A Genetic Algorithm for Searching Spatial Configurations

M. Andrea Rodríguez and Mary Carmen Jarur

Department of Computer Science, University of Concepción

and

Center for the Web Research, University of Chile

Edmundo Larenas 215, Concepción, Chile

{andrea,mjarur}@udec.cl

Abstract

Searching spatial configurations is a particular case of maximal constraint satisfaction problems, where constraints expressed by spatial and non-spatial properties guide the search process. In the spatial domain, binary spatial relations are typically used for specifying constraints while searching spatial configurations. Searching configurations is particularly intractable when configurations are derived from a combination of objects, which involves a hard combinatorial problem. This paper presents a genetic algorithm that combines a direct and an indirect approach to treating binary constraints in genetic operators. A new genetic operator combines randomness and heuristics for guiding the reproduction of new individuals in a population. Individuals are composed of spatial objects whose relationships are indexed by a content measure. This paper describes the genetic algorithm and presents experimental results that compare the genetic versus a deterministic and a local-search algorithm. These experiments show the convenience of using a genetic algorithm when the complexity of the queries and databases do no guarantee the tractability of a deterministic strategy.

Index Terms: Evolutionary computation, genetic algorithm, geographic information systems, constraint satisfaction problems, information retrieval.

1 introduction

Systems that handle spatial information face major challenges due to the complexity of the information and the large amount of data usually involved in such type of systems. Spatial information is inherently complex since it refers to objects of more than one dimension that cannot be organized in a strict one-dimensional order [44, 80]. At the same time, spatial information involves data about the geometry and attributes of objects, which tends to produce large databases. These characteristics of spatial information make the retrieval process an important part of current spatial information systems that pursue user satisfaction in terms of quality and efficiency of responses.

In the context of multimedia information retrieval, an important part of the research effort has concentrated on image retrieval [1, 6, 7, 32, 33, 31, 76, 78]. In these studies, elements within an image are codified based on visual characteristics, such as color, shape, and texture, and this codification is then used for ranking images by their similarity with respect to a user request that has been expressed by a visual example. Unlike previous studies on content-based retrieval of images, this work takes an object-oriented view [74] of spatial information and focuses on searching spatial configurations that involve searching sets of objects whose spatial relations constrain the desired answers. An example of this kind of query is "find a hospital in an urban area adjacent to a park and a highway." This kind of query can be expressed by a command language [19], or by a visual language [8, 20, 75].

A query evaluation in a spatial database combines objects from different thematic layers to construct desired answers based on these objects' spatial relations. This is a type of constraint satisfaction problem (CSP), whose particular goal is not to require that all assignments of variables in the query (i.e., instantiations with objects of the database) satisfy the query constraints, but to optimize the number of satisfied constraints. Thus, this is a maximal constraint satisfaction problem [51]. To solve this type of CSP problem in an exhaustive and complete manner may be intractable for a large database, since one could need to explore all combinatorial possibilities of objects in the database. In some cases, it is possible to reduce the domain being searched, by filtering or indexing objects based on consistency values [56, 58, 69]. This filtering, however, may not be efficient enough, since there are hard combinatorial cases when deterministic search algorithms become intractable [2, 45].

This paper describes a genetic algorithm for searching spatial configurations. Although there are multiple alternatives within the domain of evolutionary computing for treating cases of CSP problems [12, 15, 16, 79], this work uses genetic algorithms because they present a good balance between exploration and exploitation. While genetic algorithms exploit the best solutions for improvement based on fitness functions, they also explore the search space based on the probabilities of mutation and selection [52]. Unlike other heuristic methods, such *simulated annealing* [46] and *tabu search* [37], genetic algorithms handle a *population of solutions* that competes for surviving at any evolutionary cycle. In this sense, we consider that genetic algorithms may incorporate the concepts of simulated annealing and tabu search by allowing individuals to describe particular solutions [52] and by adapting a classical genetic algorithm with new operators that are adjusted to the problem domain.

Our CSP is a case where there exists a large number of possible values that can be assigned to each variable. This characteristic of the problem discourages us to use such an approach as *Ant Colony Optimization* [71], since we cannot have a pre-defined construction graph on which ants lay pheromone trails and choose their paths with respect to probabilities that depend on pheromone trails. Considering an Ant Colony approach without a pre-defined path graph, at each evolutionary cycle ants construct a complete assignment of variables that conform a solution, which in the domain of this work becomes a problem because the number of possible values that can be assigned to variables is still significant large. Based on [79], the best performing Ant Colony Optimization algorithm for many combinatorial problems are hybrid algorithms that combine an ant strategy with a local search. In addition, changes in few objects that compose a solution may strongly affect many objects' relations (i.e., constraints), so it becomes difficult to define local activity rules for agents such as the rules needed in the *swarm intelligence* approach [72].

This work uses a direct approach to treating binary constraints [16], since the characteristics of the problem allow us to redefine an operator that takes advantage of the domain knowledge and leads to an improvement in performance and quality of results. The novelty of this work is the definition of an operator that combines randomness with heuristics over an indexed domain of spatial objects for searching spatial configurations. Using different data sets, the study compares the performance of the genetic versus a deterministic approach and a local search approach to searching spatial configurations.

The organization of the paper is as follows. Section 2 introduces the domain problem. Then, Section 3 discusses related work associated with searching spatial configurations and genetic algorithms that handle constraint satisfaction problems. Section 4 describes the framework for comparing spatial configurations. Section 5 presents the genetic algorithm that has been implemented. Section 6 gives some experimental results. Conclusions are given in Section 7.

2 Problem Domain

A content-based or similarity-based searching of spatial configurations consists in evaluating and ranking configurations based on the content similarity of candidate solutions with respect to a query. In this work, queries are specified with a visual language, such as VisualSeek [76] or Query by Sketch [8, 20], and configurations (i.e., queries or candidate solutions) are sets of objects and sets of constraints defined by the topological relations between objects. Thus, a configuration can be seen as a directed and labeled graph, where nodes are the objects and directed edges are the binary topological relations between objects (Figure 1). Using a graph-based representation, a spatial configuration is a complete graph with n nodes (i.e., objects) and $n(n-1)/2$ edges (i.e., topological relations).

In spatial databases, the combination of objects creates candidate solutions (Figure 2). Queries in these systems are of variable size, and databases contain a large number of objects organized into thematic layers. Since queries are of variable size, it is impractical to a priori create configurations that could be organized and indexed in order to reduce the search domain and speed up the query processing. Thus, objects must be dynamically combined and tested against the variables specified in a user query. In an exhaustive search of most similar configurations, the systems need to perform all n-combinations of N objects, with n being the number of objects in the query and N the number of objects in the database. Thus, for $N \gg n$, the retrieval process is exponential in the size of the query ($O(N^n)$). Similarly, considering a query evaluation as a process of subgraph isomorphism, where the query is a subgraph of the large graph that represents a database, the problem of searching spatial configurations has been proved to be NP-complete [13].

The evaluation of a spatial query composed of a set of objects analyzes spatial constraints, which are based on the objects' spatial components. Spatial components, such as shape, area extent, volume, and density, are often derived from positional information; however, other spatial components, such as spatial relations of adjacency and containment, do not require absolute positional data [48]. These spatial relations represent higher levels of abstraction than positional information [9], since they can be represented by qualitative primitives, such as connectivity [65], and with respect to a qualitative frame of reference (e.g., *A is to the left of B*) [44].

Spatial relations between objects play a fundamental role in spatial information, since such relations refer to the way people perceive spatial configurations, how they reason about such configurations, and how they describe configurations in a variety of languages [24]. In the literature, three major types of spatial relations are usually distinguished [64, 86]: *topological relations* establish the concept of connectivity and are invariant under continuous transformations of rotation, translation, and scaling; *direction relations* are based on the existence of a vector space and are subject to change under rotation, while they are invariant under translation and scaling; and *distance relations* express spatial properties that reflect the concept of a metric and, therefore, change under scaling, but are invariant under translation and rotation. Among these spatial relations, topological relations have been pointed out as particularly important for describing spatial scenes, since they capture the essence of a spatial configuration—topology matters, metric refines [24].

This work concentrates on topological relations between objects in a spatial configuration, in particular, topological relations between regions. Only regions are considered because the spatial objects are represented by the objects' minimum bounding rectangles (MBRs). MBRs are commonly used in spatial information systems for their computational properties and for being usually sufficient for finding objects. Thus, in current spatial information systems, searching for MBRs represents a filter of candidate solutions.

Figure 3 shows the eight topological relations that can be found between two regions [23, 65], organized in a graph that connects conceptual neighbors derived from the concept of gradual changes [21]. For example, *disjoint* and *meet* are two neighboring relations in this graph and, therefore, they are conceptually closer than *disjoint* and *overlap*. Refinements of these topological relations can be also introduced in order to differentiate relations by taking into account metric characteristics of objects, such as relative size and distances [25, 73]. So, for instance, a pair of objects can be seen as further *disjoint* than another pair of objects of the same size if the distance between objects in the first pair is larger than the distance between objects in the second pair.

3 Related Work

There are few studies that address the retrieval of spatial configurations as a problem that is independent of the type and number of constraints. For this work, two related areas of study are spatial information retrieval and genetic algorithms that handle binary constraints.

3.1 Spatial Information Retrieval

Studies on spatial-information retrieval have focused on similarity-based search of spatial configurations [49, 54, 56, 58, 60]. These similarity-based approaches follow a common strategy: (1) they define the set of spatial relations that can be used in a query, (2) they define a similarity measure of spatial relations, and (3) they define a search algorithm for similarity retrieval.

Within the domain of spatial databases, retrieval by structural queries (i.e., by spatial relations) is done on the basis of information consisting of objects stored in relational tables and organized by thematic layers with spatial indexing methods (e.g., R-Tree)[67]. This technique answers queries as cascaded spatial joins and is restrictive to the type of objects and relations [4, 50, 59, 61, 62]. Although these studies provide efficient algorithms for answering queries based on spatial constraints, the use of these algorithms is restricted to query languages with a limited number and type of constraints. They were not intended for processing queries with variable and large number of objects, such as the case of a query by sketch.

Considering query processing as a type of constraint satisfaction problems, Papadias *et al.* [56, 57, 58, 60] address retrieval of spatial configurations without restrictions on the type of objects and relations. They employ deterministic approaches with a forward checking strategy that uses heuristics with a traditional indexing method to restrict the search domain. Although these studies present an alternative for searching spatial configurations in geographic databases, their results are still discouraging for a general and large-scale solution to the problem.

Exploring evolutionary alternatives, Papadias [55] presents a searching algorithm based on genetic operators, where he compromises optimal versus efficient solutions by searching for sub-optimal solutions rather than for the best solution. Papadias' work uses traditional genetic operators with a non-indexing database. A recent work by Arkoumanis *et al.* [2] shows promising results based on two heuristic algorithms: an evolutionary and a hill-climbing algorithm. These two algorithms look for an overall optimal assignment from an indexing database. Unlike this previous work, this paper presents an algorithm that handles a content-based indexing method based on spatial relations rather than positional information, defines a new genetic operator, uses heuristics about the relevance of constraints, and was compared and evaluated by using databases that allow experimental replication.

3.2 Genetic Algorithm for CSP

Genetic Algorithms (GAs) have been successfully applied to a number of optimization problems [3, 10, 36, 53, 63]. Some of these optimization problems can be formalized as CSP problems [47, 51], that is, problems where individuals or the population must satisfy some constraints. GAs, however, do not handle constraints directly, because the genetic operators crossover and mutation are "blind" to these constraints [10]. The main issue in CSP problems that use GAs is, therefore, how to incorporate constraints in an evolution cycle.

Handling binary constraints with genetic algorithms has several alternatives [14]. An indirect strategy considers fitness functions that involve values associated with constraint satisfactions. A common way to define this kind of fitness function is the use of penalty functions [17, 66]. Penalty functions are defined as the penalty for violated constraints or as the penalty for wrongly instantiated variables [70]. The indirect strategy is general and suitable for handling constraints, since it reduces the CSP problem to a simple optimization problem and may incorporate user preferences by adding weights of relevance in the fitness function. This strategy, however, hides the constraints within a global fitness function without ensuring the satisfaction of any constraints. In addition, it requires the definition of weights for constraint violations. This definition may need substantial knowledge about the specific problem and may change during the problem solving process [12]. In order to overcome the problems of defining weights in the penalty function, some studies have proposed to have algorithms that self adjust the weights during the search process based on the number of times constraints are violated. Examples of methods with adaptive fitness functions are: *microgenetic iterative descent* [18] and *stepwise adaptation of weights* [29, 30]. These approaches to adaptive fitness functions seem to have good performance [14]. They have been used in applications where constraints are all equally important such that the runtime adjustment prevents that constraints could never be satisfied. In our domain problem, however, there exists evidence that constraints are not equally important and that the constraints' relevance may affect the perception of the quality of solutions [34, 8].

A different strategy is a direct treatment of constraints in GAs. This strategy adapts the traditional genetic operators by including heuristics in the operators. Some adaptations to classic operators may be: elimination of unfeasible candidates, preservation of feasible candidates, repairing of unfeasible candidates, and creation of new domain-dependent operators. Eiben *et al.* [27, 28] propose two basic operators with heuristics: an *asexual operator* that transforms an individual into a new one considering changes of up to one fourth of the variables and a *multi-parent crossover operator* that generates one offspring using two or more parents. In [15], the two basic operators were compared with the result that the *asexual operator* outperforms the *multi-parent crossover* operator. In addition, different heuristics-based strategies were analyzed in combination with experimental results that compare adaptive fitness functions [15, 14, 30, 35]. The main conclusion from the comparison studies is that effective methods based on genetic algorithms for solving binary constraint problems should incorporate problem knowledge, either in the form of ad-hoc genetic operators and fitness functions or in the form of a local search procedure.

