On the Design and Analysis of Competent Selecto-recombinative GAs

Steven van Dijk steven@cs.uu.nl
Dirk Thierens dirk@cs.uu.nl
Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Mark de Berg m.t.d.berg@tue.nl
Faculty of Mathematics and Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Keywords
Map labeling, scalability, competent selecto-recombinative GAs, design rules, population sizing, convergence analysis.

Abstract
In this paper, we study two recent theoretical models—a population-sizing model and a convergence model—and examine their assumptions to gain insights into the conditions under which selecto-recombinative GAs work well. We use these insights to formulate several design rules to develop competent GAs for practical problems. To test the usefulness of the design rules, we consider as a case study the map-labeling problem, an NP-hard problem from cartography. We compare the predictions of the theoretical models with the actual performance of the GA for the map-labeling problem. Experiments show that the predictions match the observed scale-up behavior of the GA, thereby strengthening our claim that the design rules can guide the design of competent selecto-recombinative GAs for realistic problems.

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 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, 2002), 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 et al., 1994), exact schema theorems (Poli, 2000), and coarse-grained analysis of building-block evolution (Stephens and Waelbroeck, 1999).

These approaches offer valuable insights into various important aspects of genetic algorithms, but usually do not translate easily to practical design rules. Indeed, practitioners dealing with real-life problems may feel that current theoretical results are too far removed from 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 (2002). Specifically, we examine selecto-recombinative GAs, that are based on selection and recombination, on problems of bounded difficulty. It is assumed that the Building Block Hypothesis (Goldberg, 1989c) holds: good solutions can be found by finding (with selection) and combining (with recombination) building blocks. Note that we hereby exclude, for example, GAs that place a strong emphasis on mutation and use crossover as macro-mutation.

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 of the optimum. "Reasonable" means that the algorithm should *scale up* well (Thierens and Goldberg, 1993; Thierens, 1999; Sastry and Goldberg, 2003). 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 smallest 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^*) together determine the scale-up behavior of the number of fitness evaluations spent by the GA. Our goal therefore is to construct GAs that find solutions of a specific quality (for example, within 97% of the optimum) while maintaining a reasonable (for example, linear) scale-up in the number of fitness evaluations.

In this paper, we discuss two models from the literature—a population-sizing model (Goldberg et al., 1992; Harik et al., 1999) and a convergence model (Mühlenbein and Schlierkamp-Voosen, 1993; Thierens and Goldberg, 1994a; Bäck, 1995; Miller and Goldberg, 1996)—that together give a 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 ($E = O(l)$). 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.

Map labeling. The map-labeling problem comes from automated cartography and is defined as follows. Given 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. The task is to find a *labeling* (a position for each label) such that the number of non-intersecting labels is maximized. A solution for a randomly-generated map of 1000 points is shown in Figure 1. This variant of the map-labeling problem 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 (such as rivers and areas). To ease the analysis done in this paper, we will only be concerned with randomly-generated maps of uniform density such as shown in Figure 1. A more thor-

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

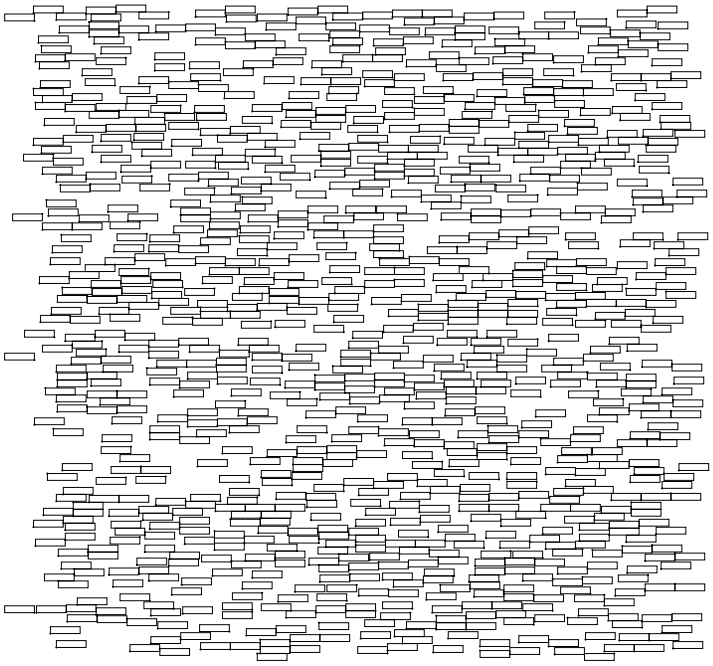


Figure 1: An example of a labeled map.

ough treatment of the map-labeling problem and the application of the GA from this paper to real cartographic maps is reported elsewhere (Van Dijk, 2001; Van Dijk et al., 2002). Note that for this problem, 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 $e_{fit} = O(l)$. Combining this with the predicted linear scale-up for the number of evaluations gives a quadratic scale-up for the amount of computational effort: $W = O(l^2)$.

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 into the conditions under which a GA performs well. Secondly, we show how these insights translate to several practical design rules. Using the rules, we design a GA that gives high-quality solutions for the map-labeling problem. We use the GA to experimentally confirm the predictions of the models, thus supporting the practical usability of the approach.

This article is structured as follows. In Section 2, we will briefly describe the two models from the literature. Next, in Section 3, we formulate the design rules and use them to design an efficient GA for the map-labeling problem. The impact of disregarding the rules is discussed as well. The map-labeling GA is not able to satisfy the underlying assumptions of the models as well as the GAs for the artificial problems on which the models were originally tested. However, since the design rules aim to satisfy the underlying assumptions of the models, we expect no serious deviations from the assumptions and thus expect the predictions to hold. Section 4 is devoted to the 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 convergence 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 given 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). 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[2] 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 at most two partial functions. If the number of sub-functions a gene can be input to is small enough, a semi-separable function will behave like a separable function. Note that the critical number of sub-functions depends on whether the optima of sub-functions that share genes agree on the values of the shared genes. The ADF is uniformly scaled if the functions $f_i(\cdot)$ have similar distributions of values.

