Constructive Genetic Algorithm for Clustering Problems

Luiz Antonio Nogueira Lorena
LAC-Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas 1758 - Caixa Postal 515
12201-970 São José dos Campos-SP Brazil
lorena@lac.inpe.br

João Carlos Furtado
Universidade de Santa Cruz do Sul
Av. Independência 2293
96815-900 Santa Cruz do Sul Brazil
jcarlosf@dinf.unisc.br

Abstract

Genetic Algorithms (GAs) have been accepted in recent years as powerful approaches to solve *optimization* problems. It is also well accepted that *building blocks* construction (schemata formation and conservation) has direct influence for a good behavior in *GA*. Schemata are usually indirectly evaluated, through a derived structure. We introduce in this work a new approach, called *Constructive Genetic Algorithm (CGA)*, which allows for schemata evaluation and for the provision of other new features to *GA*. Problems are modeled as *bi-objective* optimization problems, which consider the evaluation of two fitness functions. This double fitness process, called *fg-fitness*, evaluates schemata and structures in a common basis. The evolution process is conducted considering an adaptive rejection threshold, which contemplates both objectives and attributes a rank to each individual in population. The population is dynamic in size, composed of schemata and structures. Recombination preserves good schemata, and mutation is applied to structures to get population diversification. The *CGA* is applied to two clustering problems in graphs. Representation of schemata and structures use a binary digit alphabet, and are based on assignment (greedy) heuristics that provide a clearly distinguished representation for the problems. The clustering problems studied are the classical p-median and the capacitated p-median. Good results are shown for problem instances taken from the literatue.

Keywords

Genetic algorithms, clustering problems, p-median problems, capacitated p-median problem.

1 Introduction

Genetic Algorithms (GAs) have been recognized in recent years as powerful approaches to solve *optimization* problems (Bäck and Schwefel, 1993; Davis,1991; De Jong,1975; Goldberg,1989; Holland,1975; Lorena and Lopes, 1996; Lorena and Lopes, 1997; Michalewicz, 1996; Mitchell, 1996).The underlying foundations for such algorithms are the controlled evolution of a structured population.

The *GA* works on a set of variables called *structures*. For applying them to optimization problems, the first step is the definition of a coding scheme that allows a one-to-one mapping between solutions and structures. The following string can represent a structure $s_k = (s_{k1}, s_{k2}, ..., s_{kn})$, where n is the number of variables in the problem. A *fitness function* assigns a numeric value to each member of the current *population* (a collection of structures). The *genetic operators* used are *selection* (like tournament or biased roulette wheel) working together with a number of *crossover* and

mutation operators. The best structure is kept after a predefined number of generations (Goldberg, 1989; Holland, 1975; Michalewicz, 1996).

Holland (Holland,1975) put forward the *"building block"* hypothesis (*schema* formation and conservation) as a theoretical basis for the *GA* mechanism. In his view, avoiding disruption of good schema is the basis for the good behavior of a *GA*. A major problem with "building blocks", however, is that schemata are evaluated indirectly, via evaluation of their instances (structures). Goldberg and collaborators (Golberg et al.,1989, 1993; Kargupta,1995) have introduced an alternative *GA*, the *messy-GA*, that allows variable length strings that looks for the construction and preservation of good *"building blocks"*.

The *Constructive Genetic Algorithm (CGA)* is proposed here as an alternative to the traditional *GA* approach (Holland,1975), particularly in that *CGA* directly evaluates schemata. The population, initially formed only by schemata, is built, generation after generation, by directly searching for a population of not only well adapted structures, but also for good schemata.

Some steps in *CGA* are notably different from a classical *GA*. The *CGA* works with a *dynamic population*, initially composed of schemata, which is enlarged after the use of *recombination operators*, or made smaller along the generations, guided by an *evolution parameter*. Schemata recombination diversifies the population thereby generating new schemata or structures. At the time of its creation each schema or structure receives a rank used in the evolution analysis. Structures represent feasible solutions, undergo mutation and are compared to the best solution found so far, which is always retained. Another main difference between a classical *GA* and a *CGA* is the new *fg-fitness* process.

The *CGA* will be explained in detail in *sections 2* and *3*. We have adopted the principle of explaining the method with examples, based on the *clustering applications*.

Clustering problems generally appear in classification of data for some purpose like storage and retrieval or data analysis. Any clustering algorithm will attempt to determine some inherent or natural grouping in the data, using "distance" or "similarity" measures between individual data (Spath, 1980; Zupan, 1982). In this paper we examine the *CGA* application to two clustering problems in graphs, namely, the classical *p-median problem (PMP)* and the *capacitated p-median problem (CPMP)*.