There is a trade-off when using special representations and operators for genetic algorithms. On the one hand, this strategy loses the desirable property of being applied to a number of applications; on the other hand, it may be more efficient than general approaches. This is particularly true when domain knowledge may improve the performance of the process and the quality of the results. In our case, there are considerations that make this particular problem a suitable candidate for combining a direct and an indirect strategy for treating constraint satisfaction. First, by using a content measure it is possible to organize the information in order to avoid the exhaustive search of instances in the database that satisfy a particular constraint. Taking advantage of an indexed spatial database is not an easy task for a traditional genetic algorithm [55], but indexing schemas exist and could be exploited to improve quality and performance of results. Our content-based indexing schema provides basis for defining heuristics in new operators with the goal of making individuals better candidate solutions in subsequent evolution cycles. Second, the query constraints may be sorted by relevance. For example, *non-disjoint* relations are considered more important than *disjoint* relations [34]. This relevance of spatial relations may provide a suitable strategy for defining weights of violated constraints in the fitness function.

4 Content Description and Similarity of Spatial Configurations

Fundamental to the process of searching spatial configurations is the definition of a systematic way to compare these configurations. This study is built upon a previous work for comparing spatial configurations [38, 39] that defines a content measure of topological relations. Using a content measure allows us to characterize spatial relations so that we can compare them and organize them with an indexing schema. This is particularly important for this type of problem, where queries are composed of a variable number of objects and the database cannot be organized by sets of configurations with fixed numbers of objects. In addition to being able to differentiate topological relations, the defined content measure distinguishes among same topological relations, but with different metric characteristics. For example, the content measure recognizes different degrees of *disjointness* and *overlapping* (Figure 4). Thus, this content measure combines metric characteristics of MBRs to obtain unique and continuous values that identify topological relations.

The defined content-measure of topological relations considers three basic primitives over objects' MBRs: (1) areas of individual MBRs and areas of intersection of pairs of MBRs, (2) diagonals of MBRs, and (3) minimum internal and external distances between boundaries of MBRs (i.e., $d_i(\delta b, \delta c)$ and $d_e(\delta a, \delta c)$, respectively) (Figure 5). The intuition behind this measure is that the *distance* between objects is a basic parameter for the refinement of *disjointness*, while the *area* of the objects is a basic parameter for the refinement of *overlapping* (Equation 2).

$$F(a,b) = \frac{area(a,b) - 2area(a \cap b)}{area(a)} + \frac{distance(\delta a, \delta b)}{diagonal(a)} \tag{1}$$

where

$$distance(\delta a, \delta b) = \begin{cases} d_e(\delta a, \delta b) & \text{if } a \cap b = \oslash \\ -d_i(\delta a, \delta b) & \text{if } a \cap b \neq \oslash \end{cases}$$

This content measure is independent of such continuous transformations as scaling, transformation, and horizontal or vertical flipping. The invariance under these continuous transformations is consistent with the definition of topological relations. The content measure is also asymmetric, so describing the arrangement of two objects needs two values, one in each direction of the relation. Figure 6 presents the range of values in 2D that characterizes the topological relations between MBRs. The boundaries of the regions in this figure were determined by considering extreme cases and then creating the corresponding parametric equations. As Figure 6 reflects, all eight topological relations between two extended objects, such as those defined by the 9-Intersection model [22] and the RCC model [65], can be defined in the 2D space that maps values of the content measure (Table 1).

By having a quantitative description of topological relations within a two-dimensional (2D) space, an indexing schema can be used to avoid the exhaustive revision of the database while searching a particular topological relation. This work uses an R-Tree like structure that organizes 2D points in a balanced tree [69]. Unlike traditional indexing schemas that organize objects positions, this schema organizes spatial relations. Rectangular areas (i.e., intermediate nodes in the tree) group points in the space of relations trying to minimize the overlapping areas and building a hierarchical and balanced data structure. A balanced tree is possible because each area of a node includes a minimum and maximum number of points or sub-areas. The tree-based indexing structure used in this work has the following characteristics that are inherited from the commonly used R-Tree structure [43] (Figure 7):

- Each node in the tree, except the root, has between m and M entries, where $m = M/2$.
- For each entry $(mbr, nodedr)$ in a non-leaf node N, mbr is the directory rectangle of a child node of N with node address $nodedr$.
- For each leaf entry (pt, oid_1, oid_2), pt is the point of content-measure values that represent the relation between objects with identifiers oid_1 and oid_2.
- All leaves are at the same level.
- The root has at least two entries (unless it is a leaf).

Although the number of objects' spatial relations in a data set may be very large, the index does not need to store all relations between objects, but only the relations that are considered relevant for a particular application. In this work, relations between close neighboring objects were stored, considering that close neighboring objects are within a certain distance from each other. The maximum distance between two objects is determined as the average distance of neighboring objects. This strategy follows the principle that nearby objects are more related than distant objects [84].

With the indexed space of topological relations, two pairs of objects (g_i, g_j) and (g_k, g_l) are said to hold the same topological relation if $F(g_i, g_j) = F(g_k, g_l)$ and $F(g_j, g_i) = F(g_l, g_k)$. The exact correspondence of relations, however, is unlikely because small changes in the size or shape of objects in the query may affect the metric characteristics of the topological relations. So, exact searching is relaxed by using a threshold value τ, such that two topological relations are considered equivalent if their values of the content measure satisfy

$$|F(g_i, g_j) - F(g_k, g_l)| \leq \tau \wedge |F(g_j, g_i) - F(g_l, g_k)| \leq \tau) \qquad (2)$$

Studies have shown that there is an intrinsic relevance in the relations of objects in a configuration [8, 11, 34]. This relevance is known as the first law of geography "everything is related to everything else, but nearby things are more related than distant things" [84]. Consequently, *non-disjoint* relations are more relevant, since they indicate physical connection between objects [8, 34]. Our content measure defines continuous values, where large values indicate a larger degree of *disjointness* than small values. So, an intuitive relevance degree can be defined by sorting the query constraints from *non-disjoint* to *disjoint* relations. In particular, constraints are sorted in increasing order by the sum $F(g_i, g_j) + F(g_j, g_i)$.

5 Description of the Genetic Algorithm

The Genetic Algorithm (GA) proposed in this paper is based on the results of previous studies that suggest the incorporation of problem knowledge in the definition of ad-hoc fitness functions and new genetic operators [14, 15, 30, 35]. Following the ideas from [14, 15], this work defines a new genetic operator (i.e., an *asexual_reproduction* operator) that combines probability with local search in the exploitation of solutions. The GA uses a performance criterion for evaluation and an initial population of candidate solutions to search for a global optimum. The manipulation of the population is given by a set of genetic operators that work on the population's chromosomes or individuals, which are composed of a set of alleles (i.e., objects). At each generation or cycle of the GA, the new solutions arise from the application of the genetic operators. The fitness function, quantified by the objective function, represents the individuals' ability to survive. A summary of main features of the proposed genetic algorithm based on [14] is shown in Table 2.

The following Subsections describe the population and individual representation, the fitness function, and the genetic operators of the algorithm that searches spatial configurations. This algorithm will be called from now on GASC (Genetic Algorithm for searching Spatial Configurations).

5.1 Representation

In GASC, while objects are explicitly represented, relations (i.e., constraints) are determined through the search process. Spatial objects are the basic components of a population associated with alleles of the population's individuals. So, an individual is a candidate configuration composed of a set of objects (Figure 8). In this representation, spatial objects possess a unique identification, a semantic classification (e.g., an object is a building or a road), and a MBR.

Given a population P with a set A of n individuals, which are composed of m alleles obtained from a set G (i.e., the database), the formal description of the system is

$$P \equiv \bigcup_{i=i}^{|P|=n} a_i; a_i \in A \equiv \bigcup_{j=1}^{|A|=m} g_{i,j}; g_{i,j} \equiv (id_{i,j}, class_{i,j}, mbr_{i,j}) \in G \tag{3}$$

where

$$\begin{aligned} id_{i,j} &\in \mathbb{N} \\ class_{i,j} &\in \mathbb{N} \\ mbr_{i,j} &\equiv \{((x_{ul}, y_{ul}), (x_{lr}, y_{lr})) | x_{ul}, y_{ul}, x_{lr}, y_{lr} \in \mathbb{R} \\ & \quad \forall x, y \in mbr_{i,j}, x \leq x_{lr} \wedge x \geq x_{ul} \wedge y \leq y_{ul} \wedge y \geq y_{lr}\}. \end{aligned}$$

Like configurations in the database, a query q has the same structure as an individual of the population

$$q \equiv \bigcup_{i=i}^{|q|=m} v_i; v_i \equiv (id_i, class_i, mbr_i) \in V. \tag{4}$$

Relations (i.e., constraints) are defined over pairs of objects (i.e., alleles). Queries are pre-processed such that a relation k is less *disjoint* than relation $k+1$. Thus, given the following definition of a constraint r_k and its converse constraint $\bar{r}_k$

$$\forall k, r_k(q) \equiv (v_i, v_j) | v_i \neq v_j \in V, \bar{r}_k(q) \equiv (v_j, v_i), \tag{5}$$

the relations k and $k+1$ in query q satisfy (6), where $F()$ is the value of the content measure.

$$\forall k, (F(r_k(q)) + F(\bar{r}_k(q))) \leq (F(r_{k+1}(q)) + F(\bar{r}_{k+1}(q))) \tag{6}$$

5.2 Fitness Function

The fitness function combines the satisfaction of constraints by handling the sum of satisfied constraints weighted by the relevance of these constraints. For a query with m objects and l constraints, the fitness function is defined according to Equation 7, where τ was set to 0.01.

$$Fitness(a_i, q) = \sum_{k=1}^{l} satisfy(r_k(a_i), r_k(q))(k/l) \tag{7}$$

where

$$satisfy(r_k(a_i), r_k(q)) = \begin{cases} 1 & \text{if } (|F(r_k(a_i)) - F(r_k(q))| \leq \tau) \vee (|F(\bar{r}_k(a_i)) - F(\bar{r}_k(q))| \leq \tau) \\ 0 & \text{otherwise} \end{cases}$$

The goal of this fitness function is to obtain configurations whose objects' topological relations are equivalent to the relations between objects in a query. In order to refine the similarity ranking of candidate configurations, a second function differentiates configurations that have the same fitness value, that is, the same weighted sum of satisfied constraints. This second function measures the degree to which topological relations are equivalent. This is determined by the distance in the space of relations between content-measure values of pairs of objects in the database and pairs of objects in the query (Equation 8).

$$D(a_i, q) = \sum_{g_j, g_k \in a_i; v_j, v_k \in q} \sqrt{(F(v_j, v_k) - F(g_j, g_k))^2 + (F(v_k, v_j) - F(g_k, g_j))^2} \tag{8}$$

5.3 Algorithms and Operators

The operators in a classic genetic algorithm allow us to start with an initial population, which is usually randomly created, and to evolve and improve the initial population by reproduction, crossover, and mutation [40]. There exist multiple alternatives of implementation for genetic operators. These alternatives depend on the experience of previous studies and the specific domain knowledge that may adapt classic operators to obtain precise and efficient results. Before designing the genetic algorithm proposed in this paper, a classic genetic algorithm was implemented and evaluated with large databases. This classic genetic algorithm generated solutions that were very dissimilar to the desired answers. These results and the results previously obtained by Papadias [55] motivate the definition of a specific genetic algorithm whose description follows.

The general structure of GASC is shown in Algorithm 1, where the main difference with respect to a classic genetic algorithm is the substitution of the crossover operator by a new operator that we call *asexual_reproduction*, which has been specifically designed for this application to exploit the advantages of an indexed database of spatial relations. The criteria to stop the evolutionary cycle are to reach the maximum number of generations (i.e., *maximum_generation*) or to have an individual in the population that is an optimal solution (i.e., a solution that has all constraints satisfied).

```
Algorithm 1:
// $c$ is the number of generations
        $c := 0;$
        $population\_initialization(P(c));$
        $population\_evaluation(Pc));$
        while ($c <$ $maximum\_generation$) or (not $optimal\_solution(P(c))$) do [
            $S(c) := selection(P(c));$
            $S'(c) := asexual\_reproduction(S(c));$
            $S''(c) := mutation(S'(c));$
            $population\_evaluation(S''(c));]$
            $P(c) := substitution(S''(c));]$
            $c := c + 1;$
```

The first section of the algorithm corresponds to the initialization of variables, which includes:

- *Population_initialization*, where a random population is created. The initial population was randomly created because the heuristics that could be used in the initialization of the population are used in the reproduction operator, and we found no justification for making the effort of creating an initial population with heuristics that are also applied in the reproduction operator.
- *Population_evaluation*, where the fitness of each individual is calculated with respect to a query.