Problems that are defined by an additively decomposable function are instances for which the Building Block Hypothesis (Goldberg, 1989c) holds: they can be solved by combining smaller parts called *building blocks* of bounded size. We define a *building block* as a schema with the highest fitness in the partition corresponding to a partial fitness function.[3] Combining building blocks simply means that the final solution matches each building block.

We will consider problems of bounded difficulty for which the fitness function can be expressed as follows:

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

where partial functions f_i are defined on at most k genes, with $k \ll l$. In addition, the functions $f_i(\cdot)$ are (semi-)separable and have similar distributions of values.

The bit-counting problem ($f_{fit}(\mathbf{x}) = \sum_{i=1}^{l} x_i$, with $x_i \in \{0, 1\}$) is the most simple instance in this class of functions. A characteristic example of a problem that involves linkage 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.

[2] We will assume a simple encoding that uses a gene for each problem variable, and will use those terms interchangeably.

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

We define a *trap function* as follows:

$$f_{trap}(\mathbf{x}_{i+1\ldots i+k}) = \begin{cases} k & \text{if } u(\mathbf{x}_{i+1\ldots i+k}) = k \\ k-1-u(\mathbf{x}_{i+1\ldots i+k}) & \text{otherwise} \end{cases}, \tag{2}$$

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 $k \geq 4$ have also been called *fully deceptive functions* (Goldberg, 1989b; Deb and Goldberg, 1993), because information from all lower-order partitions (defined over less than k genes) directs the search away from the optimal schema. In order to grow the proportion of individuals in the population that match the building block (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. Linkage can be formally defined in terms of non-zero Walsh coefficients (Goldberg, 1989a), but the intuitive notion of a non-linear interaction between linked genes will suffice for this paper.

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

$$f_{fit}(\mathbf{x}) = \sum_{i=0}^{m-1} f_{trap}(\mathbf{x}_{i\cdot k+1\ldots i\cdot k+k}) \,. \tag{3}$$

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, 1989c). 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 then 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) of 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, adequate mixing can be obtained. Mixing is the recombination of schemata from the parents to form new combinations in the children. Mixing is called *perfect* when no correlations between partitions[4]—which were introduced by selection—remain after crossover. Uniform crossover is not a perfect mixer, but when the selection pressure is moderate, it is

[4]Unless otherwise specified, "partitions" refers to the partitions that correspond with the sub-functions from the fitness function.

safe to assume it is fast enough to avoid premature convergence (Thierens and Goldberg, 1993). Note that the recombination operators used in evolutionary algorithms that are based on the estimation of distributions (Mühlenbein and Paass, 1996; Bosman and Thierens, 2002; Pelikan et al., 2002) can be perfect mixers when the learned model matches the linkage.

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 ($Pr_c = 1.0$). The rate of convergence of the GA is primarily determined by the selection pressure of the selection scheme. Under the assumption of perfect mixing, the population fitness is binomially distributed, which can be approximated well with a normal distribution. Since crossover does not change the proportion of 1's—that is, the building blocks—in the population, the use of a selection scheme with constant selection pressure gives predictable convergence behavior.

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

$$t^* = \frac{(\frac{1}{2}\pi - c)\sqrt{l}}{I} = O(\sqrt{l})\,. \tag{4}$$

where l is the length of the chromosomes, I signifies the selection intensity of the selection scheme, and c is a constant depending on the proportion 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). Bäck (1995) considered (μ, λ)-selection and tournament selection, and used order statistics to generalize the results for different selection intensities. In these studies, 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 selection schemes which are rank-based.

Miller and Goldberg (1996) extended this research by considering noisy fitness functions. Furthermore, they also considered more complex problem domains than the bit-counting problem, but where the fitness function still was uniformly scaled, separable, and additively decomposable. They derived exact equations for domains where 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. Their prediction of the convergence behavior for the concatenated trap function closely matched experimental results. In addition, they showed that adding small levels of noise to the fitness function added a constant to the number of generations until convergence.

The requirement of separability can be relaxed to semi-separability by modeling the interactions between different partitions as noise. Therefore, as long as the linkage between genes from different partitions is weak, the convergence model gives a good approximation. For the case where the fitness function is exponentially scaled (instead of uniformly), similar studies (Thierens et al., 1998; Lobo et al., 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 a sufficient number of 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 reliably 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 building-block proportions had grown in the first generation, they obtained a population-sizing equation. One drawback of this approach was that it did not model the way a GA can recover from decision errors (explained later). Harik et al. (1999) addressed this issue by extending the model with the so-called gambler's-ruin model.

The decision-making model views the search process as the propagation of building blocks through the population, assuming mixing is adequate. Recall that a building block is the schema in a certain partition with the highest fitness. It is assumed that the order of the partition is bounded by a constant k. During selection (for example, a tournament of two individuals) the building block has to compete against another schema from the same partition. 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. Under the assumption of an additively decomposable fitness function, the distribution of fitnesses in the population can be approximated by a normal distribution according to the Central Limit Theorem. The probability of making the right decision Pr_{ok} (Goldberg et al., 1992) can then be formulated as a function of the cumulative distribution function of the standard normal distribution and several constants that are dependent on the problem.

The expression for Pr_{ok} is used in the gambler's-ruin model as the probability of increasing the frequency of building blocks in a certain partition. The search of a GA in a single partition is then viewed as a series of competitions that progresses until either all individuals in the population match the building block, or none does. 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; this is called the *depletion barrier*) and the existence of the building block in all individuals (n building blocks in the population; this is called 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.

The initial number of building blocks is given by x_0, which can be easily approximated if the initial population is randomly generated. 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. (1999) 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 truncation selection.

