This indicates that the prediction $t^* = O(\sqrt{l})$ holds for GAs that, in the ideal case, satisfy the following assumptions:

- The fitness function is additively decomposable, uniformly scaled, and (semi-)-separable.

[2]Unless otherwise specified, "partitions" refers to the partitions that hold the building blocks which are being mixed by the GA.

- The selection scheme is rank-based.
- Mixing is perfect: no correlations remain between genes of different partitions after crossover.
- There is no disruption of building blocks.

For the case where the fitness function is exponentially scaled (instead of uniformly), similar studies (Thierens, Goldberg, and Pereira, 1998; Lobo, Goldberg, and Pelikan, 2000) show that the number of generations is linear with respect to the input size: $t^* = O(l)$.

In the models above, it is assumed that the population size is large enough and contains sufficiently many building blocks. The next section covers models that deal with population sizing and building-block supply.

2.2 Determination of n^*

The issue of determining n^*, the minimal population size needed to solve a problem of input length l, was investigated by Goldberg et al. (1992). They provided a model of the GA based on statistical decision making. Assuming that the GA would find the best solution if the search progressed in the right direction after the first generation, they obtained a population-sizing equation. Drawbacks of this approach were that it did not model the way a GA can recover from decision errors (explained later) and did not include a building-block supply model. Harik et al. (1999) extended the model by combining the so-called gambler's-ruin model and a building-block supply model with the previous model.

The model views the search process as a propagation of building blocks through the population, assuming mixing is adequate. A building block is the schema in a certain partition with the highest fitness.[3] Since decisions are made on the level of strings, a competition between a string matching a building block and a string matching another (sub-optimal) element from the same partition can result in the loss of the building block. Such an event is called a decision error. Since building blocks have a higher fitness, on average they will win the competition. The probability of making the right decision Pr_{ok} in a competition between two chromosomes was derived by Goldberg et al. (1992):

$$Pr_{ok} = \Phi(\frac{d}{\sqrt{2(m-1)}\sigma_{part}}),$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, d is the difference between the fitness of the building block and its competitor (the schema with the next-highest fitness), m is the number of partitions, and σ_{part} is the standard deviation of the fitness contribution from a single partition.

The gambler's-ruin model views the search of a GA in a single partition as a series of competitions which progresses until either all individuals in the population match the building block, or none do. The outcome is dependent on the population size and the initial number of building blocks in the population. The model is a one-dimensional random walk between absorbing barriers, corresponding with the loss of the building block (no building blocks left; we call this the *depletion barrier*) and the existence of the building block in all individuals (n building blocks in the population; we call this

[3] The fitness of a schema is the average of the fitnesses of all possible chromosomes that match it.

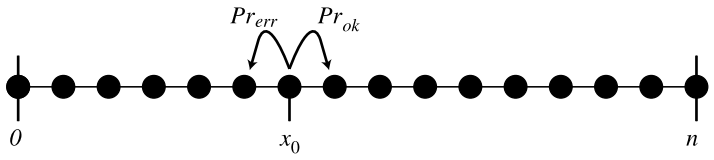

Figure 3: The gambler's-ruin model.

the *saturation barrier*). The walk starts at x_0, the number of building blocks in the initial population. Each competition advances the walk to either the saturation barrier (the string with the building block wins the competition) which increases the number of building blocks, or the depletion barrier (a decision error) which decreases the number of building blocks. Figure 3 depicts the situation after initialization, before the first competition.

The initial number of building blocks is given by x_0. A simple building-block supply model assumes each element of the partition is equally likely to be (randomly) created. Therefore, if only one element of the partition is a building block, the number of genes in the partition is k and the alphabet size is denoted by A, the expected number of building blocks in the initial population of size n is n/A^k.

The notion of competitions in the gambler's-ruin model corresponds most naturally to an incremental GA. However, the experimental results by Harik et al. show that the more conventional generational replacement scheme also agrees well with the model. As a result, we can assume that any selection scheme with constant selection pressure suffices. This implies a rank-based selection scheme, such as tournament selection or the elitist recombination scheme.

We now turn to the question of determining the probability $Pr(n)$ of the gambler eventually hitting the saturation barrier using a population of size n. This is a known result from random-walk literature (Feller, 1966):

$$Pr(n) = \frac{1-(\frac{1-Pr_{ok}}{Pr_{ok}})^{x_0}}{1-(\frac{1-Pr_{ok}}{Pr_{ok}})^{n}}. \tag{2}$$

The probability $Pr(n)$ for a single partition can be used to calculate the outcome of all the partitions searched in parallel. Hence, the following holds for n_{conv}, the number of partitions that converge to the building block:

$$E[n_{conv}] = m \cdot Pr(n), \tag{3}$$

where m is the total number of partitions and $E[\cdot]$ denotes the expected value.

Given a measure of quality $\alpha = \frac{n_{conv}}{m}$ that denotes the desired fraction of found building blocks, we can find the critical population size to obtain that result. Extracting n from Equation 3, assuming a binary alphabet, and approximating Pr_{ok} gives the following approximation of a population-sizing equation (Harik et al., 1999):

$$n \approx -2^{k-1}\ln(1-\alpha)\frac{\sigma_{part}\sqrt{\pi(m-1)}}{d}. \tag{4}$$

Note that $\lim_{\alpha\uparrow 1}\ln(1-\alpha) = -\infty$. In other words, finding the optimal solution with absolute certainty requires an infinite population size. This is only to be expected, since a GA is a stochastic algorithm. Harik et al. performed experiments to test their model

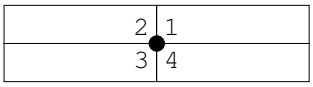

Figure 4: The four possible positions where a label can be placed.

on various domains, including the concatenated trap-function problem with overlapping partitions (they share genes). They found that the model gave a good estimate of the relation between the quality of solutions (expressed in α) and the population size. The good results on the domain with overlapping partitions show that the assumption of separability of the fitness function can be relaxed to semi-separability.

To find the relation between input size l and the optimal population size, for a certain level of quality α (for example 0.97), we observe that k, σ_{part}, and d are constants and $l = \Theta(m)$. Hence, given some α, we have

$$n^* = O(\sqrt{l}),$$

where n^* is the *critical* population size needed to find a solution with the required level of quality.

This is expected to hold for GAs that adhere to the assumptions of the gambler's-ruin model. These assumptions (again, for the ideal case) are now restated for convenience:

- The fitness function is additively decomposable, uniformly scaled, and (semi-)-separable.
- The order of partitions, k, is a fixed constant, with $k \ll l$.
- All building blocks are present in the initial population.
- The selection scheme is rank-based.
- Mixing is perfect: no correlations remain between genes of different partitions after crossover.
- There is no disruption of building blocks.