The next section in the algorithm performs the evolution cycle with the following operators:

- *Selection.* This operator selects individuals from $P(c)$ for reproduction, which are copied to an intermediate population $S(c)$ of the same size than $P(c)$. In this selection, an individual with better probability of surviving may be selected more than once. This probability of surviving is determined based on the individuals' fitness. Among the various type of selection (e.g., rolulette wheel, stochastic universal sampling, linear ranking, tournament) [52], this work uses linear ranking with bias 1.5. This strategy ranks the population by individuals' fitness (i.e., 1 to the worst, 2 to the second worst, and so on) and assigns a proportional probability of selection determined by this ranking and a factor defined in terms of the bias

[41] (Algorithm 2). Rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. The reproductive range is limited so that no individuals generate an excessive number of offspring. Ranking introduces an uniform scaling across the population and provides a simple and effective way of controlling selection pressure [5].

```
Algorithm 2:
// $n$ is the number of individuals in the population $P(c)$
// $S(c)$ is the intermediate population
// $random(1.0/n)$ is a function that returns a random number between 0 and $1/n$
        sort $P(c)$ by fitness in increasing order;
        $acum := 0$;
        $r := random(1.0/n)$;
        $j := 0$;
        for each $a_i \in P(c)$ do [
                $acum := acum + \frac{(2.0-bias)+(2rank(a_i)(bias-1.0)/(n-1))}{n}$;
                while $r < acum$ and $j < n$ do [
                        insert $a_i$ into $S(c)$;
                        $j := j + 1$;
                        $r := r + 1.0/n$]];
```

- *Asexual_reproduction*. This new operator implements a direct treatment of constraints [16] (Algorithm 3). The goal of this operator is that the reproduction of individuals should generate new individuals with one or more constraints that are satisfied. In finding alleles that satisfy constraints, this operator uses the indexed database. The operator is applied over the individuals that were selected. It selects, within an individual, the constraints defined by the pairs of objects with the worst impact on the fitness. The impact of individuals on the fitness is determined by the weighted sum of constraints violated per individual, where for n sorted constraints, the violation of constraint i is given weight $(1 - (i - 1)/n)$. Once a pair of objects is selected, and its corresponding constraint is determined, this pair of objects is replaced by using a random pair of objects that is chosen from all possible pairs that satisfy the constraint. So, each time a pair of objects is replaced, the algorithm searches in the index schema for pairs of objects that satisfy the corresponding constraint.

```
Algorithm 3:
// $a_i$ is an individual $i$ from a selected population $S(c)$
// $g_{i,j}$ is the allele located at $j$ in an individual $i$
// $con[1 \ldots m][1 \ldots 2]$, where $con[1 \ldots m][1]$ are alleles' locations in solutions and
                    $con[1 \ldots m][2]$ are the alleles' numbers of violated constraints
// $r_k(q) \equiv (v_j, v_l)$ is the constraint $k$ that relates a variables $v_j$ with $v_l$ in query $q$
// $r_k(a_i) \equiv (g_{i,j}, g_{i,l})$ is the constraint $k$ that relates object instances $g_j$ with $g_l$
                    in individual $a_i$
// $sort(con[1 \ldots m][1 \ldots 2])$ sorts based on the decreasing numbers of $con[1 \ldots m][2]$
// $tree\_search(v_{con[i][1]}, v_{con[j][1]})$ return a list of instances that satisfy constraint between
                    variables $v_{con[i][1]}, v_{con[j][1]}$
// $(t, u)$ are object instances that are randomly selected from a list of pairs of objects
//$replace(g_{i,con[1][1]}, g_{i,con[2][1]}, t, u)$ replaces alleles $con[1][1]$ and $con[2][1]$ in individual $i$
                    by instances $t$ and $u$, respectively
for each $a_i \in S(c)$ do [
    for $i = 1$ to $m$ do
        $con[i][1] = i$; $con[i][2] = 0$;
    for each $r_k(q) \equiv (v_j, v_l)$ do
        if not $satisfy(r_k(a_i), r_k(q))$ then
            $con[j][2]$ +=1; $con[l][2]$ += 1;
    $sort(con)$;
    $(t, u) := random\_selection(tree\_search(v_{con[1][1]}, v_{con[2][1]}))$;
    $a_i' := replace(g_{i,con[1][1]}, g_{i,con[2][1]}, t, u)$;]
```

As an example of how this algorithm works, consider the query and the candidate solution in Table 3. In this example, the query q is composed of 3 variables v_i and the candidate solution a_k is composed of 3 objects $g_{k,i}$. A query with three variables is characterized by 3 constraints: $r_1 \equiv (v_1, v_2)$, $r_2 \equiv (v_1, v_3)$, and $r_3 \equiv (v_2, v_3)$. By equation 7 between relations in the query and corresponding relations in the candidate solution, constraints r_1 and r_3 are not satisfied. Counting the number of violated constraints where an object participates, a rank of objects in decreasing order is $g_{k,2}$, $g_{k,1}$, and $g_{k,3}$. Thus, the first two objects in this ranking (i.e., $g_{k,2}$ and $g_{k,1}$) are the objects to be replaced, which are replaced by randomly selecting objects among candidate pairs of objects that satisfy the constraints $r_1 \equiv (v_1, v_2)$.

- *Mutation.* This operator plays a secondary role that keeps diversity in the search domain. The operator is applied with a low probability equal to 1 divided by the length of the individual, selecting random individuals and randomly changing one of their alleles. This is the classic implementation of mutation [40].

- *Substitution.* The substitution consists in replacing individuals of a population by the new individuals created via asexual reproduction. The criteria used in this implementation was elitism [52], that is, forcing the preservation of the best individuals by making new individuals replace the worst individuals in the population.

At each generation, the computational cost of GASC is $O(nm^2 + nT(N, f))$, which is affected by the number of individuals selected from the population (n), the number of alleles in individuals (m), and the efficient cost of searching in the R-Tree like structure $(T(N, f))$, with N relations and average degree f. There have been some attempts to determine the efficient cost of searching in a R-tree structure in terms of nodes access when answering a selection query [82, 83, 32]. These efficient-cost models are defined by using the number and density of data rectangles in the data set. In this work, however, the index structure organizes points rather than areas, and the density of the data is not always homogenously distributed so that the relation space does not satisfy the general assumptions made in the cost-efficiency models of an R-Tree structure.

6 Experimental Results

6.1 Data

Two databases were used for the experiments (Table 4): *Utility* and *Cell_Box*. One of them is an available database that is used as experimental data in evaluations of spatial indexing schemas [81]. It represents a real geographic database with a large collection of spatial objects whose geometry are defined by polylines. Real geographic databases usually contain more *disjoint* than *non-disjoint* relations because they store objects that are distributed over a large geographic area. Although the model for comparing configurations considers *disjoint* relations less important than *non-disjoint* relations, the content measure used in this model is well suitable for characterizing and, therefore, for comparing *disjoint* relations.

In order to have a comprehensive set of data, a synthetic database (*Cell_Box*) was created with all possible square cells that fit in a 9x9 box, considering cells whose edge lengths vary from 1 to 9. This last database contains a homogenous distribution of objects over the space where it is possible to realize complex topological configurations (e.g., a configuration with 5 objects, where each object is inside of another object).

The two databases differ in the number and frequency of occurrences of topological relations (Table 5). Table 5 shows the same number of *contains* and *inside* relations as well as the same number of *covers* and *covered_by* relations, since these relations are *converse* relations. In order to reduce the large amount of *disjoint* relations in the real database, the number of *disjoint* relations was reduced by considering a database pre-processing that eliminates *disjoint* relations between objects whose separation is larger than a given normalized distance. Thus, the system handles only relations between an object and its neighbors whose separation is less than or equal to d times the object's diagonal. The value of d was set to 4.0 for the real database. This setting was obtained from analyzing the spatial distribution of neighboring objects in the database and considering a distance with the greatest concentration of neighboring objects. For the database *Cell_Box*, the whole set of relations was used, since objects are distributed homogeneously over the space and there is not a clear distance after which the number of neighbors decreases.

The indexing structure used for organizing the content-measure values is an R-Tree with charge factor equal to 500 (i.e., a maximum number of entries in a node equal to 500). Unable to obtain an analytical computational cost of the R-Tree structure, we analyzed experimentally the number of access to the R-Tree for 1000 different searches created randomly in each database. In all searches, solutions were pairs of objects that satisfied a given topological relations, independently of the positions of objects in the space. The index of database *Utility* has 1341 nodes, whereas the index of database *Cell_box* has 192 nodes. Searches in the database *Utility* visited, on average, 53 nodes (between 1 and 219 nodes), with an average number of solutions of 1210 pairs of objects (between 1 and 6539 pairs of objects). Searches in the database *Cell_box* visited, on average, 26 nodes (between 1 and 60 nodes), with an average number of solutions of 1532 pairs of objects (between 4 and 13200 pairs of objects).

Using the content measure, further refinements of topological relations can be made. The distributions of values of content measures for each database are seen in Figures 9 and 10. These distributions become relevant when analyzing the computational cost of finding or combining candidate pairs of objects that satisfy a given spatial constraint. Relations in the database *Utility* are distributed over the whole domain of possible values and, therefore, its graph looks similar to the general graph that describes the value domain of the content measure (Figure 6). In this graph, however, portion of the space has been eliminated due to the threshold used for the *disjoint* relations. The distribution of values of the content measure for database *Cell_Box* indicates that, within a same type of topological relation, many objects have the same size and shape independently of scale and position, and therefore, the same value of the content measure. Consequently, although database *Cell_Box* consists of objects well distributed over the whole space of 9x9 cells, these objects are of regular shape and size such that points in the relation space create clusters of topological relations.

6.2 Parameter Settings

There are two major forms of setting parameters values of a Genetic Algorithm: *parameter tunning* and *parameter control* [26]. Parameter tunning consists in determining good parameter values before the run of the algorithm. Parameter control, in contrast, changes parameters during the run by using deterministic, adaptive or self-adaptive changes. In the proposed algorithm, possible parameters to be set are the *probability of reproduction* by the asexual_reproduction operator, the *probability of mutation*, the *number of individuals* in the initial population, and the *number of maximum generations*. Although a *parameter control strategy* has advantages over a *parameter tunning strategy* with respect to the dependency of parameters and time effort of parameter setting, this work uses *parameter tunning* for the following reasons:

- The *asexual_reproduction* operator differs from a crossover operator so, available strategies that have addressed parameter control for the determination of the probability of reproduction cannot be directly applied. In the proposed algorithm, an adaptive strategy for determining the probability of allele exchange is implicit in the new operator. The allele's probability for exchange depends on the number of violated constraints where the allele participates. This number of violations changes during the run and will make the probability of an allele exchange to increase as the allele's contribution to the fitness of the corresponding chromosome (i.e., individual) decreases.

- There exist previous studies that have addressed the tunning of the probability of mutation p_m. The formula $p_m = 1/m$ adopted in this work, with m being the length of the bitstring, outperforms other fixed values of p_m [77].

- The number of individuals, i.e., the size of the population, was fixed for the whole run of the algorithm in order to avoid combinations with other forms of control of the *asexual_reproduction* operator that could trigger problems related to the *transitory* behavior of the GASC. Taking the common strategy of adaptive population size based on the merit of fitness [26], which is also based on some kind of parameter control (i.e., the allele exchange probability of the *asexual_reproduction* operator), a poor performance of the reproduction operator would have difficulties for recovering as the size of the population shrinks.

- The GASC uses two stop strategies: (1) it stops the algorithm when the optimal solutions was found or (2) it stops when the number of generations reaches a maximum. To the best of our knowledge, no study has set the maximum number of generations by a parameter control strategy.

The opportunity to work with databases of different number of objects allows us to obtain conclusions concerning the effect of the size and complexity of the databases on the setting of parameters of GASC. The goal is to keep a balance between minimizing the computational cost and obtaining good fitness values. To do so, we ran twenty times GASC with different settings and randomly created queries with 5 and 10 objects (Table 6). The criterion used in selecting these queries was to have examples of contrasting queries in terms of the occurrence number of topological relations in the databases. All queries had an exact matching in a database, that is, they exist in one of the databases. The settings consider a range in the number of individuals from 50 to 300 and, in the number of maximum generations, from 100 to 1000. The maximum of 300 individuals and 1000 generations was defined in order to keep the execution time within a time frame not longer than 30 minutes. In addition, two different probabilities of applying asexual reproduction were analyzed, probability equal to 0.6 and 0.8. The experiment uses a computer with operative system Linux, a processor Pentium IV of 2.4 Mhz, and 1 Gbyte of RAM.

The study of results includes three analyses: (1) variance of results that were obtained in the twenty runs of GASC, (2) fitness values and computational cost depending on the probability of asexual reproduction, number of generations, and number of individuals and (3) sensibility of the computational cost by changing the number of individuals and number of generations given a selected probability of asexual reproduction.

GASC is a stochastic algorithm, because, given the same input, different outputs are possible. To obtain a tendency of results, GASC searches twenty times for solutions of a query, and the best solutions are always selected. As it was indicated above, the goal of the setting is to find good results at a minimal computational cost. A way to decrease this computational cost is to reduce the number of times that GASC is executed for each query. Then, this analysis checks the best fitness values for groups of results after running once, five times, ten times, fifteen times, and twenty times GASC. Figure 11 shows results for different number of executions that presented the major variations among the different queries. In this graph, variations are normalized values between 0 and 100 that represent the differences between the best results of twenty executions with respect to the best results of one, five, ten, and fifteen executions. As the number of executions, the number of individuals, and the number generations increase, less variations of the best results occur. For the results shown in Figure 11, the average difference with respect to the results obtained with twenty executions was 0.55, 0.16, 0.06, 0.04, with standard deviations of 0.47, 0.35, 0.23 and 0.20 for one, five, ten, and fifteen executions of CSGA, respectively. For the other queries, results show minimum variations among number of executions.