The formulation as a random walk allows for the calculation of the probability $Pr(n)$ of the gambler eventually hitting the saturation barrier using a population of

size n (Feller, 1966):

$$Pr(n) = \frac{1 - (\frac{1-Pr_{ok}}{Pr_{ok}})^{x_0}}{1 - (\frac{1-Pr_{ok}}{Pr_{ok}})^{n}} \,. \tag{5}$$

Given a measure of quality α that denotes the desired fraction of partitions that converge to the building block, the smallest population size to obtain a solution of the desired quality can be derived. Note that each partition is assumed to converge independently from any other partition, therefore $\alpha = Pr(n)$. Extracting n from Equation 5 and approximating Pr_{ok}, the following approximation of a population-sizing equation was found by Harik et al. (1999):

$$n^* \approx -\ln(1-\alpha)\beta\sqrt{(m-1)} = O(\sqrt{l}) \,, \tag{6}$$

where $m = l/k$ denotes the number of partitions and β is a constant that depends on the properties of the problem.

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 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 suggest that the assumption of separability of the fitness function can be relaxed to semi-separability.

2.3 Model assumptions

The convergence model and the population-sizing model share a set of underlying assumptions. Under these assumptions, the convergence model predicts that the number of generations until a properly-sized population is converged scales as $t^* = O(\sqrt{l})$. In addition, the minimal population required to reliably find a solution of a certain quality scales as $n^* = O(\sqrt{l})$. These assumptions are as follows:

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

The assumptions may appear to be quite strict. Is it possible to design a competent GA that adheres to these assumptions closely enough to find solutions for a realistic problem? Recall that we consider a GA competent when it finds solutions with a specified lower bound on quality and good scale-up behavior. To answer the question, we first turn the assumptions into practical design rules a GA practitioner can follow. We subsequently apply these rules to design a GA for the map-labeling problem and empirically compare its performance with the predictions of the models.

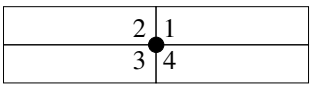


Figure 2: The four possible positions where a label can be placed.

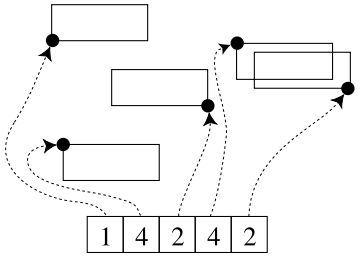


Figure 3: The encoding for a map.

3 Design rules and application to map labeling

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 we can turn into design rules. The rules we will formulate in this section are reminiscent of the "six conditions for GA success" developed by Goldberg et al. (1992), but ours are more explicit. Note that the "six conditions", although conjectured, formed the basis of much of the research described in Section 2 (Goldberg, 2002). We will use the rules 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 2) 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}.

3.1 The design rules

We will now state the design rules that we can derive from the models. Each rule is immediately followed by a description of its application in the design of the map-labeling GA:

1. **Use a fitness function that is additively decomposable, uniformly scaled and (semi-)separable.** The models suggest that a problem is tractable for a GA when the linkage partitions the representation into groups of strongly-linked genes that do not overlap much. Within each partition the schema with the highest fitness (the building block) can be found by selection and the final solution then can be found by combining all building blocks. Given a representation that holds a gene for each problem variable, a fitness function that is additively decomposable, uniformly scaled and (semi-)separable induces a linkage that makes the problem tractable for a GA.

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, as shown in Figure 3. The fitness function of our map-labeling GA satisfies the required form by just counting 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 3 is 3. The fitness function can be expressed as an additively

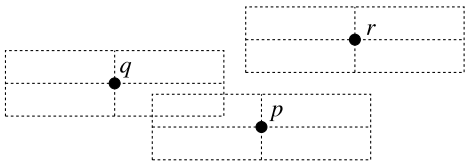


Figure 4: The *rival* relationship: cities p and q are rivals, but p and r are not. Together p and q form a rival group.

decomposable function:

$$f_{fit}(\mathbf{x}) = \sum_{i=1}^{n_{pts}} free(\mathbf{x}_i)\,, \tag{7}$$

where n_{pts} denotes the number of points on the map. We define two points as *rivals* if their labels can intersect (see Figure 4). A *rival group* consists of a certain point and its rivals. Now $\mathbf{x}_i$ denotes the genes corresponding to the rival group of point i. 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. If the rival groups are bounded in size, then each gene will be input to a bounded number of sub-functions, which makes the fitness function semi-separable.

2. **Identify building blocks.** The fact that a problem is tractable only means that a GA is, in principle, capable of solving it. Of course, the GA needs to be carefully designed in order to succeed in this task. The next critical step therefore is obtaining a good assessment of the linkage of the problem. To construct a competent GA, it is necessary to know where to search for the building blocks of the solution.

Map labeling is interesting in that the linkage of the problem is reasonably clear since it can be inferred from the geometry. Given the representation described earlier, 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 its rivals. That is, we assume the best labelings for rival groups are building blocks. Note that we make an assessment of the linkage which may disregard non-linearities over many genes. In essence, we trade solution quality for performance. It depends on the specific problem whether this is acceptable, but many NP-hard problems require such a reduction to remain tractable. For map labeling, we hypothesize that close-to-optimal solutions can still be found when the problem is treated as having bounded difficulty by only considering the non-linearities between genes that correspond with the rival-group relationship.

Note that for other problems an acceptable assessment of the linkage is not so easily found. In that case, the natural course of action would be to try and learn the linkage (Kargupta, 1996; Munetomo and Goldberg, 1999; Harik, 1999; Pelikan et al., 2002). Alternatively, one can resort to traditional trial-and-error of design choices (which have implicit assumptions about the linkage) and hope to find a competent GA by chance.

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, we chose the elitist recombination scheme (Thierens and Goldberg, 1994b). In this rank-based scheme, two parents are chosen randomly from the pop-

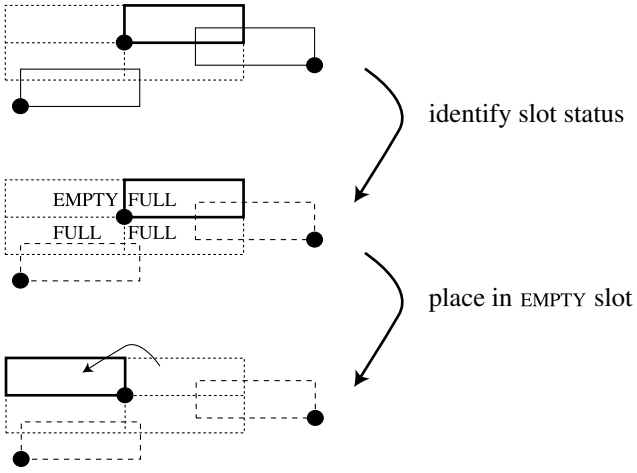


Figure 5: The geometrically local optimizer resolves conflicts after crossover.

ulation. 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. Ideally, after crossover no correlations between elements of different partitions exist. Adequate mixing can, for example, be achieved by using what Miller and Goldberg (1996) called a "uniform building-block crossover"—a crossover that chooses at random for each partition the parent from which values are copied.