The *PMP* is a classical location problem. The objective is to locate p facilities (medians) so as to minimize the sum of the distances from each demand vertex to its nearest facility (Hakimi,1964; Hakimi,1965). The problem is well known to be NP-hard (Garey and Johnson, 1979), and several heuristics have been developed for p-median problems (Densham and Rushton, 1992; Goodchild and Noronha, 1983; Rolland et al.,1997; Rosing and ReVelle, 1997; Rosing et al.,1998; Teitz and Bart, 1968). More complete approaches explore a *search tree* (Beasley, 1993; Christofides and Beasley, 1982; Efroymson and Ray, 1966; Galvão and Raggi, 1989; Jarvinen et al.,1972; Neebe, 1978). Other approaches consider *Lagrangean relaxation* and subgradient optimization in a primal-dual viewpoint (Beasley, 1993; Senne and Lorena, 2000).

The *CPMP* considers capacities for the service to be given by each median. The total service demanded by vertices identified by p-median clusters can not exceed the service capacity. Apparently, the *CPMP* was not so intensively studied as the classical *PMP*. Similar problems appeared in the works (Klein and Aronson, 1991; Maniezzo et al.,1998; Mulvey and Beck, 1984; Osman and Christofides, 1994).

This paper is organized as follows. The *CGA* description in divided in two sections. In section 2 we present aspects of modeling to be considered when solving a problem

using *CGA*. Modeling involves definitions of the schema and structure representations and the consideration of the problems at issue as *bi-objective* optimization problems. The evolution process is also described in this section. Section 3 describes the *CGA* operators, namely, selection, recombination and mutation, as well as the definition of an initial population and a *CGA* pseudo-code. Section 4 shows computational results using instances from the literature. We conclude in section 5 with a summary of the *CGA* performance and characteristics.

2 CGA modeling

In this section is described the modeling phase of the *CGA*. The clustering problems are formulated as bi-objective optimization problems. Two fitness functions are defined on the space of all schemata and structures that can be obtained using a specific representation. The evolution process considers the two objectives on an adaptive rejection threshold, which gives ranks to individuals in population and yields a dynamic population.

2.1 Structure and schema representation

Very simple structure and schema representations are adopted for problems *PMP* and *CPMP*. They use a binary alphabet, and assignment heuristics make a clear and independent connection with the two clustering problems. The use of the same kind of representation allows the remaining steps in *CGA* to be valid for both problems.

Suppose a given graph *G=(V,E)*. A clustering problem in graphs can be stated as the search for partitions on the vertex set V in a (generally) predefined number of clusters, optimizing some measure on combinations of vertices and/or edge weights. The problems considered in this paper are clustering problems in graphs.

A typical instance of the problems is composed of n demand points (vertices) $V = \{1,...,n\}$, and a distance (weight) matrix $[\mu_{jl}]$, indicating distances between pairs of vertices, such that $\mu_{jl} \geq 0$, $\mu_{jj} = 0$ and $\mu_{jl} = \mu_{lj}$ for all $j, l \in V$.

To define the representation some vertices are elected as the *seeds*, i. e., the initial vertices in clusters that attract, in some way, the other vertices that participate in the representation.

Starting with an example for the *PMP*, consider the following 3-median solution for a complete graph *G(V,E)* instance with 10 vertices (see *figure 1*).

A partition on the index vertex set V is then made, yielding two blocks, the *seed* set and the *non-seed* set, where the seeds are the *medians*. For the *3-median solution* of *figure 1*, $V_1(s_k) = \{1, 9, 10\}$ is the seed set, and $V_0(s_k) = \{2, 3, 4, 5, 6, 7, 8\}$ is the non-seed set. Vertices 1, 9 and 10 are the medians, and the others are assigned to a median. The *3-median structure* for the example will be $s_k = (1, 0, 0, 0, 0, 0, 0, 0, 1, 1)$, where each position s_{kj} in s_k, receiving *labels* 1 or 0, means that vertex j belongs to sets $V_1(s_k)$ or $V_0(s_k)$, respectively.

Structure s_k is not completely defined, as we do not know the non-median assignments. For the clustering problems, after the initial seed identification, an *Assignment Heuristic* is employed to assign non-seed vertices to clusters. For the *PMP* each non-median vertex is assigned to the nearest identified median. Algorithm *AH1*, formalizes the assignments.