Figures 12 and 13 show the average fitness values for settings that vary in the number of generations, number of individuals and probability of asexual reproduction. This analysis considers only the best results among the first five executions of the algorithm for different queries. These values are normalized by the number of constraints satisfied such that 100% means that all constraints where satisfied. A clear tendency is that as the number of individuals and number of generations increase, the fitness improves. Considering results for each query separately, when GASC converges to an optimal solution (i.e., all constrained are satisfied and the fitness is equal to 100%), this fitness is obtained for subsequent next increasing values of individuals and number of generations. These figures do not indicate major differences in the fitness values when using a probability of asexual reproduction equal to 0.6 or 0.8. Figure 14 shows the average computational cost when using a probability of asexual reproduction equal to 0.6 or 0.8. With respect to the computational cost, a probability of asexual reproduction equal to 0.6 outperforms the computational cost of using a probability of asexual reproduction equal to 0.8. Based on the previous results, the last analysis assumes a probability of reproduction equal to 0.6 and 5 executions of the algorithms.

An example of the analysis of sensitivity of the computational cost based on number of individuals and number of generations is shown in Figure 15. This Figure illustrates results of applying GASC in the search of solutions for a query with five object in the database *Utility*. In this figure, cost was measured in terms of the normalized number of constraint evaluations (i.e., values between 0% and 100% of the computational cost). As Figure 15 shows, increasing the number of individuals has a worse impact on the computation cost than increasing the number of generations.

Based on the results of the experiments with different settings, the number of individuals and number of generations was set to 100 and 200, respectively, with five executions of the algorithm, and probability of reproduction of individuals equal to 0.6. The criteria used for making the decision were to minimize the number of individuals and generations while keeping the fitness value about 75% and running time not superior to 10 minutes.

6.3 Evaluation of the Genetic Algorithm

The evaluation of GASC considers comparisons of results for 60 random queries of five and ten objects (i.e., ten and forty-five constraints, respectively) per database. The comparison was made with respect to two classes of algorithms that have been previously used for similar spatial configuration retrieval: (1) a deterministic algorithm (DA) based on a *forward checking* strategy [58] and a local search algorithm (LS) based on a *hill_climbing* strategy [2, 55]. An additional reason to compare GASC to a hill_climbing algorithm was to show that, despite the resemblance of a local search with the *asexual_reproduction* operator, GASC was not a hidden hill_climbing algorithm.

The *forward checking* strategy of DA takes the constraints one by one and searches for pairs of objects that satisfy this constraint [69]. It then performs a join operation [42] to combine the results of the search of objects that satisfy individual constraints. The algorithm skips constraints that cannot be satisfied by any of the candidate solutions, but it forces that at least one of the constraints must be satisfied. DA uses the content-indexing schema so that the instantiation of variable are taken from a compatible domain of variables. Taking into consideration the indexing schema and that all variables in the query are interrelated, no other strategies, such as *conflict-directed backjumping* [85], that enhance the classical forward checking algorithm was applicable. The complexity of this algorithm is NP (i.e., non-deterministic polynomial time), since it depends on the combination of an underdetermined number of pairs of objects that are retrieved from the index structure when searching for solutions of individual constraints.

The *hill Climbing strategy* of LS uses an iterative improvement technique. It starts with a single solution that is created with a random assignment of variables. At each iteration, the solution is modified by the re-instantiation of variables that increase the similarity of the solution with respect to the query. This re-instantiation of variables takes objects that violate the largest number of constraints and replaces them with random selected objects from the domain of variables that satisfy the corresponding constraint. The re-instantiation is similar to the *asexual_reproduction* operator of GASC; however, this re-instantiation in LS moves always in the direction of increasing similarity. At each iteration, the computational cost of LS is $O(m^2+mT(N,f))$, with m^2 being the computational cost of finding the objects that violate the most constraints among the m objects in the solution and $(T(N,f))$ being the efficient cost of searching in the R-Tree with N relations and average degree f. This is the same computational cost of the *asexual_reproduction* operator of the GASC in one life cycle of a single population's individual. The algorithm finishes when it finds an optimal solution or exceeds a maximum number of constraint evaluations, which in this case was set to 1.300.000 evaluations (the double of the average number of constraint evaluations used by the GASC).

The graphs in Figures 16, 18, 17, and 19 show fitness values of results for queries with five and ten objects made to the databases *Utility* and *Cell_Box* with the three algorithms. DA was unable to find solutions in cases where the combinational join process exceeds the computational capability (i.e, over 4 hours of executions or the process runs out of memory). In all other cases, DA finds the optimal solution. LS was always unable to find optimal solutions, and GASC finds in most cases optimal solutions. The percentages of optimal solutions that were found for each algorithm in each database for queries with five and ten objects are shown in Table 7. When GASC did not find optimal solutions, it finds solutions with average fitness of 45% and 10%, and 98% and 91% for queries with five and ten objects in database *Utility*, and queries with five and ten objects in database *Cell_Box*, respectively. In cases when GASC did not find optimal solutions, the algorithm did not find candidate solutions in the database *Utility* that were similar to the queries. In the database *Cell_Box*, in contrast, there exist candidate solutions that were similar, but not necessarily equivalent, to the queries.

Computational cost was measured in terms of the number of constraints violated. Figures 20 and 21 show number of constraint evaluations in the search for queries with 5 objects of DA and GASC in databases *Utility* and *Cell_box*. The number of evaluations for LS was always the same and was set to double the average number of constraint evaluations needed by GASC. We omitted the constraint evaluations for queries with 10 objects, since there were no enough solutions of DA to make an adequate comparison with GASC. In cases with five objects and when DA finds solutions, GASC uses less number of constraint evaluations in the 67% and 87% of queries in databases *Utility* and *Cell_Box*, respectively. The average number of constraints evaluations of GASC for queries with five objects to databases *Utility* and *Cell_Box* was 380,583 and 415,641, respectively.

The components of computational cost differ between DA and GASC. While DA is affected by the combinatorial process of joining partial solutions, GASC is strongly affected by the search in the index structure. The average number of visited nodes in the R-Tree by GASC was 2,649,294 for database *Utility* and 565,332 for database *Cell_Box*, whereas the average number of visited nodes in the R-Tree by DA was 8,674 for database *Utility* and 2,208 for database *Cell_Box*. The combinatorial process in DA is an intrinsic part of the way the algorithm works. The search in the index structure, in contrast, is an independent process of GASC that could be optimized by selecting another index schema that could better adjust to the distribution of points in the relation space. Moreover, being the reproduction of individuals in the population independent of each other, the reproduction in the evolutionary cycle could be a parallel process, which was left out of the scope of this paper.

Although both databases differ drastically in size, the computational cost in terms of constraint evaluations did not reflect significant differences between searching with GASC in the database *Utility* and *Cell_box*. This indicates that it is not only the volume of data what matters, but the distribution of occurrences of topological relations in the database. *Cell_Box* is a small database with topological relations that cannot be further differentiated by metric refinements. Thus, the index structure has many overlapping points in the relation space and, therefore, many occurrences of a particular topological relation. Consequently, the computational cost used for retrieving occurrences of topological relations is proportional to the size of the index schema and to the number of occurrences that are retrieved (i.e., density of points in the relation space).

7 Conclusions

This work has presented a genetic algorithm for searching spatial configuration. This algorithm is based on an *asexual_reproduction* operator that replaces the classical *crossover* operator. The new operator treats constraint satisfaction between spatial objects and uses a content-based indexed database of objects spatial interrelations. Thus, the genetic algorithm combines randomness with a controlled search domain. The index schema controls the search domain, since only objects that satisfy query constraints are retrieved. As main conclusion, the genetic algorithm is a good alternative to solve this type of searching problem when the complexity of the database makes a deterministic approach intractable. In such cases, the computational cost of the genetic algorithm is always bounded and affected by the number of individuals and generations, and by the time of searching in the indexing structure of the database.

The experiment for setting the parameters of the algorithm indicates that increasing the number of individuals has a stronger impact on the computational cost than increasing the number of generations. The algorithm converges to solutions as the number of individuals and number of generations increase. Increasing the number of constraints affects the performance of the algorithm; however, the effect of the number of constraints on the performance of the algorithm can also be handled by the number of constraints that are satisfied per individual in the evolutionary cycle.

The genetic algorithm always outperforms the hill_climbing algorithm, and in most cases, it also outperforms the deterministic algorithm. The genetic algorithm is able to find solutions when the deterministic strategy exceeds the computational capability, given good results in over 85% of the cases. The deterministic algorithm, on the other hand, always finds optimal solutions. The main issue is to characterize the complexity of the database to make a correlation between this characterization and the performance of the genetic and deterministic algorithms. The deterministic strategy is less efficient when a topological relation, with its corresponding metric refinement, has many occurrences in the database. In such case, the combinatorial process of joining constraint satisfactions may become intractable.

In terms of computational cost measured by the number of constraints, the genetic algorithm needs on average less number of constraint evaluations than the deterministic strategy. On the other hand, the genetic algorithm requires a large number of searches in the indexing structure. We visualize some strategies to improve the computational cost of the genetic algorithm. The first strategy is to optimize the search process in the index structure. This optimization should consider a structure that handles points that are not homogenously distributed over the space of relations. A second strategy is to implement the genetic algorithm on a distributed network such that reproduction of individuals becomes a parallel process. Finally, strategies for pre-processing the query that reduce the number of constraints [68] may also be considered to simplify user queries.

Acknowledgment

This work has been partially funded by Millenium Nucleus Center for Web Research, grant P01-029-F, Mideplan, Chile.

References

[1] I. Ahmad and W. Grosky. Spatial similarity-based retrievals and image indexing by hierarchical decomposition. In *International Database and Engineering Applications Symposium*, Montreal, Canada, 1997.

[2] D. Arkoumanis, M. Terrovitis, and L. Stamatogiannakis. Heuristic algorithms for similarity configuration retrieval in spatial databases. In *Helenic Conference on Artificial Intelligence*, pages 141–152, Thessalonia, Greece, 2002.

[3] T. Bäck. *Proceedings of the Seventh International Conference on Genetic Algorithms.* Morgan Kaufmann, 1997.

[4] L. Becker, A. Giesen, K. Hinrichs, and J. Vahrenhold. Algorithms for performing polygonal map overlay and spatial join on massive data sets. In R. H. Güting, D. Papadias, and F. Lochovsky, editors, *Advances in Spatial Databases-6th International Symposium, SSD '99, Hong Kong, China, vol. 1651*, pages 270–285, Hong-King, China, 1999. Springer-Verlag.

[5] G. Bilchev and I. Parmee. *Evolutionary Computing*, chapter The Ant Colony Methaphor for Searching Continuous Design Spaces, pages 25–39. Springer Verlag, Sheffield, UK, 1995.

[6] A. Bimbo, E. Vicario, and D. Zingoni. A spatial logic for symbolic description of image contents. *Journal of Visual Languages and Computing*, 5:267–286, 1994.

[7] A. D. Bimbo, E. Vicario, and D. Zingoni. Symbolic description and visual querying of image sequences using spatio-temporal logic. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):609–621, 1995.

[8] A. Blaser. *Sketching Spatial Queries.* PhD thesis, Department of Spatial Information Science and Engineering. Orono, ME: University of Maine, 2000.

[9] T. Bruns and M. Egenhofer. Similarity of spatial scenes. In M. Kraak and M. Molenaar, editors, *Seventh International Symposium on Spatial Data Handling (SDH '96)*, pages 4A.31–42, Delft, The Netherlands, 1996.

[10] L. Chambers. *The Practical Handbook of Genetic Algorithms: Applications.* Chapman Hall, 2001.

[11] E. Clementini, J. Sharma, , and M. Egenhofer. Modeling topological relations: Strategies for query processing. *Computers and Graphics*, 18(6), 1994.

[12] C. Coello. Theoretical and numerical constraints handling techniques used with evolutionary algorithms. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

[13] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[14] B. Craenen, A. Eiben, and J. Hemert. Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 7(5):424–444, 2003.

[15] B. Craenen, A. Eiben, and E. Marchiori. Solving constraint satisfaction problems with heuristic-based evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 1571–1577, California, USA, 1999. IEEE Press.

[16] B. Craenen, A. Eiben, and E. Marchiori. *The Practical Handbook of Genetic Algorithms Applications*, chapter How to handle Constraints with Evolutionary Algorithms, pages 341–362. Chapman Hall CRC, Boca Ratón, FI, 2001.

[17] D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications.* Springer-Verlag, 1997.

[18] G. Dozier, J. Bowen, and D. Bahler. Solvig small and large constraint satisfaction problems usign a heuristic-based microgenetic algorithms. In *First Conference on Evolutionary Computation*, pages 306–311, 1994.

[19] M. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[20] M. Egenhofer. Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4):403–424, 1997.

[21] M. Egenhofer and K. Al-Taha. Reasoning about gradual changes of topological relations. In A. Frank, I. Campari, and U. Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Lecture Notes in Computer Science 639*, pages 169–219, Pisa, Italy, 1992. Springer-Verlag.

[22] M. Egenhofer, E. Clementini, and P. Di Felice. Topological relations between regions and holes. *International Journal of Geographic Information Science*, 8(2):129–142, 1994.

[23] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[24] M. Egenhofer and D. Mark. Naive geography. In Frank and W. Kuhn, editors, *Spatial Information Theory: A Theoretical Basis for Geographic Information Systems, International Conference COSIT'95, Lecture Notes in Computer Science*, pages 1–14, Semmering, Austria, 1995. Springer-Verlag.

[25] M. Egenhofer and A. Shariff. Metric details for natural-language spatial relations. *ACM Transactions on Information Systems*, 16(4):295–321, 1998.

[26] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[27] A. Eiben, P.-E. Raué, and Z. Ruttkay. Heuristic genetic algorithms for constrained problems. Technical Report IR-337, Vrije Universiteit Amsterdam, The Netherlands, 1993.