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 transferred 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. Consequently, if the local labeling of the chosen point and its rivals is indeed a building block, it is transferred 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 map.

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

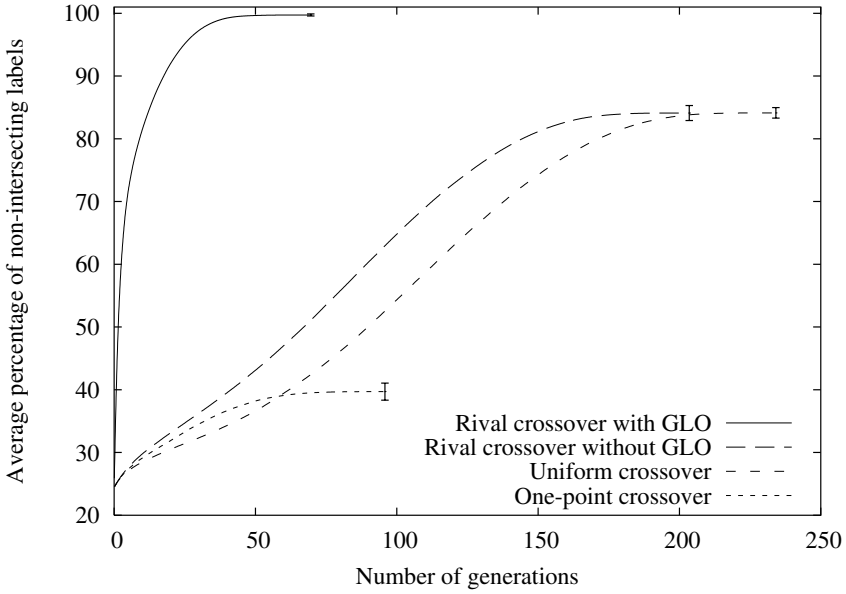


Figure 6: Comparison of crossover operators. $Pr_c = 1.0$, no mutation.

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 5). 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.

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. Therefore, no blind mutation is used ($Pr_m = 0.0$). Our use of the GLO allows us to place 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. 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.

3.2 Evaluation of the GA

To evaluate whether the design rules helped us in the construction of a competent GA, we'll first compare against other map-labeling algorithms. Next, we'll investigate the impact of deliberately violating the rules.

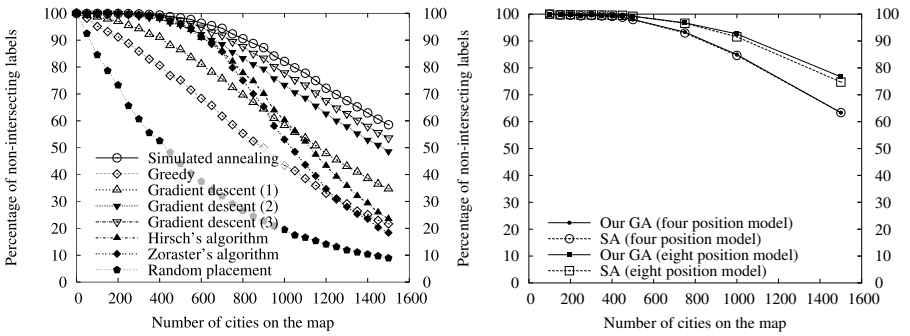


Figure 7: Left: results of the comparison of labeling algorithms by Christensen et al. Right: Comparison of simulated annealing with our GA for the four- and eight-position model.

Comparison with a simple GA. In Figure 6, we compare our GA using the so-called "rival" crossover against simple GAs using uniform and one-point crossover. To make easy comparison of results possible, all experiments in this paper are presented in terms of generations. For the runs done with the elitist recombination scheme (implemented as an incremental scheme), one generation is taken to be $\frac{1}{2}n$ recombinations. The graphs show the average of five runs on five different maps each (25 runs in total). 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 GAs used a population size of 200 and the elitist recombination scheme. A run is terminated when the average fitness equals the fitness of the best individual. Runs that include mutation are also terminated if they have performed $400 \cdot 10^6$ label-intersection tests.

The results shown in Figure 6 demonstrate that 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 et al. (1995). Their conclusion was that the best results were obtained by a simulated-annealing algorithm. Figure 7 shows their 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 were created by placing an increasing number of points in an area of fixed size. Therefore, the maps get increasingly more dense until it becomes impossible to label all points. In addition, the optimal number of free labels is not known, unlike with our maps. Since the comparison by Christensen et al., several other algorithms have been proposed, such as genetic algorithms (Verner et al., 1997; Raidl, 1998), heuristics for maximum independent set (Strijk et al., 2000; Verweij, 2000; Strijk, 2001), and tabu-search (Yamamoto et al., 2002). They perform similar, or slightly better than simulated annealing. A comparison between our implementation of the simulated-annealing algorithm and our genetic algorithm is shown in Figure 7. 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 through the GLO (Van Dijk, 2001). We can conclude that