Note that these assumptions subsume the assumptions from the convergence model.

These assumptions may appear to be quite strict. One of the purposes of this paper is to examine whether this is really the case. How much can one deviate from these requirements before the models no longer give meaningful predictions of the GA's behavior? Are GAs that were designed to solve real-world problems (as opposed to artificial problems such as the concatenated trap function) still able to adhere to these assumptions? In the next sections, we will show that it is indeed possible for a practical GA to adhere to these assumptions close enough that the models still give valuable predictions.

3 Design of a GA for the map-labeling problem

The overview of the models from the previous section showed the underlying assumptions about the problem and the GA. As such, it provides us with several insights that

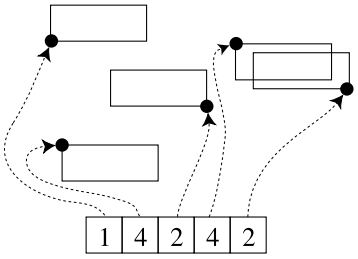

Figure 5: The encoding for a map.

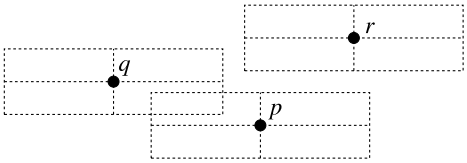

Figure 6: Cities p and q are rivals, but p and r are not.

we can turn into design rules. The rules we will formulate in this section are reminiscent of the guidelines of Goldberg (1999), but are more explicit. We will use them to design a GA that will find solutions for instances of the map-labeling problem. Recall that the map-labeling problem consists of placing a label in one of four positions (see Figure 4) for a set of points such that the number of labels that do not intersect another label is maximized. We denote the number of points on the map with n_{pts}. The GA that will solve the map-labeling problem will represent a labeling by a string of numbers between one and four, indicating positions. Each city has an index which indicates its position in the string. See Figure 5.

We will now state the design rules that we can derive from the models:

1. **Identify building blocks.** Both models show a strong dependency on the notion of a building block. The convergence model defines convergence as the state in which each partition is converged to the building block. The population sizing model views the search as the parallel growth of building blocks in the separate partitions. To construct a competent GA, it is necessary to know what the building blocks of the solution are. More generally, the designer needs to make an assessment of the linkage of the representation. The identification of the linkage of the problem is a necessary first step that will be used in the application of the following rules.

   Map labeling is interesting in that the linkage of the problem is reasonably clear since it can be inferred from the geometry. Given a representation in which every point on the map corresponds with a gene on the chromosome, linkage between two genes can be suspected when the two corresponding points are close together. Thus, we assume the building blocks consist of good labelings of a city and several close-by cities. To make this more concrete, we define two points as *rivals* if their labels can intersect (see Figure 6). A *rival group* consists of a certain point and its rivals. We assume the best labelings for rival groups are building blocks. Note that we make an assessment of the linkage which will likely neglect higher-order relations. As this is a NP-complete problem, that is necessary to obtain a favorable trade-off between solution quality and running time. We hypothesize that close-to-optimal solutions can still be found when the problem is treated as having bounded difficulty by using the

lower-order relations that are defined by the rival relationship.

Note that for other problems the linkage is not so easily found. In that case, one has either to learn the linkage (Kargupta, 1996; Harik, 1997; Munetomo and Goldberg, 1999; Pelikan et al., 2002), or resort to traditional trial-and-error.

2. **Use a fitness function that is additively decomposable, uniformly scaled and (semi-)separable.** The models demand the fitness function to be in a certain form. For some problems, the designer may have little choice in the form of the fitness function. In other cases, an effort can be made to keep the fitness function in the required form. The map-labeling problem is a good example. Let us consider another GA for the map-labeling problem. The GA by Verner, Wainwright, and Schoenefeld (1997) uses the following fitness function (to be minimized):

$$f_{fit}(\mathbf{x}) = w_1 \cdot overlap(\mathbf{x}) + w_2 \cdot area(\mathbf{x}) - w_3 \cdot distFact(\mathbf{x}).$$

The number of overlapping labels is measured by $overlap(\cdot)$ and the total area of all overlapping labels is given by $area(\cdot)$. The function $distFact(\cdot)$ measures the *distance factor* of a point feature with an overlapping label, defined as the sum of distances from the center of its label and the centers of the labels from the four nearest features. The distance factor of a point feature with a non-intersecting label is zero. Verner et al. say that "the values for w_1, w_2 and w_3 were arrived at by *numerous* trial and error testing" (emphasis added and notation changed). Tuning the weighting factors $w_1 \ldots w_3$ takes a lot of time and makes the algorithm inflexible. In addition, the fitness function has lost the preferred characteristics of being uniformly scaled.

For these reasons, the fitness function of our map-labeling GA just counts the number of *free* labels. A label is free when it does not intersect any other label. For example, the fitness of the little map of Figure 5 is 3. The fitness function can be expressed as a uniformly scaled, semi-separable ADF:

$$f_{fit}(\mathbf{x}) = \sum_{i=1}^{n_{pts}} free(\mathbf{x}_i)\,, \tag{5}$$

where n_{pts} denotes the number of points on the map, and $\mathbf{x}_i$ denotes the genes corresponding to the rival group of the i'th feature. The function $free(\mathbf{x}_i)$ returns 1 if the label of point i is free, and 0 if it intersects another label. Hence, this fitness function is uniformly scaled.

3. **Use a rank-based selection scheme.** The models depend on the assumption of constant selection pressure, which implies a rank-based selection scheme. For our map-labeling GA, the elitist recombination scheme (Thierens and Goldberg, 1994b) was used. In this rank-based scheme, two parents are chosen randomly from the population. Crossover is always performed, and two children are generated. From this family of four, the two best individuals replace the parents in the population. In the case of ties, children precede their parents. The scheme is conceptually simple, easily implemented, and simplifies the structure of the GA. In addition, it preserves good solutions by having elitism on the family level. The parameter Pr_c, which denotes the probability that crossover is applied, is set to 1.0, because there is no danger of losing fit parents.