[28] A. Eiben, P.-E. Raué, and Zs. Ruttkay. Solving constraint satisfaction problems using genetic algorithms. In *First Conference on Evolutionary Computation*, pages 542–547, 1994.

[29] A. Eiben, J. vam der Hauw, and J. van Hamert. Graph coloring with adapative evolutionary algorithms. *Journal of Heuritics*, 4(1):25–46, 1998.

[30] A. Eiben, J. van Hemert, E. Marchiori, and A. Steebeek. Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P/ Schwefel, editors, *5th Conference on Parallel Problem Solving from Nature. LNCS 1498*, pages 196–205. Springer-Verlag, 1998.

[31] E. El-Kwae and M. Kabuka. A robust framework for content-based retrieval by spatial similarity in image databases. *ACM Transactions on Information Systems*, 17(2):174–198, 1999.

[32] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petrovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[33] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petrovik, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

[34] J. Florence and M. Egenhofer. Destribution of topological relations in geographic datasets. In *ACSM/ASPRS*, Baltimore, MD, 1996.

[35] P. Galiner and J.-K Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3:379–397, 1999.

[36] M. Gen and R. Cheng. *Genetic Algorithms and Engineering Design.* John Wiley and Sons, 1997.

[37] F. Glover and M. Laguna. *Tabu Search.* Kluwer, London, 1997.

[38] F. Godoy and A. Roddríguez. Defining and comparing content measures of topological relations. *To appear in GeoInformatica*, 2004.

[39] F. Godoy and A. Rodríguez. A quantitative description of spatial configurations. In D. Richardson and P. van Ossterom, editors, *Spatial Data Handling*, pages 299–311, Ottawa, Canada, 2002. Springer Verlag.

[40] D. Goldberg. *Genetic Algorithm in Searching, Optimization, and Machine Learning.* Addison Wesley, 1989.

[41] J. Grefenstette. *Handbook of Evolutionary Computation*, chapter Rank-Based Selection, page C.2.4. Institute of Physics Publishing, IOP Publishing, Bristol, UK, 1998.

[42] R. Gutting. Geo-relational algebra: A model for query language for geometric database systems. In J. Schmidt, S. Ceri, and M. Missikoff, editors, *Advances in Database Technology, LNCS 303*, pages 506–527, Venice, Italy, 1988. Springer-Verlag.

[43] A. Guttman. A dynamic index structure for spatial searching. In *International Symposium on the Management of Data, ACM SIGMOD*, pages 45–57. ACM Press, 1984.

[44] D. Hernández. *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin, 1994.

[45] M. Jarur and A. Rodríguez. A non-deterministic versus deterministic approach to searching spatial configurations. In A. Abraham, J. Ruiz del Solar, and M. Koppen, editors, *Soft Computing Systems*, pages 602–611, Amsterdam, The Netherlands, 2002. IOS Press.

[46] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[47] P. Ladkin and R. Maddux. On binary constraint problems. *Journal of the ACM*, 41(3):435, 1994.

[48] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, San Diego, CA, 1992.

[49] S. Lee and F. Hsu. Spatial reasoning and similarity retrieval of images using 2d c-strings knowledge representation. *Pattern Recognition*, 25(3):305–318, 1992.

[50] N. Mamoulis and D. Papadias. Integration of spatial join algorithms for processing multiple inputs. In *ACM SIGMOD International Conferences on Management of Data*, Pennsylvania, 1999.

[51] P. Meseguer. Constraint satisfaction problems: an overview. *AICOM*, 2(1):3–17, 1989.

[52] Z. Michalewicz. *Genetic Algorithm + Data Structures = Evolution Programs*. Springer Verlag, 1994.

[53] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[54] M. Nabil, A. Ngu, and J. Shepherd. Picture similarity retrieval using 2D projection interval. *IEEE Transactions on Knowledge and Data Engineering*, 8(4), 1996.

[55] D. Papadias. Hill climbing algorithms for content-based retrieval of similar configurations. In *ACM Conference on Information Retrieval*, Athens, Greece, 2000. ACM Press.

[56] D. Papadias, D. Arkoumanis, and N. Karacapilidis. On the retrieval of similar configurations. In T. Poiker and N. Chrisman, editors, *8th International Symposium on Spatial Data Handlin*, pages 510–521, Vancouver, 1998. International Geographical Union.

[57] D. Papadias, P. Kalnis, and Mamoulis. Hierarchical constraint satisfaction in spatial databases. In *Annual Meeting of the AAAI*, Orlando, FI, 1999.

[58] D. Papadias, N. Mamoulis, and V. Delis. Algorithms for querying spatial structure. In *24th VLDB Conference*, pages 546–557, New York, 1998.

[59] D. Papadias, N. Mamoulis, and Y. Theodoridis. Constraint-based processing of multiway spatial joins. *Algorithmica*, Special Issue on Algorithms for GIS, 1999.

[60] D. Papadias, M. Mantzouroguannis, P. Kalnis, N. Mamoulis, and I. Ahmad. Content-based retrieval using heuristic search. In *ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 168–175, Berkeley, CA, 1999.

[61] A. Papadopoulos, P. Rigaux, and M. Scholl. A performance evaluation of spatial join processing strategies. In R. H. Güting, D. Papadias, and F. Lochovsky, editors, *Advances in Spatial Databases-6th International Symposium, SSD '99*, pages 286–307, Hong-King, China, 1999. Springer-Verlag.

[62] H.-H. Park, G.-H. Cha, and C.-W. Chung. Multi-way spatial joins using R-Trees: Methodology and performance evaluation. In R. H. Güting, D. Papadias, and F. Lochovsky, editors, *Advances in Spatial Databases-6th International Symposium, SSD '99*, pages 229–250, Hong-King, China, 1999. Springer-Verlag.

[63] V. Porto, N. Saravanan, D. Waagen, and A. Eiben. *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, vol. 1447*. Springer-Verlag, 1998.

[64] D. Pullar and M. Egenhofer. Toward formal definitions of topological relations among spatial objects. In *Third International Symposium on Spatial Data Handling*, Sydney, Australia, 1988.

[65] D. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swarthout, editors, *Principles of Knowledge Representation and Reasoning, KR '92*, pages 165–176, Cambridge, MA, 1992. Morgan Kaufmann.

[66] J. Richardson, M. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms wih penalty functions. In *Third International Conference on Genetic Algorithms*, pages 191–197, George Madison University, 1989. Morgan Kaufmann.

[67] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases: with Application in GIS*. Academic Press, San Diego, CA, 2002.

[68] A. Rodríguez and M. Egenhofer. Query pre-processing of topological constraints: Comparing a composition-based with a neighborhood-based approach. In T. Hadzilacos, Y. Manolopoulos, J. Roddick, and Y. Theodoridis, editors, *Advances in Spatial and Temporal Databases LNCS 2750*, pages 362–379, Santorini Greece, 2003. Springer-Verlag.

[69] A. Rodríguez and F. Godoy. A content-based approach to searching spatial configurations. In M. Egenhofer and D. Mark, editors, *GIScience 2002, Lecture Notes en Computer Science*, pages 260–275, Berlin, 2002. Springer-Verlag.

[70] T. Runnarson and X. Yao. Constrained evolutionary optimization: The penalty function approach. In *Evolutionary Optimization*, pages 87–113, Norwell, MA, 2002. Kluwer.

[71] L. Schoofs and B. Naudts. An colonies are good solving constraint satisfaction problems. In *Congress on Evolutionary Computation*, pages 1190–1196, Piscataway, New Jersey, 2000. IEEE Press.

[72] L. Schoofs and B. Naudts. Swarm intelligence on the binary constraint satisfaction problem. In *Congress on Evolutionary Computation*, pages 1444–1449, Picataway, New Jersey, 2002. IEEE Press.

[73] R. Shariff. *Natural-Language Spatial Relations: Metric Refinements of Topological Properties.* PhD thesis, University of Maine, Orono, ME, 1996.

[74] S. Shekhar, M. Coyle, B. Goyal, D.-R. Liu, and S. Sarkar. Data models in geographic information systems. *Commubications of the ACM*, 40(4):103–111, 1997.

[75] J. Smith and S.-F. Chang. Visually searching the Web for content. *IEEE Multimedia*, pages 12–20, 1997.

[76] J. Smith and S.-F. Chang. Integrated spatial query and feature image query. *Multimedia Systems*, 7:129–140, 1999.

[77] J. Smith and T. Fogarty. Self-adaptation of mutation rates in a steady-state genetic algorithm. In *3rd IEEE Conference on Evolutionary Computation*, pages 318–323. IEEE Press, 1996.

[78] J. Smith, R. Mohan, and C.-S. Li. Content-based transcoding of images in the internet. In *IEEE International Conference on Image Processing ICIP*, Chicago, IL, 1998.

[79] C. Solnon. Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutonary Computation*, 6(4):347–357, 2002.

[80] O. Stock. *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[81] Y. Theodoridis. *The Rtree Portal: Code, Data Collection, Publications [http://www.rtreeportal.org/]*, 2003.

[82] Y. Theodoridis and T. Sellis. A model for the prediction of R-Tree performance. In U. Dayal, editor, *Symposium on Principles of Database Systems*, pages 161–171, Montreal, Canada, 1996. ACM Press.

[83] Y. Theodoritis, E. Stefanakis, and T. Sellis. Efficient cost models for spatial queries using R-Trees. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):19–32, 2000.

[84] W. Tobler. *Philosophy in Geography*, chapter Cellular Geography. D. Reidel Publishing Company, Dordrecht, Holland, 1979.

[85] J. van Hemert. Comparing classical methods for solving binary constraint satisfaction problems with state of the art evolutionary computation. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, editors, *Applications of Evolutionary Computing, LNCS 2279*, pages 81–90, Kursale, Ireland, 2002. Springer-Verlag.

[86] M. Worboys. A geometric model for planar geographical objects. *International Journal of Geographical Information Systems*, 6(5):353–372, 1992.

List of Figures

List of Tables

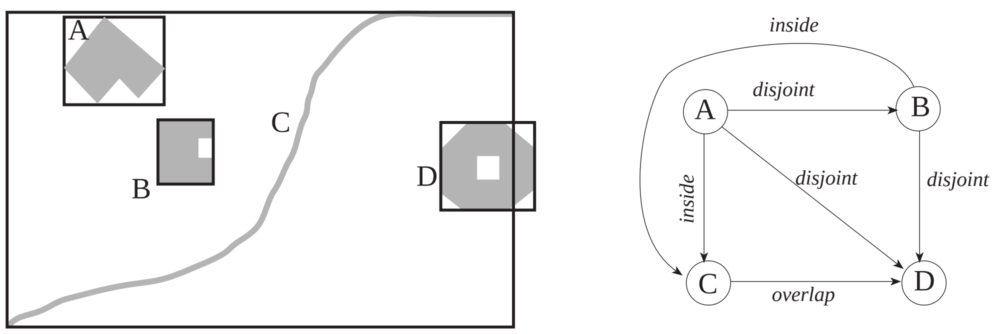


Figure 1: Graph representation of a configuration.

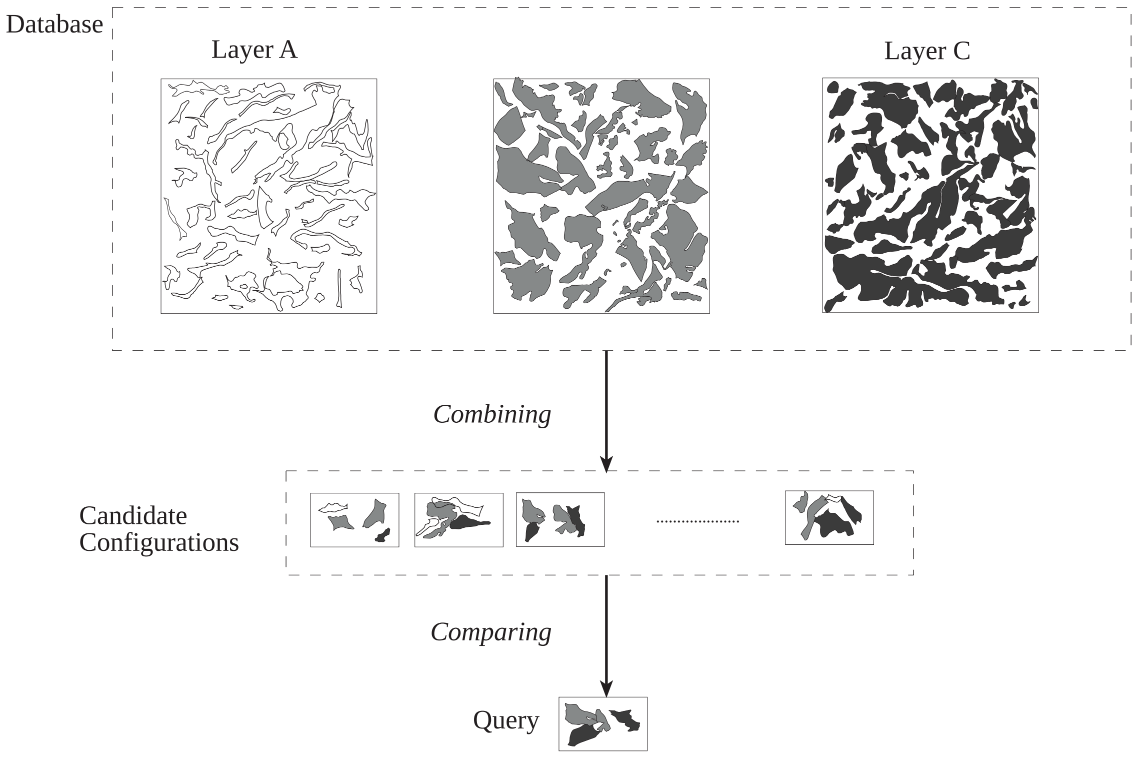


Figure 2: Similarity-based searching of spatial configurations in spatial databases.