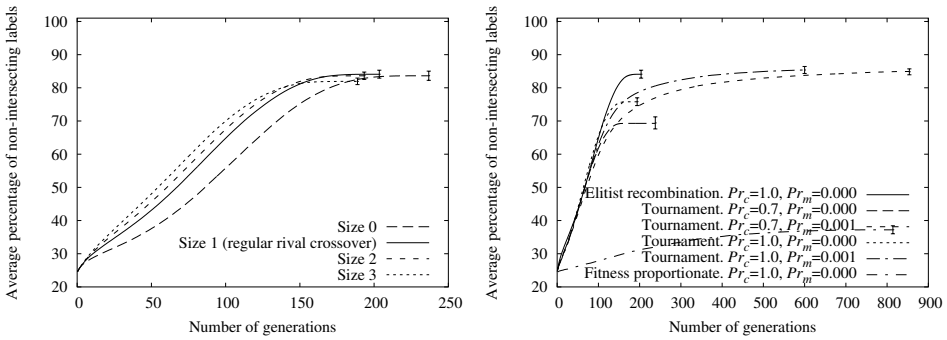


Figure 8: Left: varying the rival-group size. The GAs use rival crossover without GLO, the ER scheme, $Pr_c = 1.0$, and no mutation. Right: different selection schemes. The GAs use rival crossover without GLO.

our GA succeeds in finding solutions of good quality. In Section 4, we will turn to the question of whether the computation time is reasonable—that is, whether the scale-up behavior is favorable. Specifically, we will be interested in whether the scale-up matches the prediction of the models.

3.3 Violating the design rules

It is instructive to study what happens if the design rules are *not* followed. We will take each of the rules and discuss the impact of ignoring it. Experiments were all performed with a population size of 200.

1. The first design rule prescribes the fitness function to be in a certain form. The fitness function and the representation together define the true linkage between genes. Assuming a representation that uses a gene for each problem variable, a fitness function of this form induces a linkage that allows separate partitions to be searched independent from each other and thus makes the problem tractable for a selecto-recombinative GA. This seems to suggest, for example, that adding penalty functions to the fitness function that are not additively decomposable is a technique that may adversely affect the performance of a GA. Consequently, when the map-labeling GA was extended to handle real-world maps with cartographic constraints (Van Dijk, 2001), those constraints were not enforced by using penalty functions but instead by using local rules in the GLO.

2. Identify building blocks: Failing to find a good assessment of the linkage can have a dramatic impact on the performance of the GA. A good assessment of the linkage is needed to design the crossover operator and minimise disruption. The standard crossover operators (one-point, two-point, and uniform crossover) can be viewed as assessments of linkage that attempt to cover broad classes of problems.

   The design of the rival crossover for the map-labeling problem was guided by the assessment of the linkage. The comparison with standard crossovers (with their implicit assumptions about the linkage), shown in Figure 6, demonstrates that it performs better even without the GLO. The assessment of linkage was derived from the rival relationship, which states that two labels are rivals if they can intersect. We can generalize this to include the notion of size, which simply uses the

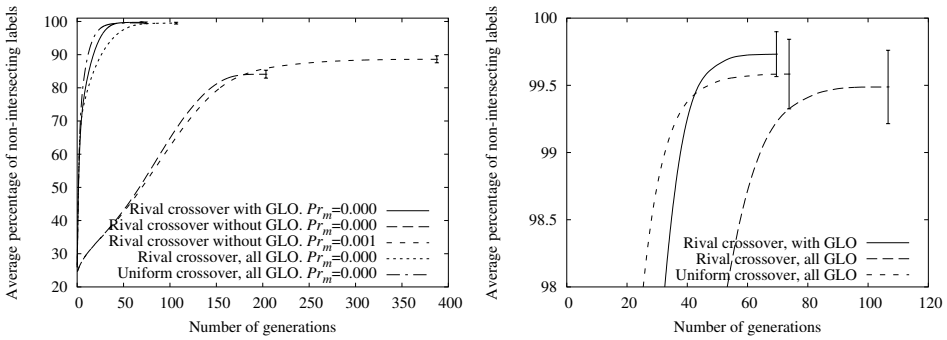


Figure 9: Use of the geometrically local optimizer. The GAs use the ER scheme, $Pr_c = 1.0$. Right: close-up of left graph.

rival relation in a transitive sense. For example, a rival group of zero size only contains the center label. Using size one gives the usual rival relation, and the rival group of size two contains the labels of the size-one group and all their rivals. Different sizes perform differently, depending on how well the assessment matches the true linkage. Figure 8 (left) shows a comparison of different sizes, demonstrating that size one was the appropriate choice.

3. Use a rank-based selection scheme: A selection scheme with constant selection pressure is necessary to maintain a steady growth of the proportion of building blocks. Figure 8 (right) shows runs done with different selection schemes, employing rival crossover without the GLO. For the sake of completeness, the effect of varying the parameters Pr_c and Pr_m is also shown for tournament selection. The results clearly show that fitness-proportionate selection without a scaling method is inferior to tournament selection and the ER scheme. Experiments done with rival crossover that used the GLO gave similar results.

4. Ensure good mixing of building blocks: Failure to mix rapidly enough can cause premature convergence to an inferior solution, as demonstrated by the comparison of crossovers in Figure 6. Note that the GA using one-point crossover converges before it has had time to mix the building blocks.

5. Minimise disruption: Disruption during crossover will degrade the proportion of building blocks in the population. The GLO successfully combats disruption, allowing the GA to find a near-optimal solution (see Figure 9 (left)). Recall that we only apply the GLO to points where disruption could have arisen. The combined use of all applications of the GLO can be seen as a local optimizer in a hybrid GA. Interestingly, applying the GLO to all genes after uniform crossover gives good solutions too (see Figure 9 (right)). This GA is very similar to the one proposed by Raidl (1998). Note that applying the GLO to all genes degrades the performance of rival crossover[5]. This is to be expected, since rival crossover is not disruptive enough to counter the strong bias towards local optima that arises by using GLO

[5] A Mann-Whitney test at significance level 0.05 confirms that the quality of the solutions found by the proposed GA with rival crossover was significantly better than that of the GA with uniform crossover and the GLO applied everywhere. The quality of both was significantly better than that of the GA with rival crossover and GLO applied everywhere.