4. **Ensure good mixing of building blocks.** Fast mixing of building blocks is critical to the success of the GA. Combined with the assumption of an additively decomposable fitness function, it allows an appeal to the Central Limit Theorem to use a normal approximation of the distribution of fitnesses in the population. Ideally, after crossover no correlations between building blocks exist. This is best achieved by using what

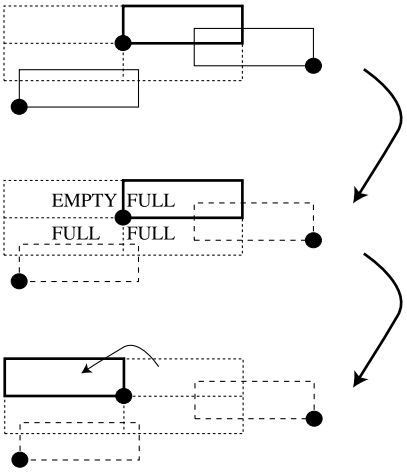

Figure 7: The geometrically local optimizer resolves conflicts after crossover.

Miller and Goldberg (1996) called a "uniform building block crossover"—a crossover that chooses at random for each partition in which a building block can reside whether to copy from either parent.

The assessment of linkage as the rival relationship allows us to design a crossover operator for the map-labeling GA that mixes on the level of building blocks. The crossover operator works by generating a set S of points. The labelings of points in S are transfered from the first parent to the first child. Labelings of points not in S are taken from the other parent. The other child is constructed in a complementary fashion—that is, the first child inherits labelings of points in S from the *second* parent, and the rest from the first parent. The crux lies, of course, in the method to construct the set S. This is done by randomly picking a point on the map and placing its rival group in the set. Since we assess the linkage as the rival relationship, the rival group is a partition which can hold a building block. If the local labeling of the chosen point and its rivals is indeed a building block, it is transfered undisrupted to the child. The next point is again chosen randomly, and this process is repeated until the size of S exceeds half the total number of points on the maps.

5. **Minimize disruption.** Both models assume that no disruption of building blocks takes place, so naturally it is important to minimize disruption as much as possible. One can either do this by mixing less aggressively, sometimes copying whole chromosomes (by setting $Pr_c < 1$), or repairing disrupted building blocks after crossover. We use the latter method for our example GA.

The geometry of the map-labeling problem (the rival relationship) helps in identifying locally bad solutions. After crossover, points which came from S but had a rival not in S (and vice-versa) can have new conflicts, making the crossover rather disruptive. On these points a so-called *geometrically local optimizer*, or GLO for short, is applied, which tries to resolve the conflict in two phases (see Figure 7). Firstly, it determines the status of each candidate label position. The status is EMPTY if the label is free when it is placed there. It is FULL otherwise. Secondly, from the EMPTY positions one is randomly chosen. If no position is EMPTY, nothing changes.

The role of the GLO can be viewed in a more general setting as a way to supply the GA with building blocks which may have been disrupted by crossover. Note that

this is only possible because the geometry of the problem allows us to say whether a certain change (such as placing a label in an empty slot) will locally improve the solution. In the more general case of the full map-labeling problem, cartographic rules will be enforced on a local scale, and the GLO still acts as a supply of building blocks which the GA uses to build a globally good solution. More details on GAs for more complicated map-labeling problems are given elsewhere (Van Dijk, 2001). The use of the GLO makes the GA more flexible. For example, it keeps the fitness function simple and uniformly scaled by enforcing cartographic rules in the GLO instead of specifying them as penalty functions in the fitness function.

The GLO renders mutation in the traditional sense (assigning a random allele to a randomly chosen gene) obsolete. Therefore, the probability of mutation Pr_m is set to zero.

6. **Ensure good building-block supply.** Both models assume a good supply of building blocks. Two well-known mechanisms of building-block supply are initialization and mutation. Initialization forms building blocks in the initial population and mutation introduces them during the run of the algorithm. The GLO is a more explicit kind of building-block supply than blind mutation, since it cannot make the solution worse. Our use of the GLO allows us to do away with the traditional kind of mutation and places modest demands on the initialization operator. Since the GLO will introduce building blocks during the run, there is no need for all building blocks to be present in the initial population. Therefore, for initialization, we assign a random position to each label of each individual in the population. As will become apparent in Subsection 4.2, the population size can be kept small.

Comparison with a simple GA

In Figure 8, we compare our GA using the so-called "rival" crossover against simple GAs using uniform and one-point crossover. The graphs show the average of five runs on five different maps. The last point of each graph shows the standard deviation of the fitnesses of the best solutions of all runs. The maps are created as explained in Subsection 4.2, and place 1000 points on a grid of 650 units squared (an example of such a map is shown in Figure 1). The comparison is done using the number of label-intersection tests, as this is a direct measure of computational effort. Our GA is able to find near-optimal solutions on the dense, randomly-generated test data. More extensive comparisons are beyond the scope of this paper and can be found elsewhere (Van Dijk, 2001).

Comparison with other map-labeling algorithms

A comparison of several map-labeling algorithms was done by Christensen, Marks, and Shieber (1995). Their conclusion was that the best results were obtained by a simulated-annealing algorithm. Figure 9 shows the experimental results of running the algorithms on randomly-generated maps. Note that these maps differ from those we use in the rest of this paper, since they are created by placing an increasing number of points in an area of fixed size. Therefore, the optimal number of free labels is not known, unlike with our maps. Since the comparison by Christensen et al. (1995), several other algorithms have been proposed, such as genetic algorithms (Verner et al., 1997; Raidl, 1998), tabu-search (Yamamoto, Lorena, and Camâara, 1999), and heuristics for maximum independent set (Strijk, Verweij, and Aardal, 2000; Verweij, 2000;

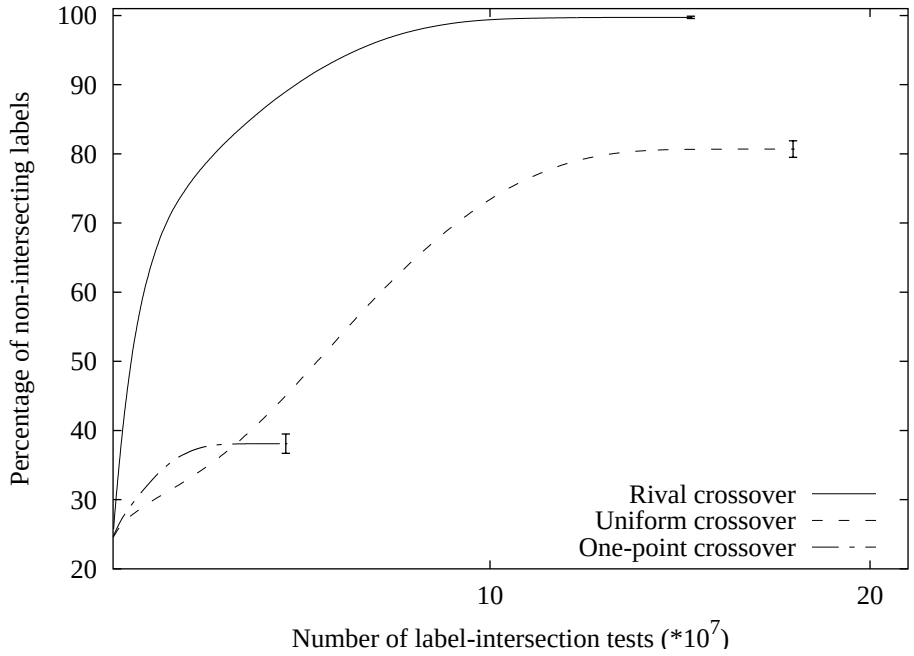

Figure 8: Comparison of crossover operators. The selection scheme is the elitist recombination scheme, $Pr_c = 1.0$, $Pr_m = 0.0$, and the population size is 200.