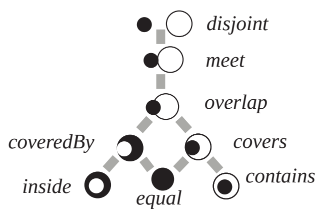


Figure 3: The eight topological relations between regions organized by their conceptual neighborhoods [21].

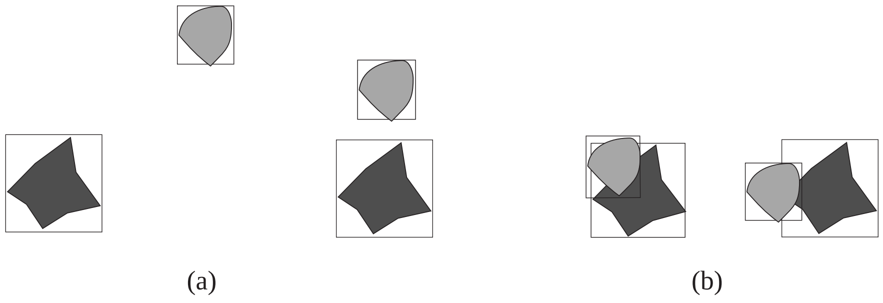


Figure 4: Metric differences of topological relations: (a) degree of *disjointness* and (b) degree of *overlapping*.

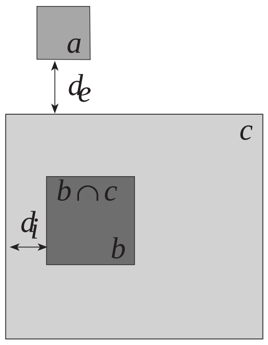


Figure 5: Primitives of the content measure.

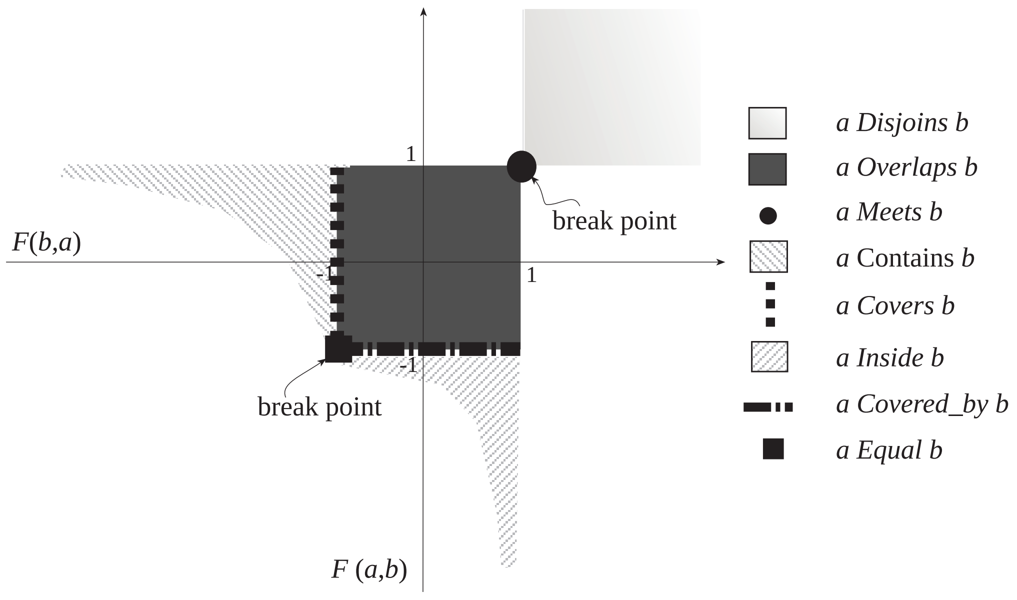


Figure 6: Values of the content measure that are classified into eight topological relations between regions.

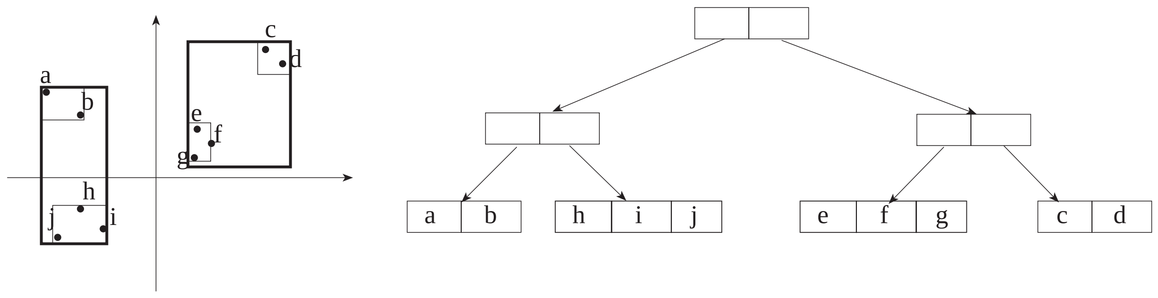


Figure 7: R-Tree of content-measure values with charge factor M equal to 5.

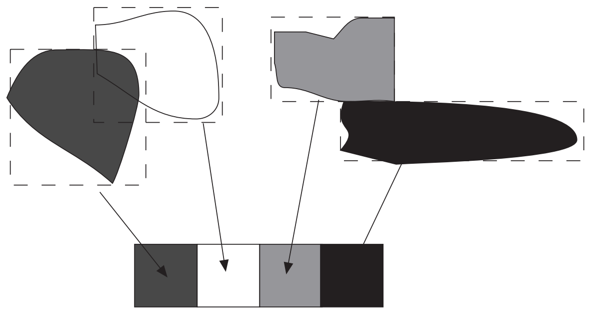


Figure 8: Spatial objects as alleles of the population's individuals.

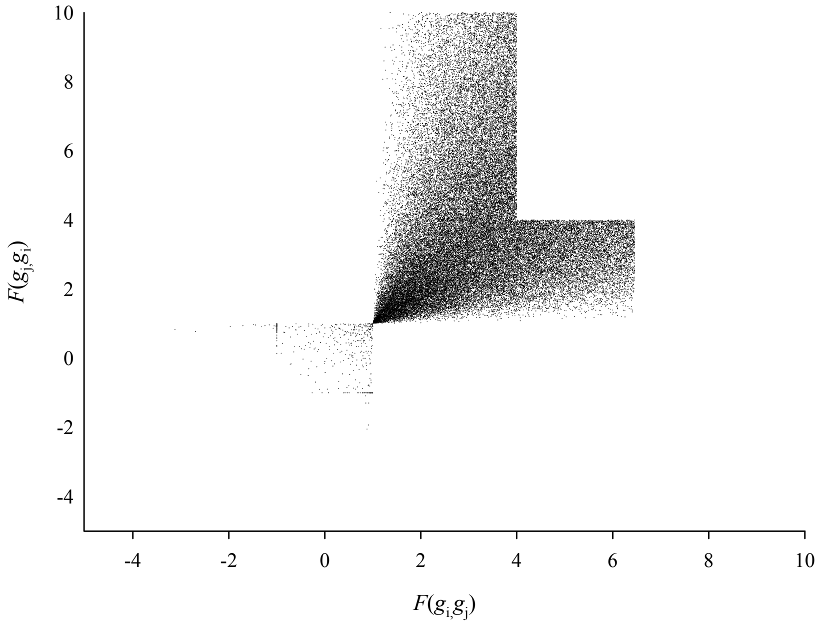


Figure 9: Distribution of content-measure values for database Utility.

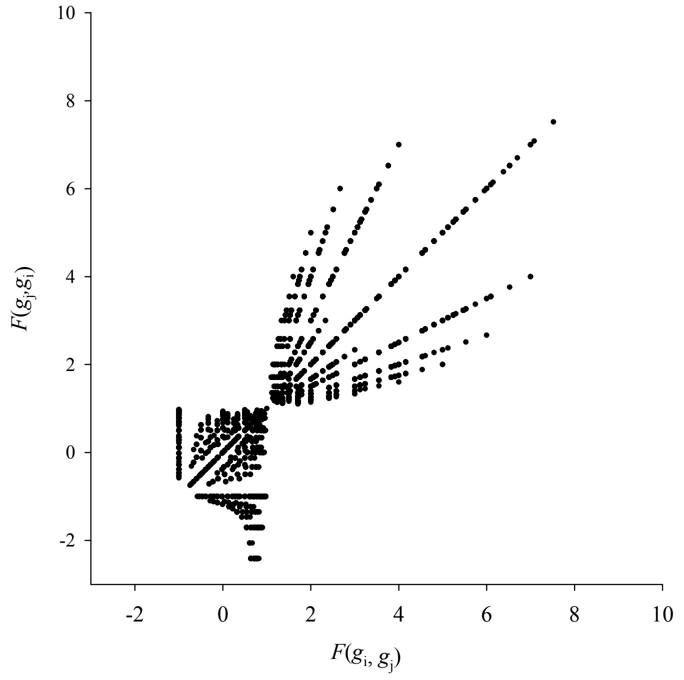


Figure 10: Distribution of content-measure values for database Cell_box.

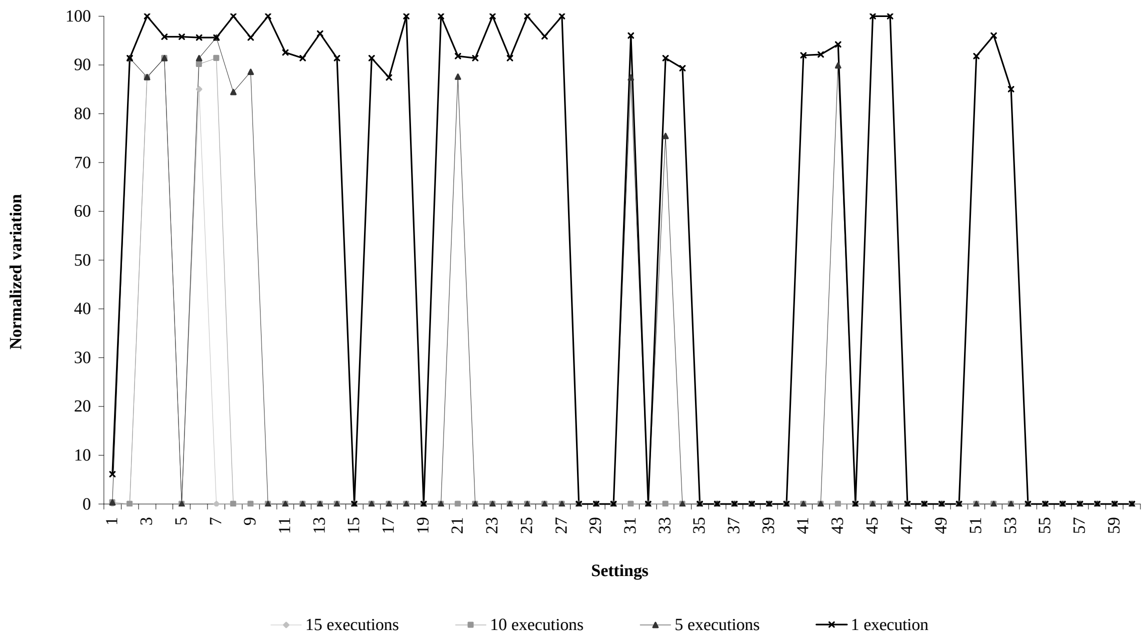


Figure 11: Variations of best results with respect to twenty executions for different settings: settings 1-10: 50 individuals with number of generations that varies from 100 to 1000, settings 11-20: 100 individuals with number of generations that varies from 100 to 1000, settings 21-30: 150 individuals with number of generations that varies from 100 to 1000, settings 31-40: 200 individuals with number of generations that varies from 100 to 1000, settings 41-50: 250 individuals with number of generations that varies from 100 to 1000, and settings 51-60: 300 individuals with number of generations that varies from 100 to 1000.