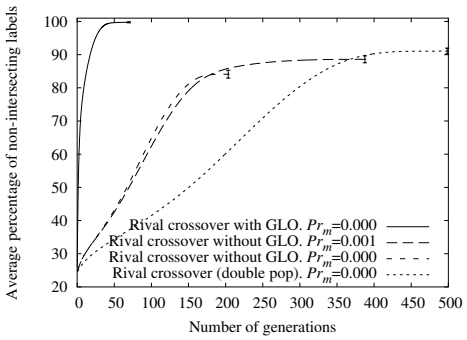


Figure 10: Different sources of building blocks. The GAs use the ER scheme and $Pr_c = 1.0$.

on every gene. Selecto-recombinative GAs construct good solutions by mixing and recombining building blocks. On the other hand, other GAs may also find solutions of the same quality by employing a different search process.

6. Ensure good building-block supply: Good solutions cannot be assembled without the required building blocks. Building blocks can be provided in the initial population, or by genetic operators during the run. Figure 10 shows that performance improves with mutation, and also when the population size is doubled. In the proposed GA, the GLO acts as a major source of building blocks, so no (blind) mutation is used and populations can be kept small.

4 Experimental evaluation

In the previous sections, we examined the theoretical models and turned their underlying assumptions into practical design rules. Using these rules, we developed a GA that is able to find solutions of good quality. To qualify as a "competent" GA, it should also have good scale-up behavior. 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 attainable for a real-world GA. 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.

4.1 Adherence of model assumptions for map-labeling GA

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

The fitness function is additively decomposable: Equation 7 shows that 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, the number of rivals is distributed approximately normal with mean 6.6 and standard deviation 1.6.

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: We will experimentally test the predictions for two rank-based selection schemes: tournament selection and the elitist recombination scheme.

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 limited influence on the behavior of the algorithm. Since building blocks can be disrupted, the saturation barrier in the gambler's-ruin model 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 unless mixing is substantially slower.

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 use of randomly-generated maps allows us to systematically vary the input size of the algorithm and know beforehand an optimal solution. Results of the GA on real cartographic maps are described elsewhere (Van Dijk, 2001). The randomly-generated 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 placing its label where it would not intersect another label. If the label could not be placed, the point was discarded. 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. Experiments were performed for both tournament selection (with tournament size of two) and the elitist recombination scheme. Both selection schemes have the same selection intensity (Thierens, 1997).

We present the experimental results in a number of steps. We start by looking 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.

4.2.1 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 5 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 points). Each partition corresponds to 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 $\mathbf{x}^*(n)$ is equal to the number of partitions that converge to the optimal schema. Therefore, the following holds for the expected fitness of the final solution when a population size of n is used:

$$E[f_{fit}(\mathbf{x}^*(n))] = n_{pts} \cdot Pr(n)\,, \tag{8}$$

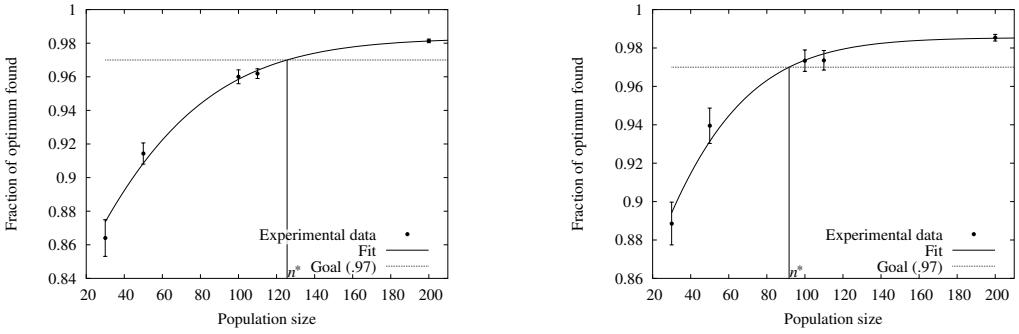
where $Pr(n)$ is as given in Equation 5, and $E[\cdot]$ denotes the expected value.

For maps of size $n_{pts} \in \{500, 1000, 1500, 2000, 4000, 7000, 10000\}$ we ran the GA with population size $n \in \{30, 50, 100, 110, 200\}$. For map sizes smaller or equal to 2000, the GA was run eight times with different seeds for the random-number generator on eight different maps of the same size. For larger maps, due to computational constraints, the GA was run four times on four different maps. For each map, the eight (or four) runs of the GA were averaged. For each population size, this results in a single data point.

The experimental data for a map of 4000 points with GAs that use tournament selection is shown in Figure 11(a) (note that the figure is scaled to make 1 the optimum). We fitted Equation 8 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 and the experimental results for a map with 4000 points are shown in Figure 11(b).

4.2.2 The experimental critical population size

The critical population size for each map of a certain size is found by fitting Equation 8 to the experimental data and using the function to find the point where the fitness was 97% of the optimum. Since maps are constructed in a way that all labels can be placed without intersections, the optimum is n_{pts} labels placed. The critical population size



(a) Tournament selection

(b) Elitist recombination

Figure 11: Fit of gambler's-ruin prediction to data for maps with 4000 cities. Note that the figure is scaled to make 1 the optimum.

can be calculated as

$$n^* = g^{-1}(0.97 \cdot n_{pts}), \tag{9}$$

where $g^{-1}(\cdot)$ denotes the inverse of the function $g(\cdot)$ resulting from fitting Equation 8 to the experimental data.

The critical population size is calculated in this way for the eight (or four) maps of each map size. Therefore, for each map size, we obtain a data point that gives us the average critical population size for maps of that size. The results are plotted in Figure 12, 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 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 presented in Figure 12 as well 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.

4.2.3 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 $g(x) = \Theta(1 - \frac{1}{x})$ is fitted. This function was chosen as it fits the experimental data reasonably well and it allowed us to obtain a good interpolation. After fitting the function to the data, the critical number of generations t^* is given as $t^* = g(n^*)$.[6] This is done for each map size, and the results are shown for both selection schemes in

[6]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^*.