Strijk, 2001). They perform similar, or slightly better than simulated annealing. A comparison between the simulated-annealing algorithm and our genetic algorithm is shown in Figure 10. It shows the results are comparable. An advantage of the GA over the SA algorithm is that additional constraints—that is, cartographic rules—are more easily incorporated. More details can be found in Van Dijk's thesis. We can conclude that our GA succeeds in finding solutions of good quality. In the next section, we will turn to the question whether the computation time is reasonable—that is, whether the scale-up behavior is favorable.

4 Experimental evaluation

In the previous sections, we examined the theoretical models and derived several design rules. Using these rules, we developed a GA that is able to find solutions of good quality. According to the models, the number of fitness evaluations should scale linearly if the GA adheres to the assumptions of the models. The question is whether these assumptions are realistic. Our GA does not match them completely, and it may deviate too much from them for the predictions to hold. First, we will check each assumption of the models and discuss how much the GA adheres to it. Then we will experimentally find the critical population size and number of generations until convergence, and see whether their scale-up behavior matches with the prediction of the models. This will gives us an idea about the applicability of the models in a practical setting.

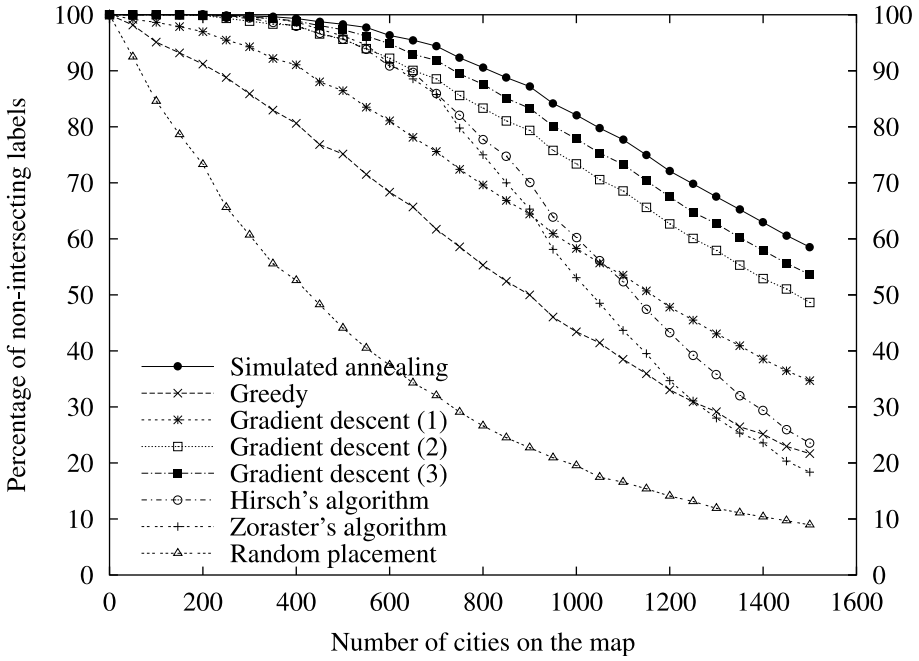

Figure 9: Results from the paper by Christensen et al.

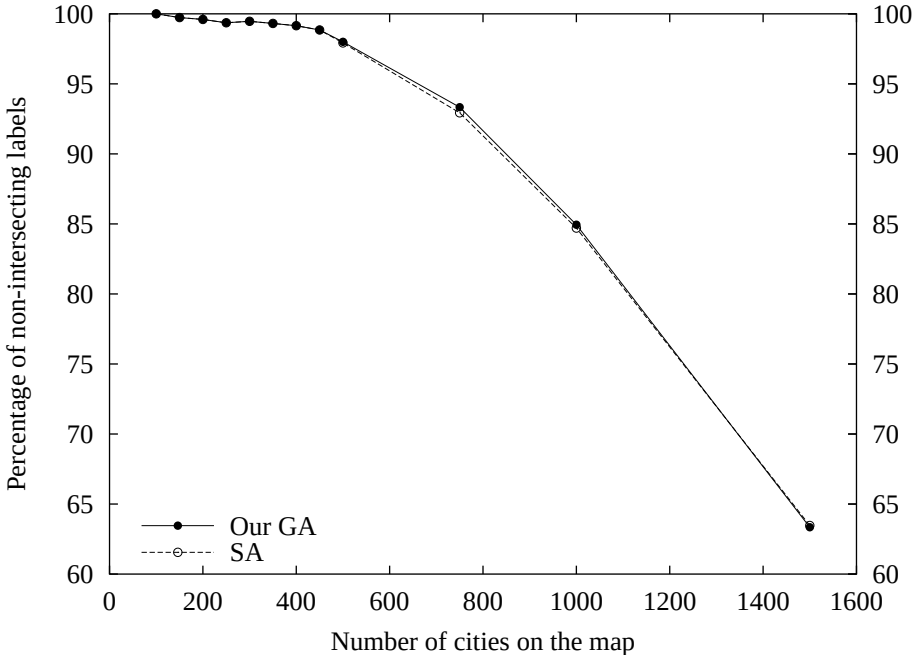

Figure 10: Comparison of SA with our GA for the four-position model. Also shown is a run of SA with the GLO.

4.1 Adherence of assumptions

We will now check all model assumptions which were stated in Section 2:

The fitness function is additively decomposable: Equation 5 shows the fitness function can be expressed as an ADF.

The order of partitions, k, is a fixed constant, with $k \ll l$: The partitions in the map labeling problem (rival groups) are not of fixed order, but the largest rival group can be taken as a conservative estimate. Moreover, the size of rival groups does not vary too much (on the dense maps used in the experiments, cities have on average about six or seven rivals, with a maximum of 14 rivals).

The fitness function is uniformly scaled: Each partial function can contribute either zero or one to the overall fitness. Therefore, the fitness function is uniformly scaled.