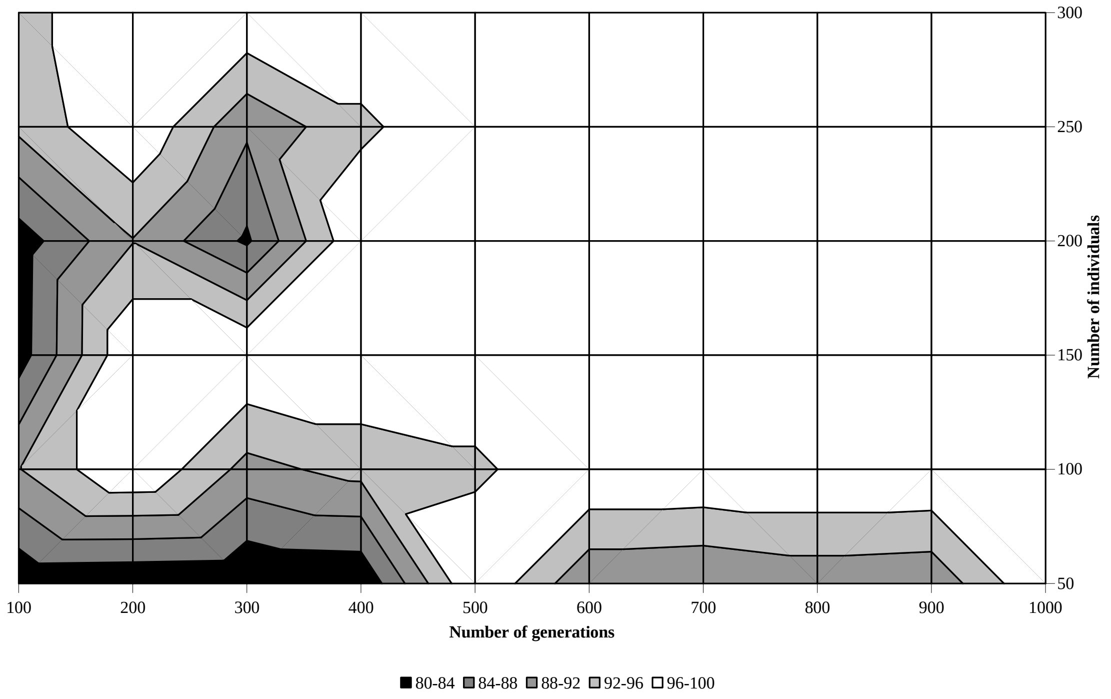


Figure 12: Average fitness values with variations in number of individuals and number of generations for probability of asexual reproduction equal to 0.6.

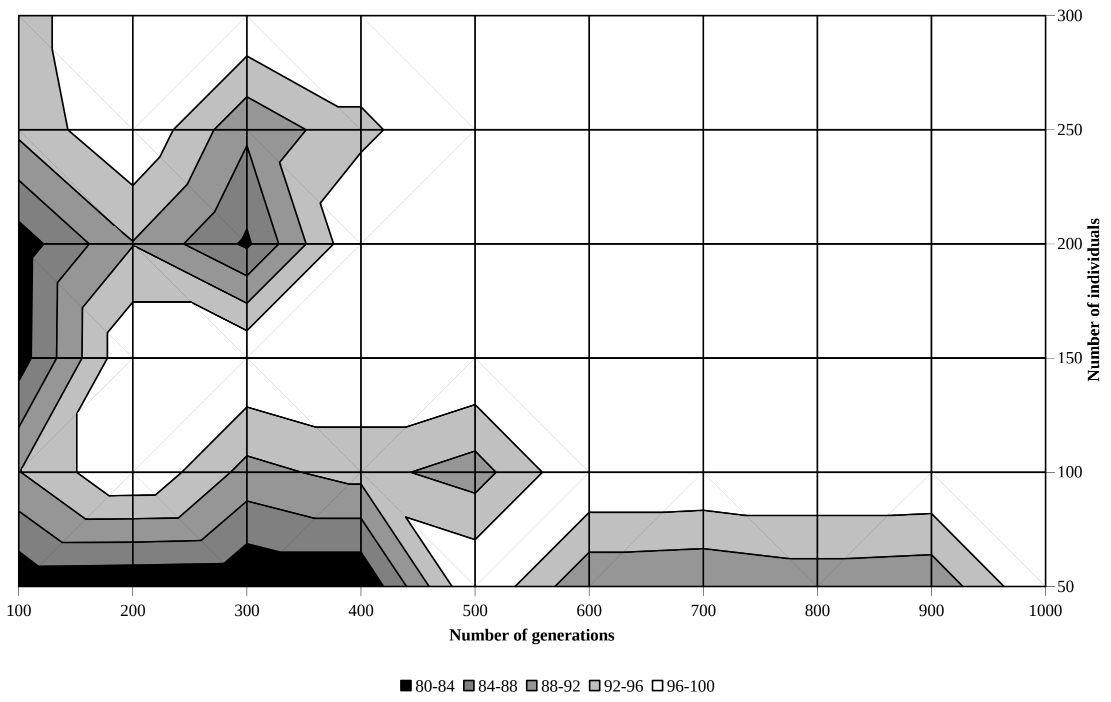


Figure 13: Average fitness values with variations in number of individuals and number of generations for probability of asexual reproduction equal to 0.8.

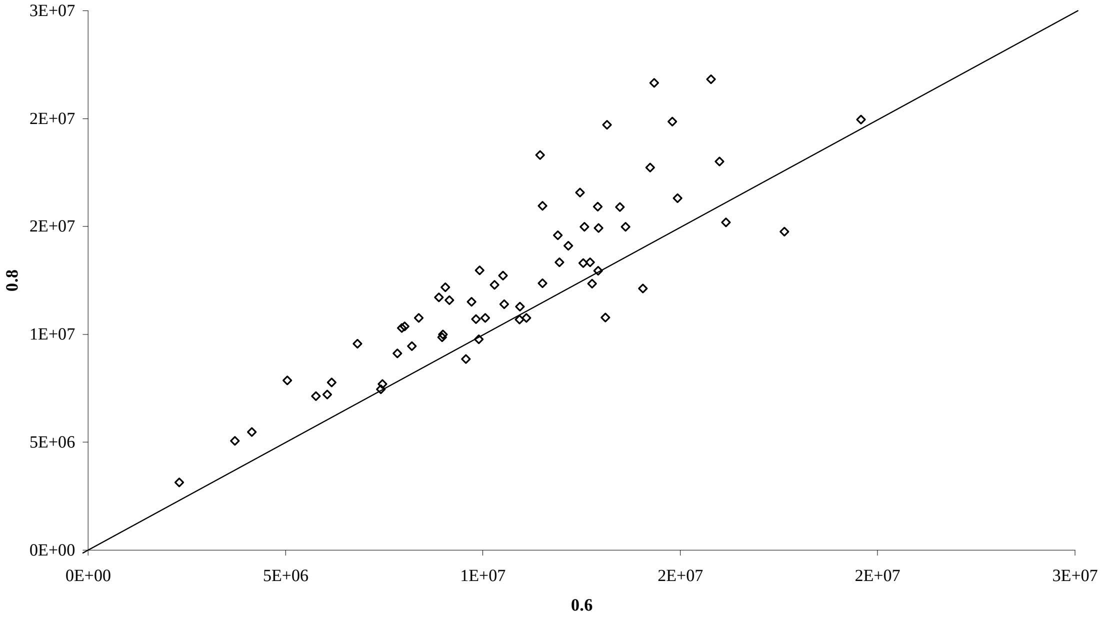


Figure 14: Comparing average number of constraint evaluations for probability of asexual reproduction equal to 0.6 and 0.8.

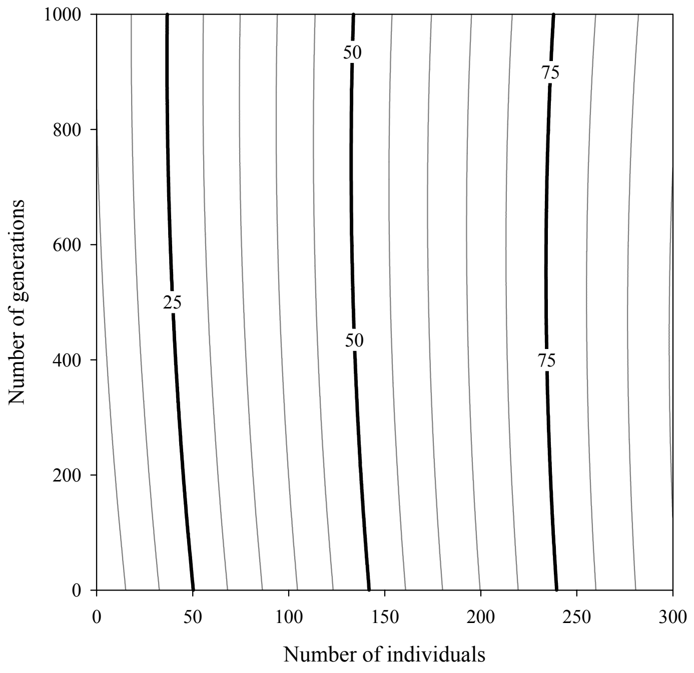


Figure 15: Sensitivity of the computational cost in terms of number of individuals and number of generations.

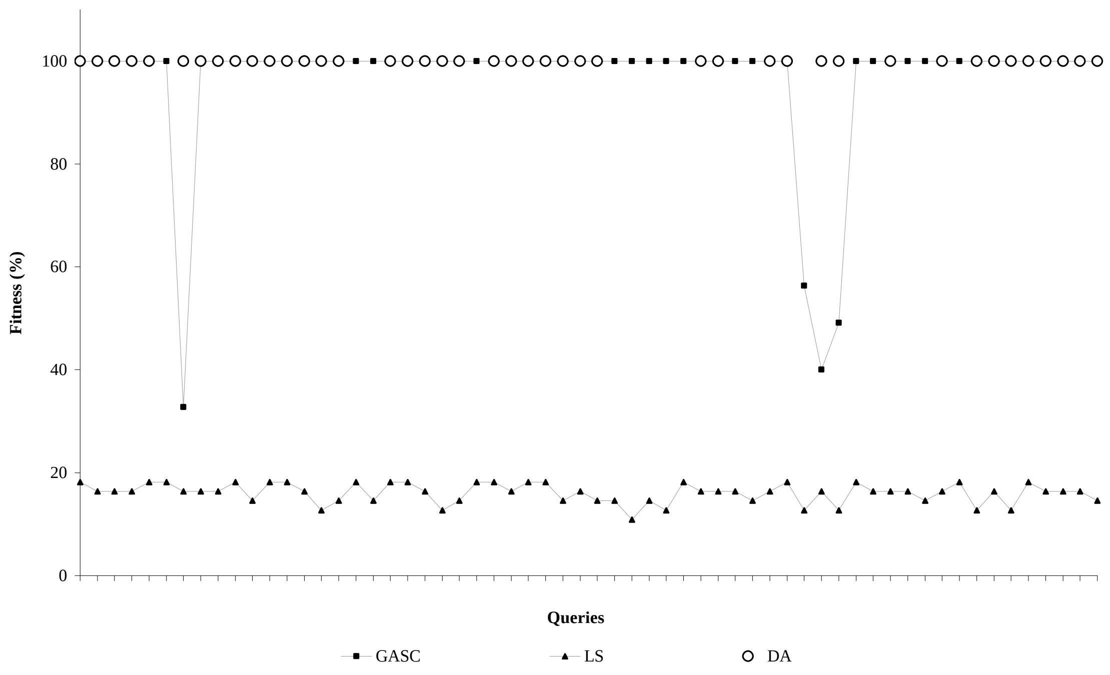


Figure 16: Fitness values for queries with five objects in database Utility.

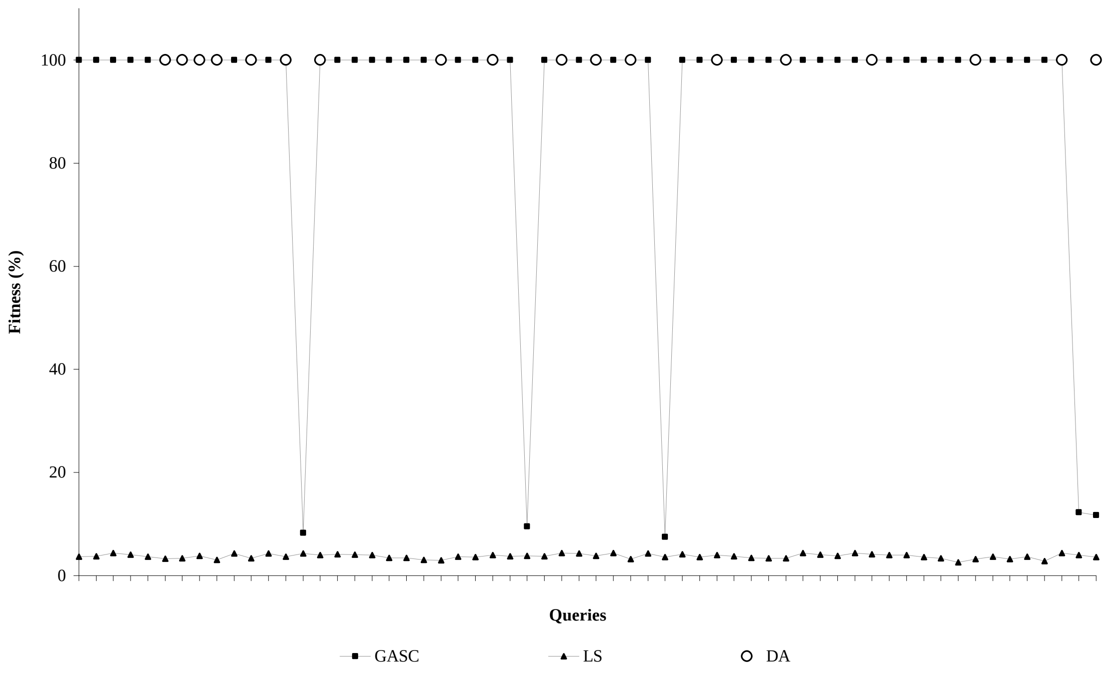


Figure 17: Fitness values for queries with ten objects in database Utility.