AH1
Read s_k,

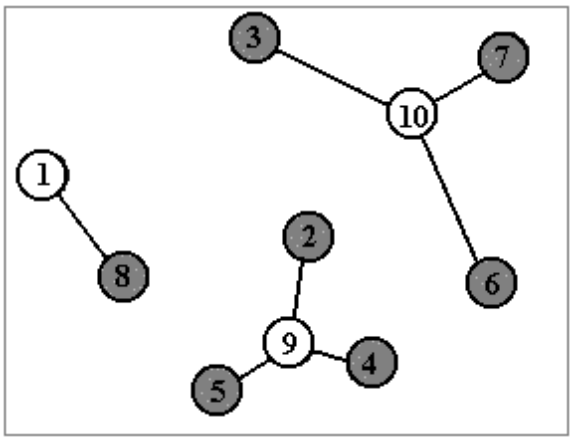

Figure 1: A 3-median solution

$V_1(s_k) = \{\zeta_1, \zeta_2, ..., \zeta_{|V_1(s_k)|}\}$,
$V_0(s_k) = \{\beta_1, \beta_2, ..., \beta_{|V_0(s_k)|}\}$;
For $i = 1$ to $|V_1(s_k)|$ do
$C_i(s_k) := \{\zeta_i\}$;
end_for
For $j = 1$ to $|V_0(s_k)|$ do
$r := index\left[Min_{\{i=1,...,|V_1(s_k)|\}}\{\mu_{\zeta_i\beta_j}\}\right]$;
$C_r(s_k) := C_r(s_k) \cup \{\beta_j\}$;
end_for

After the application of *AH1*, exactly $p = |V_1(s_k)|$ clusters are identified (for the example in *figure 1*, $C_1(s_k) = \{1, 8\}, C_2(s_k) = \{2, 4, 5, 9\}$,and $C_3(s_k) = \{3, 6, 7, 10\}$), corresponding to the median set $V_1(s_k) = \{1, 9, 10\}$.

The *CGA* works directly with *schemata*. A *3-median schema* for the example can be $s_k = (1, \#, 0, 0, 0, \#, 0, 0, 1, 1)$, as *figure 2* clarifies. The defined sets are: $V_1(s_k) = \{1, 9, 10\}$, $V_0(s_k) = \{3, 4, 5, 7, 8\}$ and the new set $V_\#(s_k) = \{2, 6\}$. $V_\#(s_k) = V - (V_1(s_k) \cup V_0(s_k))$ is formed by vertices not considered by the 3-median schema. The "do not care" label # will be used herein to distinguish this condition. Observe that the number of medians is the same on schemata and structures, determining the name *p-median schema* (a condition that can be relaxed).

The same heuristic *AH1* is then used here to make the assignments, giving a clearly and unique representation. The clusters identified in *figure 2* are $C_1(s_k) = \{1, 8\}$, $C_2(s_k) = \{4, 5, 9\}$, and $C_3(s_k) = \{3, 7, 10\}$.

The structure and schema s_k defined above can also be used on the *CPMP* representation. The only modification is that now the clusters have capacities. Heuristic *AH2* is the corresponding assignment heuristic in this case. Assume, for instance, that the cluster capacities are the same (a condition that can also be relaxed).

Each non-median vertex is assigned to their nearest median if the cluster capacity

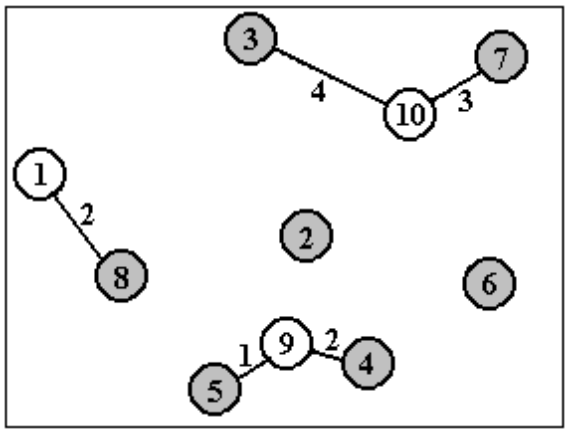

Figure 2: A 3-median solution

is not violated.