The fitness function is semi-separable: Each city occurs in a bounded number of rival groups, since the number of rivals is bounded. Therefore, each gene is input to a bounded number of partial functions, and the fitness function is semi-separable.

All building blocks are present in the initial population: Since building-block formation is possible, and indeed very likely to happen, this requirement can be relaxed.

The selection scheme is rank-based: This requirement is met by using either tournament selection or the elitist recombination scheme. We will experimentally test the predictions for both selection schemes.

Mixing is perfect: Rival crossover can be seen as a kind of uniform crossover on the level of the building blocks. As a result, we can be reasonably confident that mixing is performed adequately.

No disruption of building blocks takes place: In practice, some disruption takes place but is minimized due to the use of the geometrically local optimizer with the effect that it has a marginal influence on the behavior of the algorithm. Since building blocks can be disrupted, the saturation barrier is not absorbing. However, a gambler that reached the saturation barrier will with high probability stay in its proximity.

We find that some assumptions are not adhered to exactly but deviate somewhat. Still, we expect that the deviation is not serious enough to falsify the prediction. More generally speaking, we expect the models to be quite liberal in their assumptions. For example, the models assume that mixing completely removes the correlations between building blocks (or rather, their partitions) that were introduced by selection. We do not expect radically different behavior if mixing slightly less.

Therefore, we conclude that we can be reasonably confident that none of the underlying assumptions is seriously violated. We expect to see the scale-up behavior predicted by the models. The next section is devoted to experimentally putting this expectation to the test.

4.2 Empirical results

Experimental data was gathered by running the GA on randomly generated maps. The maps were square grids, embedded on a torus (to remove boundary effects). They were generated by repeatedly selecting uniformly at random a location for a point and its label on the torus, making sure the label did not overlap other labels. All labels had fixed dimensions of 30 by 7 units. Afterwards, the labels were removed and the GA was used to find a placement for the labels again. This way we were certain that it was possible to place all labels without intersecting other labels, and the optimum was always the number of cities on the map. The density δ of the map is the number of units on the grid for each point. For all subsequent experiments, we used a density of $\delta = 450$—that is, an area of 670 units squared contains 1000 points. The density is equal for all maps. Therefore, maps with more points are bigger.

In the remainder of this section, functions will be fitted to data by using the Levenberg-Marquardt algorithm for non-linear least-squares fitting, with the data points weighted by their standard deviation.

We present the experimental verifications in a number of steps. First, we investigate the influence of selection pressure. One of the most critical assumptions of the models is that mixing is perfect. If the selection pressure is too high, this assumption is clearly violated. We then look at how the gambler's-ruin model can be fitted to our experimental data. This allows us to derive the critical population size. The number of generations to converge for a GA using the critical population size can subsequently be found. We show that the functions predicted by the models can be fitted well to the experimental results. Finally, the total, minimal number of function evaluations can be derived and is shown to be linear in the input size.

Selection pressure

We start by investigating the selection pressure. An efficient GA uses a selection scheme that pushes as hard as possible while still allowing mixing to occur adequately. We performed experiments to find the optimal tournament size for the tournament selection scheme. The GA used a population size of 200, which, as will become apparent from the other experiments, is large enough. The termination criterion used for all runs was convergence of fitness, that is, the average fitness becomes equal to the fitness of the best individual. The GA was run five times with different seeds for the random-number generator on five different maps of 1000 points. The average of those twenty-five runs was used as a data point.

The results are shown in Figure 11. As can be seen in the figure, the optimal tournament size seems to be two or three, since with larger tournaments the quality begins to drop too much. We used a tournament size of two in all following experiments.

Use of the gambler's-ruin model

Since we have argued that the assumptions are not significantly violated, we should be able to apply the gambler's-ruin model to describe the behavior of the GA for map labeling. Equation 2 gives us the probability $Pr(n)$ a certain gambler hits the saturation barrier. We have $m = n_{pts}$ gamblers running in parallel in the GA (where n_{pts} is the number of features). Each partition corresponds with a rival group. The optimal schema of the partition will place the label of the central point of the rival group in a free position. Given a population size n, the fitness $f_{fit}(\mathbf{x}^*(n))$ of the final solution

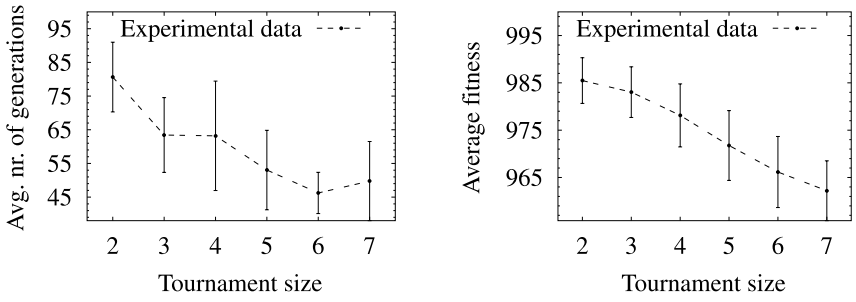

Figure 11: Influence of different selection pressures for tournament selection. On the left, the average number of generations is shown. On the right, the average fitness is shown.

$\mathbf{x}^*(n)$ is equal to the number of partitions that converge to the optimal schema (given by Equation 3: $E[n_{conv}] = m \cdot Pr(n)$). Therefore, the following holds for the expected fitness of the final solution when a population size of n is used:

$$E[f_{fit}(\mathbf{x}^*(n))] = E[n_{conv}] = n_{pts} \cdot Pr(n)\,, \tag{6}$$

where $Pr(n)$ is as given in Equation 2, and $E[\cdot]$ denotes the expected value.

For maps of size $n_{pts} \in \{200, 500, 1000, 1500, 2000, 4000, 7000, 10000\}$ we ran the GA with population size $n \in \{10, 30, 50, 100, 110, 200\}$. The GA was run three times with different seeds for the random-number generator on three different maps of the same size. The average of those nine runs was used for further computation.

The experimental data for maps of 10000 points is shown in Figure 12 (note that the figure is scaled to make 1 the optimum). We fitted Equation 6 to this data. The closeness of the fit shows that the gambler's-ruin model gives a reasonably close approximation of the relation between population size and the quality of the final solution. All experiments were also done for the elitist recombination scheme. Experimental results for maps with 10000 points are given in Figure 13 (again, be aware that the figure is scaled to make 1 the optimum). Note that the fit shown for elitist recombination is better than the one for tournament selection.