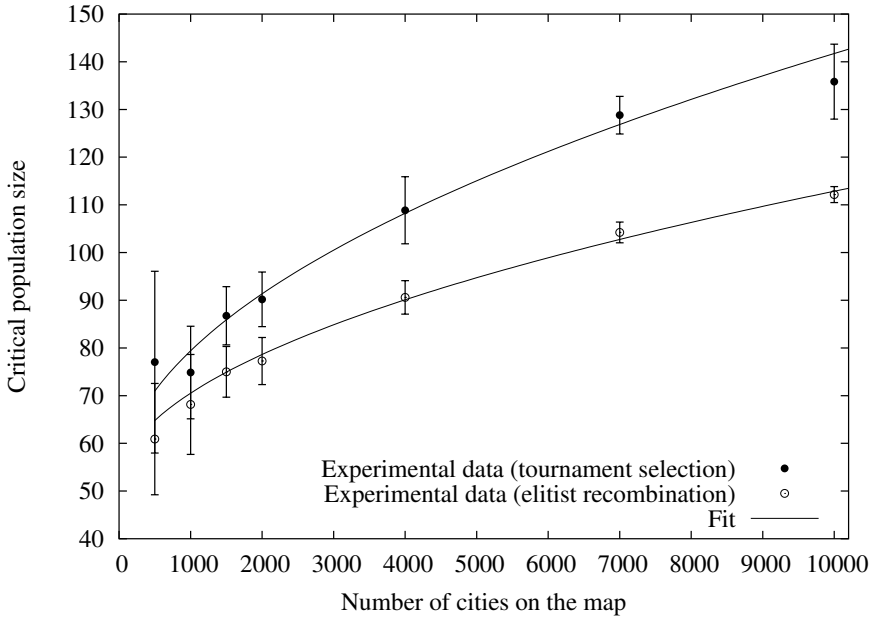


Figure 12: Critical population size for a quality of 97% of the global optimum. Shown is experimental data with the fit of the predicted function, for tournament selection and the elitist recombination scheme.

Figure 13. The experimental results are shown with a fit to a square root. The prediction of $t^* = O(\sqrt{l})$ is confirmed.

4.2.4 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 14 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 14 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. However, the runs with tournament selection were done without any form of elitism, so this may account for the difference.

5 Conclusion

In this paper, we examined two theoretical models from the literature that allow us to make predictions about the scale-up behavior of selecto-recombinative GAs under certain assumptions. The models describe the search of a GA in terms of the growth (by selection) and the recombination (by crossover) of building blocks. This view is suitable for problems of bounded difficulty. The assumptions of the models were turned into practical design rules that a GA-practitioner can follow to design a competent GA. A GA is deemed competent when its solutions satisfy a specified lower bound on quality and its scale-up behavior—the relation between input size and the number of fitness

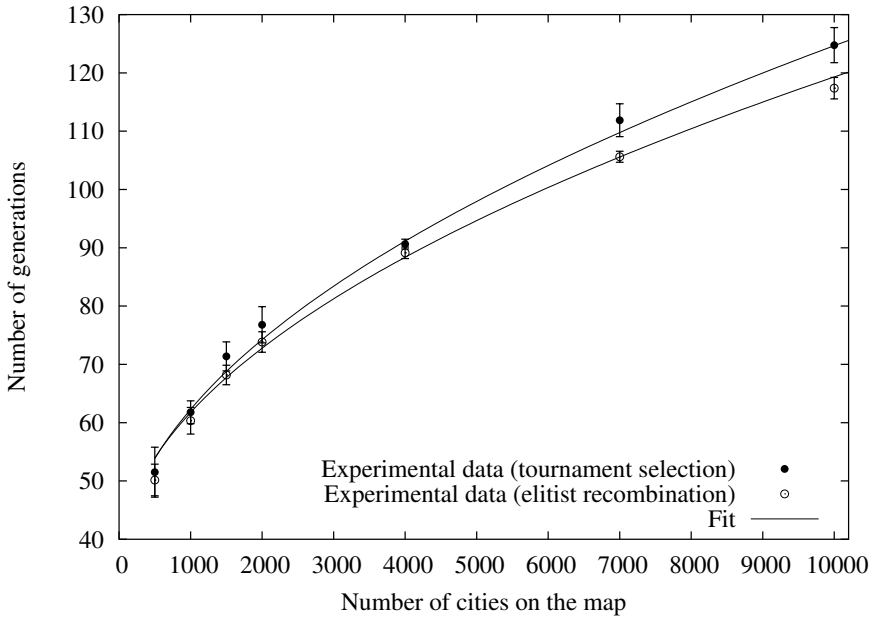


Figure 13: Run time in number of generations of GA when using the critical population size. Shown is experimental data with the fit of the predicted function, for tournament selection and the elitist recombination selection scheme.

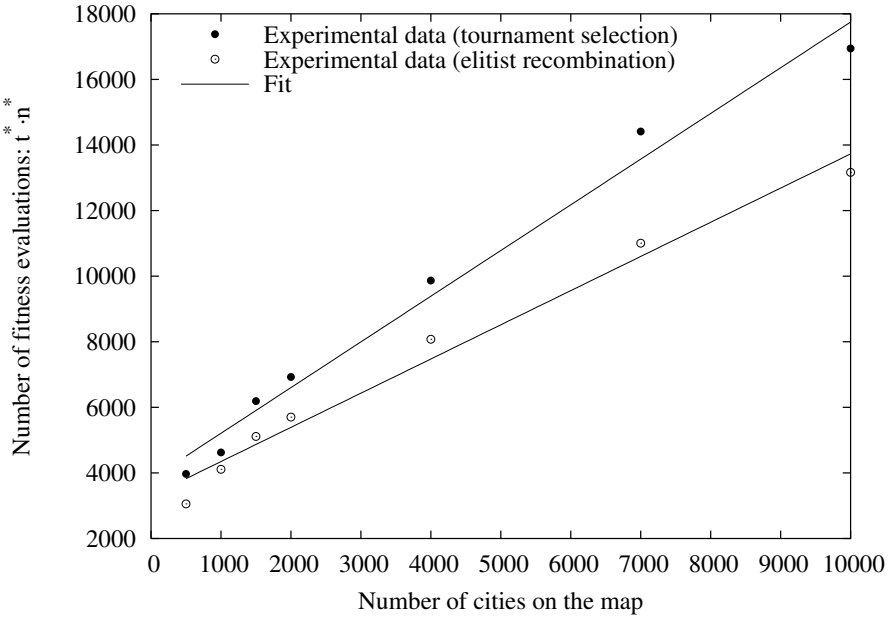


Figure 14: The scale-up behavior of the number of fitness evaluations is linear.

evaluations—is reasonable (for example, a low-order polynomial). To test the practical usefulness of the rules, they 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. The scale-up behavior of the GA was experimentally determined and matched the predictions of the models: the number of fitness evaluations scales linearly with respect to the input size.