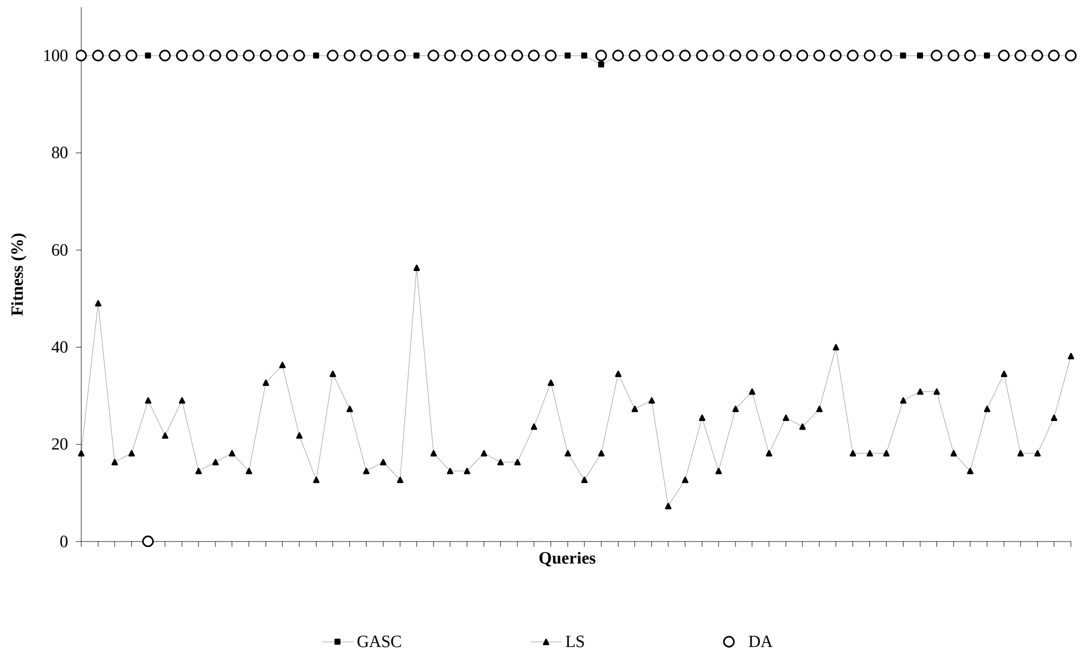


Figure 18: Fitness values for queries with five objects in database Cell_Box.

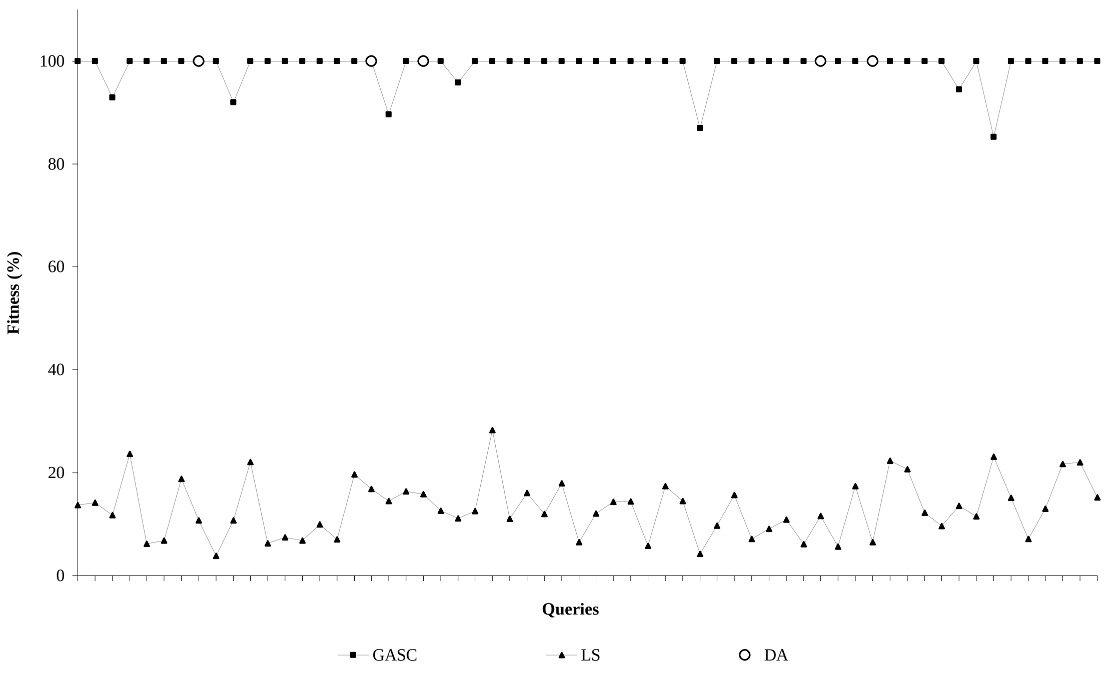


Figure 19: Fitness values for queries with ten objects in database Cell_Box.

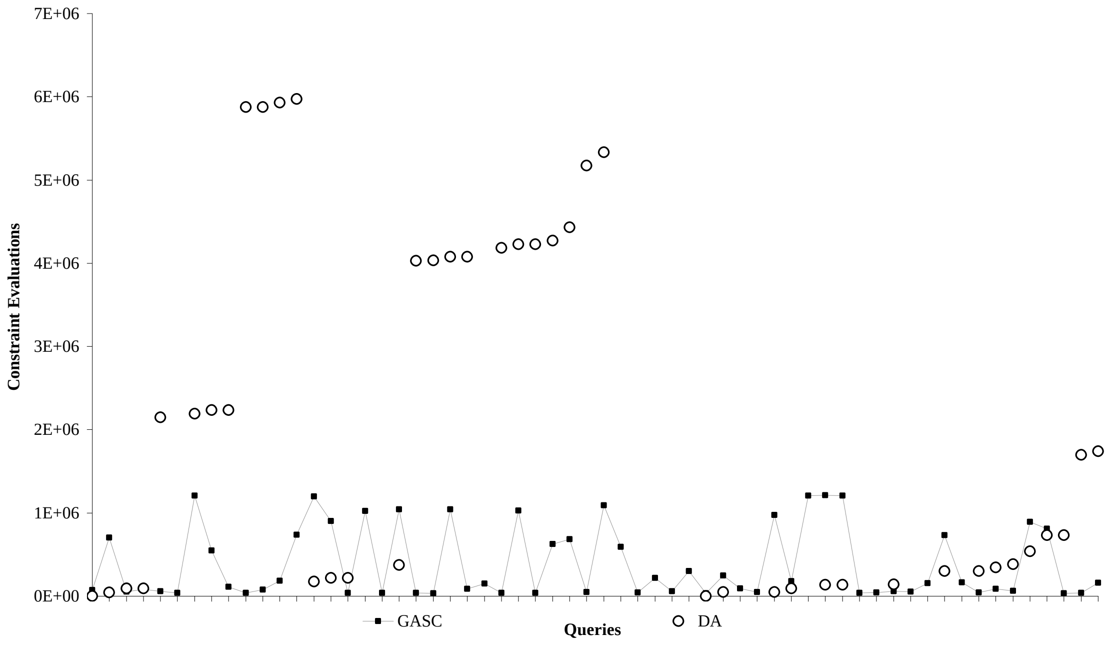


Figure 20: Number of constraint evaluations for queries with five objects in database Utility.

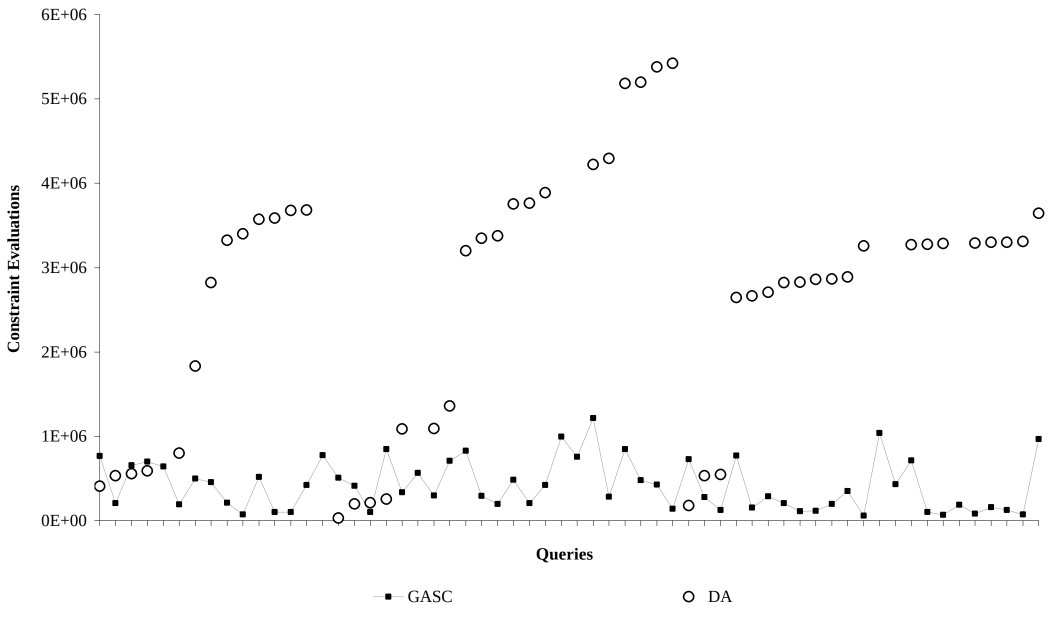


Figure 21: Number of constraint evaluations for queries with five objects in database Cell_Box.

Topological Relation	**Value Range**
disjoint	$F_m(a,b) > 1 \wedge F_m(b,a) > 1$
meet	$F_m(a,b) = 1 \wedge F_m(b,a) = 1$
overlap	$F_m(a,b) < 1 \wedge F_m(b,a) < 1$
equal	$F_m(a,b) = -1 \wedge F_m(b,a) = -1$
covers	$\lvert F_m(a,b)\rvert < 1 \wedge F_m(b,a) = -1$
covered_by	$F_m(a,b) = -1 \wedge \lvert F_m(b,a)\rvert < 1$
inside	$F_m(a,b) < -1 \wedge \lvert F_m(b,a)\rvert < 1$
contains	$\lvert F_m(a,b)\rvert < 1 \wedge F_m(b,a) < -1$

Table 1: Values of the content measure for the different topological relations between regions.

Model	Steady state
Representation	Specific
Fitness function	Penalty with violated constraints; with w_i defined by type of constraint
Recombination	Asexual heuristic operator
Mutation	Random
Parent selection	Lineal ranking
Survivor selection	Replace worst
Constraint handling	Mixed direct-indirect
Fitness adjustment	None
Use of heuristics	ad-hoc operator
Extra	Indexed domain of variables by content measure of topological relations

Table 2: Main features of the proposed genetic algorithm.

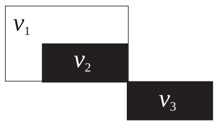
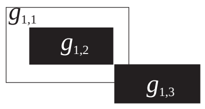
Configuration	Constraints	Constraint violations per object
Query:		
	$F(v_1, v_2) = 0.33$ $F(v_2, v_1) = -1.0$ $F(v_1, v_3) = 1.0$ $F(v_3, v_1) = 1.0$ $F(v_2, v_3) = 1.0$ $F(v_3, v_2) = 1.3$	
Candidate Solution a_k:		
	$F(g_{k,1}, g_{k,2}) = 0.19$ $F(g_{k,2}, g_{k,1}) = -1.22$ $F(g_{k,1}, g_{k,3}) = 1.0$ $F(g_{k,3}, g_{k,1}) = 1.0$ $F(g_{k,2}, g_{k,3}) = 1.3$ $F(g_{k,3}, g_{k,2}) = 1.3$	$g_{k,2} : 2$ $g_{k,1} : 1$ $g_{k,3} : 1$

Table 3: Selecting alleles in the *asexual_reproduction* operator.

Database	**Object type**	**#Objects**	**MBR's size**	**Index size**
Utility (Germany)	polylines	17790	0.2 MB	5 MB
Cell_box	rectangules	285	0.1 MB	1.3 MB

Table 4: Number of objects in databases.

Database	*disjoint*	*meet*	*overlap*	*coveredBy*	*covers*	*inside*	*contains*
Utility	316,939	21,416	1,222	386	386	77	77
Cell_box	38,808	13,200	14,748	5,016	5,016	2,058	2,058

Table 5: Ocurrence number of topological relations in databases.

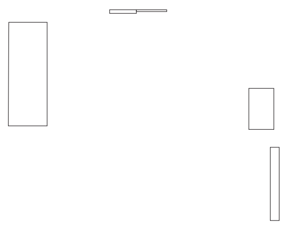
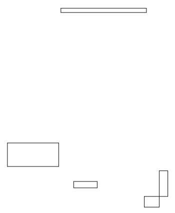
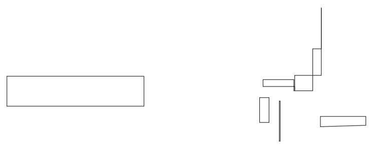
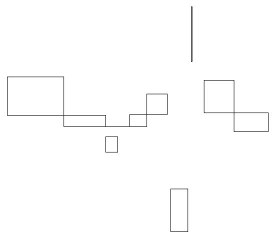
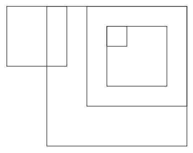
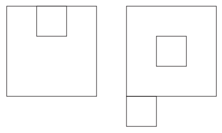
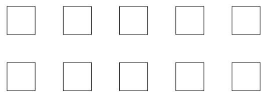
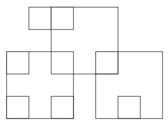
Database	**Objects**	**Query 1**	**Query 2**
Utility	5		
	10		
Cell_box	5		
	10		

Table 6: Queries for parameter setting.

	DA	**LS**	**GASC**
Utility			
5	73	0	93
10	30	0	92
Cell_box			
5	92	0	98
10	8	0	88

Table 7: Percentages of optimal solutions of each algorithm.