The experimental critical population size

The critical population size for each map of a certain size is found by fitting Equation 6 to the experimental data and using the function to find the point where the fitness was 97% of the optimum. Since it is guaranteed that all labels can be placed without intersections, the optimum is n_{pts} labels placed. The critical population size can be calculated as

$$n^* = g^{-1}(0.97 \cdot n_{pts})\,,$$

where $g^{-1}(\cdot)$ denotes the inverse of the function $g(\cdot)$ resulting from fitting Equation 6 to the experimental data.

The critical population size is calculated in this way for each map size. The results are plotted in Figure 14, where a square-root function is fitted to verify the prediction of the gambler's-ruin model. This prediction, which states that the relation between

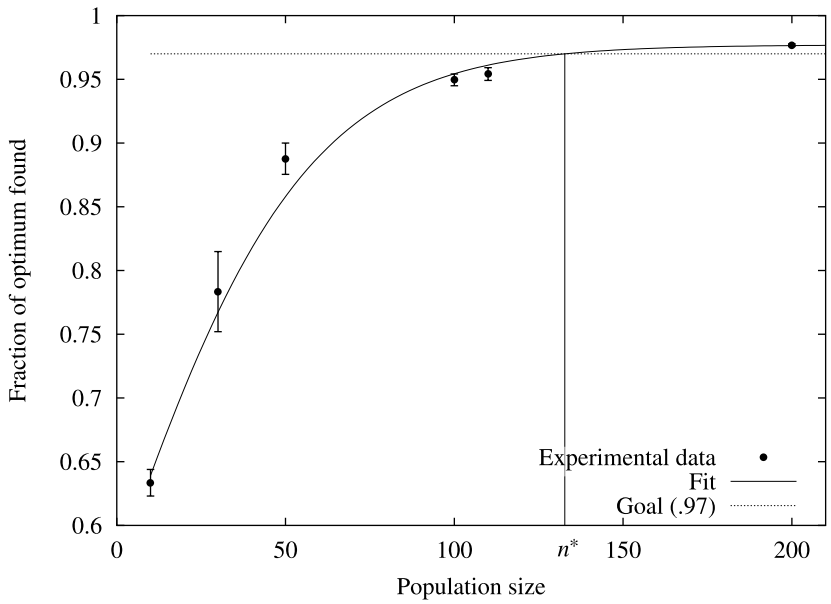

Figure 12: Fit of gambler's-ruin prediction to data for maps with 10000 cities. The GA used tournament selection, $Pr_c = 1.0$, $Pr_m = 0.0$. Note that the figure is scaled to make 1 the optimum.

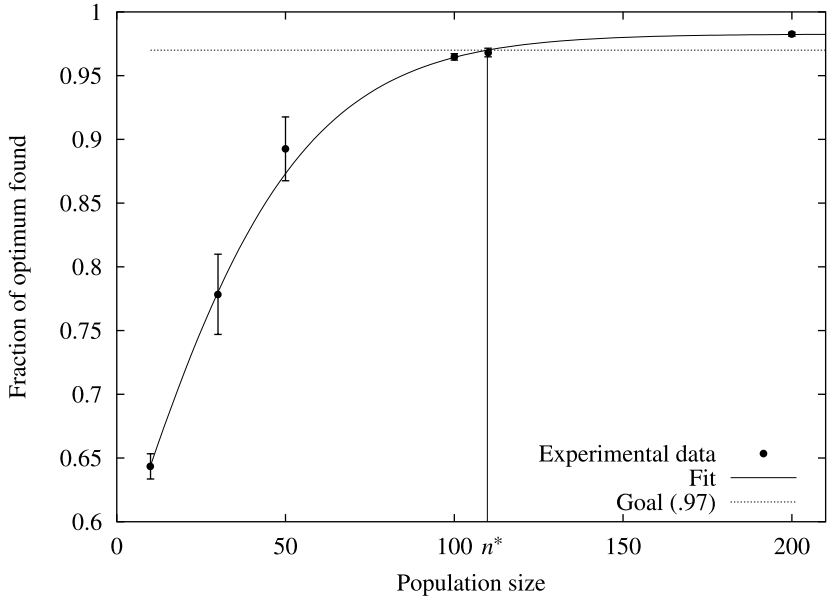

Figure 13: Fit of gambler's-ruin prediction to data for maps with 10000 cities. The GA used the elitist recombination scheme, $Pr_c = 1.0$, $Pr_m = 0.0$. Note that the figure is scaled to make 1 the optimum.

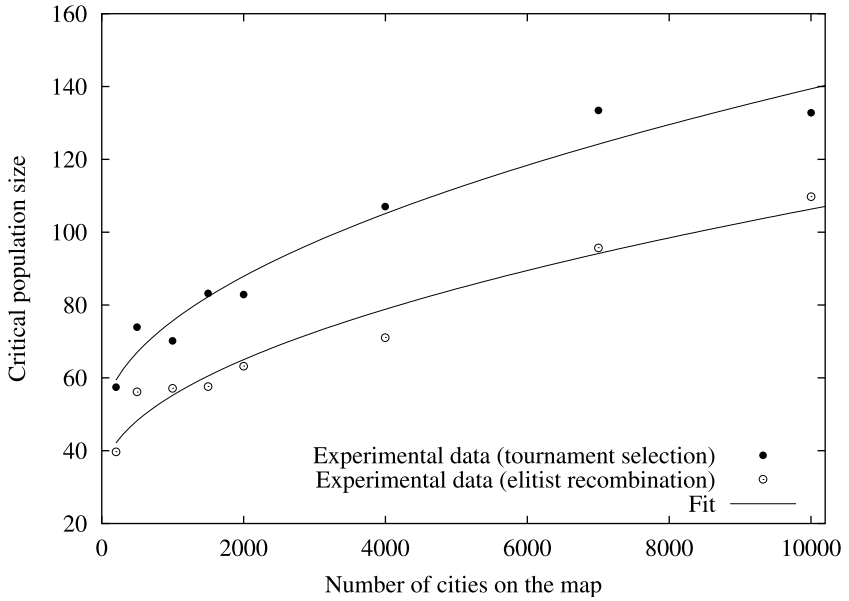

Figure 14: Critical population size for a quality of 97% of the global optimum. Shown is experimental data with the fit of the predicted function, for tournament selection and the elitist recombination scheme.

critical population size n^* and problem length l should be $n^* = O(\sqrt{l})$, is confirmed. Also it is clear that very small population sizes are sufficient.

We also tried the same experiments with elitist recombination instead of tournament selection as the selection scheme. The results are also shown in Figure 14 and show the same scale-up behavior. Note that elitist recombination succeeds in finding solutions of the same quality but can use smaller populations than tournament selection.