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 important real-world problems are either problems of bounded difficulty, or can be solved satisfactorily by assuming they are. Therefore, one can expect that 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 straightforward 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 to some extent but still showed the expected scale-up behavior. This suggests that the design rules can be applied to a broad range of problems. Other examples of the successful application of the design rules are GAs for other cartographic problems such as line simplification and generalization (Van Dijk et al., 2002), and a GA for the automated construction of a Bayesian network from data (Van Dijk et al., 2003). 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 selecto-recombinative GA. For example, for a problem for which the linkage of the building blocks is not readily available, the natural course of action would be to try to learn the linkage. When the GA converges prematurely, one can consider the design of crossover in terms of mixing and disruption, or investigate the building-block supply in the GA. The design rules therefore guide the designer in developing a competent GA. However, the design rules followed fairly straightforwardly from the theoretical results on models of selecto-recombinative GAs. Consequently, we advocate an increased awareness of the theoretic results that are available in the literature, and of their usefulness for practical GA design.

References

T. Bäck. 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, 1995.

P. A. N. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

E. Cantú-Paz et al., editors. *Lecture Notes in Computer Science, Volume 2723: Proceedings of the Genetic and Evolutionary Computation Conference*, 2003. Springer-Verlag.

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

K. Deb and D. E. Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10(4), 1993.

W. Feller. *An Introduction to Probability Theory and its Applications (2nd edition)*. Wiley, 1966.

M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the Seventh Annual ACM Symposium on Computational Geometry*, pages 281–288, 1991.

D. E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3(2):129–152, 1989a.

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

D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989c.

D. E. Goldberg. *The Design of Innovation. Lessons from and for Competent Genetic Algorithms*. Kluwer, 2002.

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

G. Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report 99010, University of Illinois, 1999.

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

T. Jansen and I. Wegener. Real royal road functions - where crossover provably is essential. In L. Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 375–382. Morgan-Kaufmann, 2001.

H. Kargupta. *SEARCH, Polynomial Complexity, And The Fast Messy Genetic Algorithm*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

H. Kargupta. The gene expression messy genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 814–819. IEEE Press, 1996.

F. G. Lobo, D. E. Goldberg, and M. Pelikan. Time complexity of genetic algorithms on exponentially scaled problems. In D. Whitley et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 151–158. Morgan-Kaufmann, 2000.

J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1996.

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

H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions: I. binary parameters. In W. Ebeling et al., editors, *Lecture Notes in Computer Science, Volume 1141: Proceedings of the Parallel Problem Solving from Nature IV Conference*, pages 178–187. Springer-Verlag, 1996.

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

M. Munetomo and D. E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In W. Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 433–440. Morgan-Kaufmann, 1999.

M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, 21(1):5–20, 2002.

R. Poli. 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*, pages 143–163. Morgan-Kaufman, 2000.

G. Raidl. A genetic algorithm for labeling point features. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, pages 189–196, Las Vegas, NV, 1998.

J. Rowe. The dynamical systems model of the simple genetic algorithm. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing Series, pages 31–58. Springer-Verlag, 2001.

K. Sastry and D. E. Goldberg. Scalability of selectorecombinative genetic algorithms for problems with tight linkage. In Cantú-Paz et al. (2003), pages 1332–1344.

J. L. Shapiro, A. Prügel-Bennett, and L. M. Rattray. A statistical mechanical formulation of the dynamics of genetic algorithms. In T. C. Fogarty, editor, *Lecture Notes in Computer Science, Volume 865: Proceedings of the AISB Workshop on Evolutionary Computing*, pages 17–27. Springer, 1994.

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

T. Strijk. *Geometric Algorithms for Cartographic Label Placement*. PhD thesis, Utrecht University, Department of Computer Science, 2001.

T. Strijk, B. Verweij, and K. Aardal. Algorithms for maximum independent set applied to map labelling. Technical Report UU-CS-2000-22, Department of Computer Science, Utrecht University, 2000.

D. Thierens. Selection schemes, elitist recombination, and selection intensity. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms and their Applications*, pages 152–159. Morgan-Kaufmann, 1997.

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

D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms and their Applications*, pages 38–45. Morgan-Kaufmann, 1993.

D. Thierens and D. E. Goldberg. Convergence models of genetic algorithm selection schemes. In Y. Davidor et al., editors, *Lecture Notes in Computer Science, Volume 866: Proceedings of the Parallel Problem Solving from Nature III Conference*, pages 119–129. Springer-Verlag, 1994a.

D. Thierens and D. E. Goldberg. Elitist recombination: An integrated selection recombination GA. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 508–512. IEEE Press, 1994b.

D. Thierens, D. E. Goldberg, and A. G. Pereira. 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, 1998.

S. van Dijk. *Genetic Algorithms for Map Labeling*. PhD thesis, Utrecht University, 2001. URL `http://www.cs.uu.nl/people/steven/download/thesis.pdf`.

S. van Dijk, D. Thierens, and M. de Berg. Using genetic algorithms for solving hard problems in GIS. *GeoInformatica*, 6(4):381–413, 2002.

S. van Dijk, D. Thierens, and L. C. van der Gaag. Building a GA from design principles for learning Bayesian networks. In Cantú-Paz et al. (2003), pages 886–897.

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

B. Verweij. *Selected Applications of Integer Programming: A Computational Case Study*. PhD thesis, Utrecht University, Department of Computer Science, 2000.

M. D. Vose. *The Simple Genetic Algorithm: Foundations and theory*. MIT Press, 1999.

M. Yamamoto, G. Camâara, and L. A. N. Lorena. Tabu search heuristic for point-feature cartographic label placement. *GeoInformatica*, 6(1):77–90, 2002.