The number of generations

The number of generations needed to converge, when the population is equal to the critical population size, is obtained in a similar fashion. The critical population size n^* has already been calculated. For each input size l, a function is fitted to a set of data points. Each point consists of the population size and the number of generations that were spent to find a solution of the required quality. To these data points, the function $g(x) = O(1 - \frac{1}{x})$ if fitted. This function was chosen as it fits the experimental data reasonably well. After fitting the function to the data, the critical number of generations t^* is given as $t^* = g(n^*)$.[4] This is done for each map size, and the results are shown for both selection schemes in Figure 15. The number of generations for the elitist recombination scheme, an incremental replacement scheme, is calculated by dividing the number of recombinations by half of the population size. The experimental results are shown with a fit to a square root. The prediction of $t^* = O(\sqrt{l})$ is confirmed.

[4]The use of an interpolation averages out stochastic effects. This was deemed more reliable than deriving t^* from additional runs with a population size of n^*.

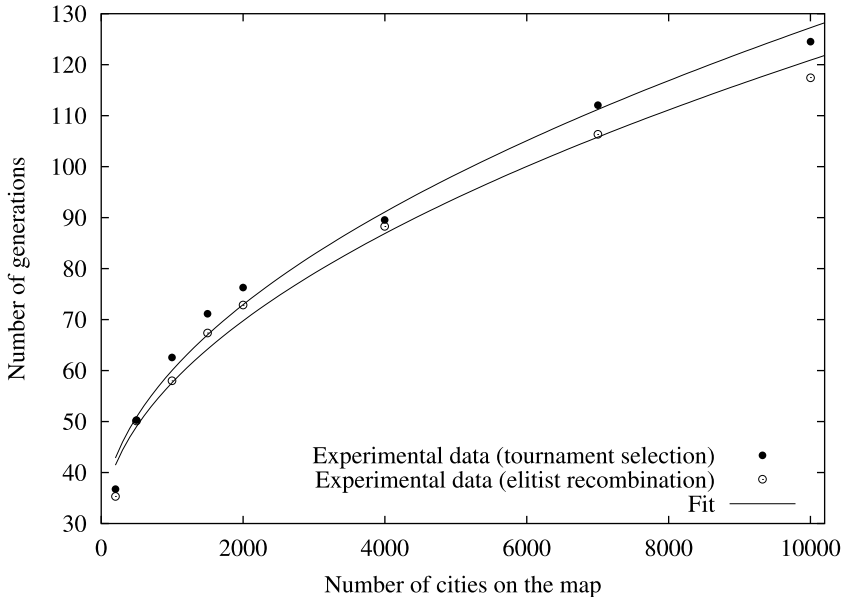

Figure 15: Run time in number of generations of GA when using the critical population size. Shown is experimental data with the fit of the predicted function, for tournament selection and the elitist recombination selection scheme.

Total amount of computational effort

Since the number of fitness evaluations is $E = t^* \cdot n^*$, it follows that $E = O(l)$ (the number of evaluations scales up linearly with the problem size). In Figure 16 the required number of evaluations for a given map size—the optimal population size times the number of generations until convergence—is plotted, and a linear function is fitted to it.

Figure 16 shows that the GA using tournament selection, compared with the GA using elitist recombination, requires more computational effort to obtain the same level of quality. Tournament selection also gives less reliable results, because the spread of the final solutions is larger than with elitist recombination.

5 Conclusion

In this paper, we examined two theoretical models from literature that allow us to make predictions of the scale-up behavior of GAs under certain assumptions. The models present a view of the way a GA works that stresses issues like good mixing, minimizing disruption and ensuring good building-block supply. With these insights in mind, we developed several design rules. These were used to design a GA for the map-labeling problem, an NP-hard cartographic problem. The GA was able to find solutions of good quality. Then we tested whether the use of the design rules (inspired by the theoretical models) for the design of the map-labeling GA resulted in scale-up behavior as predicted by the models. We experimentally found the critical population size and number of generations until convergence for different input sizes and compared these with the predictions of the models. It was found that the scale-up behavior matched the

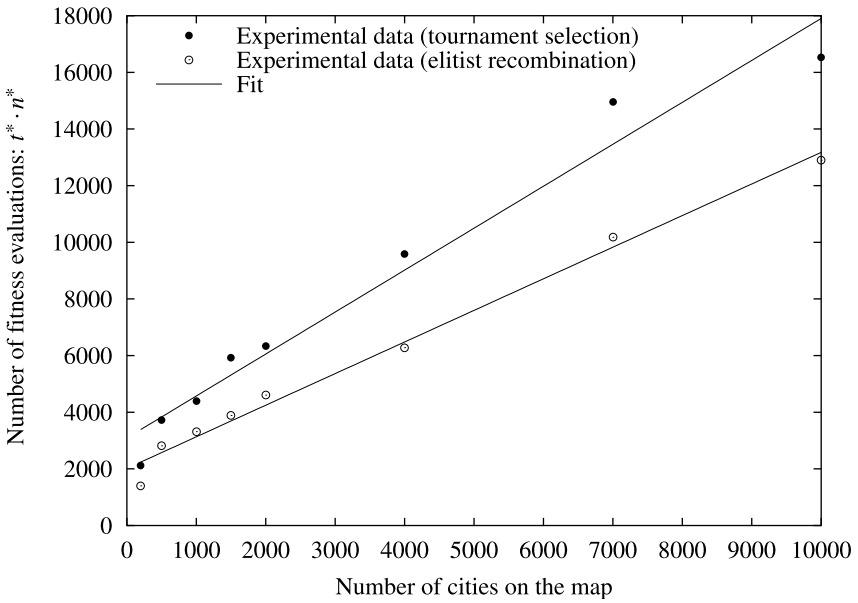

Figure 16: The scale-up behavior of the number of fitness evaluations is linear.

predictions. As a result, we have shown that the number of fitness evaluations scales linearly with respect to the input size for our map-labeling GA.

Much theoretical research has been done on problems of bounded difficulty, such as the bit-counting problem and the concatenated trap function. This is justified by the implicit claim that many real-world problems are either problems of bounded difficulty, or can be solved satisfactorily by assuming they are (such as the map-labeling problem). Therefore, one can expect that the theoretical results are useful in the design and analysis of GAs for real-world problems. We have shown that for at least one instance of an NP-hard problem (namely, the map-labeling problem), the reduction to a problem of bounded difficulty allows us to quickly find solutions that are near optimal.

For some problems, it will be fairly straight-forward to apply the design rules to obtain a competent GA. It is significant that the assumptions that underlie the models are more liberal than one may expect, since the map-labeling GA was allowed to deviate from the assumptions in some extent but still showed the expected scale-up behavior. This suggest that the design rules can be applied to a broad range of problems. A notable case of successful GA design is the grouping GA of Falkenauer (1996), which uses a special representation and operators to match the building-block structure of grouping problems. Whereas this GA was not designed using explicit design rules, we believe it could be largely derived from our rules.

But even for problems for which the application is less obvious, the design rules highlight the important issues that need to be considered in order to design a competent GA. For example, for a problem in which the linkage of the building blocks is not readily available, the natural course of action would be to try to learn the linkage. Another option would be to accept that one can not decide a priori which crossover is best and try to find a linkage-respecting crossover by trial and error. Thus the design rules guide the designer in developing a competent GA.

References

Bäck, T. (1995). Generalized convergence models for tournament- and (μ,λ)-selection. In L. J. Eshelman (editor), *Proceedings of the sixth international conference on genetic algorithms and their applications* (pages 2–8). Morgan-Kaufman.

Bulmer, M. G. (1980). *The mathematical theory of quantitative genetics.* Oxford: Clarendon Press.

Christensen, J., Marks, J., and Shieber, S. (1995). An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, *14*(3), 203–232.

Deb, K., and Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*(4).

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, *2*(1), 5–30.

Feller, W. (1966). *An introduction to probability theory and its applications (2nd edition).* Wiley.

Formann, M., and Wagner, F. (1991). A packing problem with applications to lettering of maps. In *Proceedings of the seventh annual ACM symposium on computational geometry* (pages 281–288).

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, *3*(2), 153–171.

Goldberg, D. E. (1989b). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley.

Goldberg, D. E. (1999). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In P. Bentley (editor), *Evolutionary design by computers* (pages 105–118). Academic Press.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*(4), 333–362.

Harik, G. (1997). *Learning linkage to efficiently solve problems of bounded difficulty using genetic algorithms.* PhD thesis, University of Michigan, Ann Arbor.

Harik, G., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253.

Jansen, T., and Wegener, I. (2001). Real royal road functions - where crossover provably is essential. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (editors), *Proceedings of the genetic and evolutionary computation conference* (pages 375–382). Morgan-Kaufmann.

Kargupta, H. (1995). *SEARCH, polynomial complexity, and the fast messy genetic algorithm.* PhD thesis, University of Illinois at Urbana-Champaign.

Kargupta, H. (1996). The gene expression messy genetic algorithm. In *Proceedings of the IEEE international conference on evolutionary computation.* IEEE Press.

Lobo, F. G., Goldberg, D. E., and Pelikan, M. (2000). Time complexity of genetic algorithms on exponentially scaled problems. In D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer (editors), *Proceedings of the genetic and evolutionary computation conference* (pages 151–158). Morgan-Kaufmann.

Marks, J., and Shieber, S. (1991). *The computational complexity of cartographic label placement* (Technical Report No. TR-05-91). Harvard CS.

Miller, B. L., and Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4*(2).

Mühlenbein, H., and Mahnig, T. (1999). Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, *7*, 19–32.

Mühlenbein, H., and Paass, G. (1996). From recombination of genes to the estimation of distributions: I. binary parameters. In W. Ebeling, I. Rechenberg, H. P. Schwefel, and H.-M. Voigt (editors), *Lecture notes in computer science, volume 1141: Proceedings of the parallel problem solving from nature IV conference* (pages 178–187).

Mühlenbein, H., and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: Continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Munetomo, M., and Goldberg, D. E. (1999). Identifying linkage groups by nonlinearity/non-monotonicity detection. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (editors), *Proceedings of the genetic and evolutionary computation conference* (pages 433–440). Morgan-Kaufmann.

Pelikan, M., Goldberg, D. E., and Lobo, F. G. (2002). A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, *21*(1).

Poli, R. (2000). Recursive conditional schema theorem, convergence and population sizing in genetic algorithms. In L. D. Whitley and M. D. Vose (editors), *Proceedings of the sixth foundations of genetic algorithms conference.* Morgan-Kaufman.

Raidl, G. (1998). A genetic algorithm for labeling point features. In *Proceedings of the international conference on imaging science, systems, and technology* (pages 189–196).

Rowe, J. (2001). The dynamical systems model of the simple genetic algorithm. In L. Kallel, B. Naudts, and A. Rogers (editors), *Theoretical aspects of evolutionary computing* (pages 31–58). Springer-Verlag.

Shapiro, J. L., Prügel-Bennett, A., and Rattray, L. M. (1994). A statistical mechanical formulation of the dynamics of genetic algorithms. In T. C. Fogarty (editor), *Lecture Notes in Computer Science 865.* Berlin: Springer.

Stephens, C., and Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, *7*(2), 109–124.

Strijk, T. (2001). *Geometric algorithms for cartographic label placement.* PhD thesis, Utrecht University, Department of Computer Science.

Strijk, T., Verweij, B., and Aardal, K. (2000). *Algorithms for maximum independent set applied to map labelling* (Technical Report No. UU-CS-2000-22). Department of Computer Science, Utrecht University.

Thierens, D. (1999). Scalability problems of simple genetic algorithms. *Evolutionary Computation*, *7*(4), 331–352.

Thierens, D., and Goldberg, D. E. (1994a). Convergence models of genetic algorithm selection schemes. In Y. Davidor, H. P. Schwefel, and R. Männer (editors), *Lecture notes in computer science, volume 866: Proceedings of the parallel problem solving from nature III conference* (pages 119–129). Springer-Verlag.

Thierens, D., and Goldberg, D. E. (1994b). Elitist recombination: An integrated selection recombination GA. In *Proceedings of the IEEE international conference on evolutionary computation* (pages 508–512). IEEE Press.

Thierens, D., Goldberg, D. E., and Pereira, A. G. (1998). Domino convergence, drift, and the temporal-salience structure of problems. In *Proceedings of the IEEE world congress on computational intelligence* (pages 535–540). IEEE Press.

Van Dijk, S. (2001). *Genetic algorithms for map labeling.* PhD thesis, Utrecht University.

Verner, O., Wainwright, R., and Schoenefeld, D. (1997). Placing text labels on maps and diagrams using genetic algorithms with masking. *INFORMS Journal on Computing*, *9*(3), 266-275.

Verweij, B. (2000). *Selected applications of integer programming: A computational case study.* PhD thesis, Utrecht University, Department of Computer Science.

Vose, M. D. (1999). *The simple genetic algorithm: Foundations and theory.* MIT Press.

Yamamoto, M., Lorena, L. A. N., and Camâara, G. (1999). Tabu search application for point features cartographic label placement problems. In *Proceedings of the third metaheuristics international conference.*