AH2
Read s_k,
Q,
$V_1(s_k) = \{\zeta_1, \zeta_2, ..., \zeta_{|V_1(s_k)|}\}$,
$V_0(s_k) = \{\beta_1, \beta_2, ..., \beta_{|V_0(s_k)|}\}$;
$\alpha_{\beta_j}, j = 1, ..., |V_0(s_k)|$,
$\alpha_{\zeta_i}, i = 1, ..., |V_1(s_k)|$,
For $i = 1$ to $|V_1(s_k)|$ do
$Q_i := Q - \alpha_{\zeta_i}$,
$C_i(s_k) := \{\zeta_i\}$;
end_for
For $j = 1$ to $|V_0(s_k)|$ do
$r := index\left[Min_{\{i=1,...,|V_1(s_k)|\}}\{\mu_{\zeta_i\beta_j}|Q_i - \alpha_{\beta_j}| \geq 0\}\right]$;
$Q_r := Q_r - \alpha_{\beta_j}$,
$C_r(s_k) := C_r(s_k) \cup \{\beta_j\}$;
end_for

Define X as the set of all structures and schemata that can be generated by the $0 - 1 - \#$ string representation. For the *PMP*, the assignment heuristic *AH1* allows completeness to X, in the sense that an optimal solution to the problem is always in X. The same is not true on heuristic *AH2* for *CPMP* (the capacity constraints may not be correctly represented).

The assignment heuristics can be defined in various ways, and more elaborated they are, probably better solutions they found to the clustering problems (although, generally increasing computational times).

2.2 The bi-objective optimization problem

The *CGA* is proposed to address the problem of evaluating schemata and structures in a common basis. While in the other evolutionary algorithms, evaluations of individuals are based on a single function (the fitness function), in *CGA* this process relies on two functions, mapping the space of structures and schemata onto $\Re_+$.

Let$X$ be the set of all structures and schemata that can be generated by the $0-1-\#$ string representation of *section 2.1.*, and consider two functions f and g, defined as $f: X \rightarrow \Re_+$ and $g: X \rightarrow \Re_+$ such that $f(s_k) \leq g(s_k)$, for all $s_k \in X$. We define the double fitness evaluation of a structure or schema s_k, due to functions f and g, as *fg-fitness*.

The *CGA* optimization problem implements the *fg-fitness* with the following two objectives:

1. (*interval minimization*) Search for $s_k \in X$ of minimal $\{g(s_k) - f(s_k)\}$.

   This objective can be accomplished by schemata or structures. The response to the evolutionary search is given on a very adapted structure, then, a further optimization objective is needed to guide the search to find structures. This second objective is then:

2. (g *maximization*) Search for $s_k \in X$ of maximal $g(s_k)$.

   Considering the schema representation, the *fg-fitness* evaluation increases as the number of labels # decreases, and therefore structures have higher *fg-fitness* evaluation than schemata.

To attain these purposes, a problem to be solved using *CGA* is modeled as the following *Bi-objective Optimization Problem (BOP)*:

$$Min\{g(s_k) - f(s_k)\}$$
$$Max\ g(s_k)$$
$$subject\ to\ g(s_k) \geq f(s_k), s_k \in X$$

Functions f and g must be properly identified to represent the optimization objectives of the problems at issue. The *fg-fitness* process is particularized in the following for the clustering problems.

Consider a structure or schema $s_k \in X$. For *PMP* and *CPMP*, after the application of the assignment heuristics *AH1* or *AH2*, $p = |V_1(s_k)|$ clusters $C_i(s_k)$ are identified, corresponding to the median set $V_1(s_k) = \{\zeta_1, \zeta_2, ..., \zeta_p\}$.

Function g is defined by $g(s_k) = \sum_{i=1}^{p} \sum_{j \in C_i(s_k)} \mu_{\zeta_i j}$, and function f is defined by $f(s_k) = \sum_{i=1}^{p} \lambda_i.[|C_i(s_k)| - 1]$, where $\lambda_i = Min_{j \in C_i(s_k)}\{\mu_{\zeta_i j}\}$ is the minimum cost *(distance)* assigned in cluster i, and $|C_i(s_k)|$ is the cardinality of the set $C_i(s_k)$.

For the schema (1,#,0,0,0,#,0,0,1,1) represented in *figure 2*, we have $f(1,\#,0,0,0,\#,0,0,1,1) = 2*1 + 1*2 + 3*2 = 10$, and $g(1,\#,0,0,0,\#,0,0,1,1) = 2+1+2+3+4 = 12$.

Clearly $f(s_k) \leq (s_k)$, for all $s_k \in X$. The objective in *BOP* of minimizing the interval $g(s_k) - f(s_k)$ is directly related to the optimizing objectives on *PMP* and *CPMP*. The evaluation of function f gives an "ideal" evaluation for the distances of assigned vertices in clusters, while function g gives the actual evaluation for the distances of the assigned vertices. If s_k is a structure, $g(s_k)$ gives the solution value for the respective problem, and the best (*minimum*) value is retained in the process.

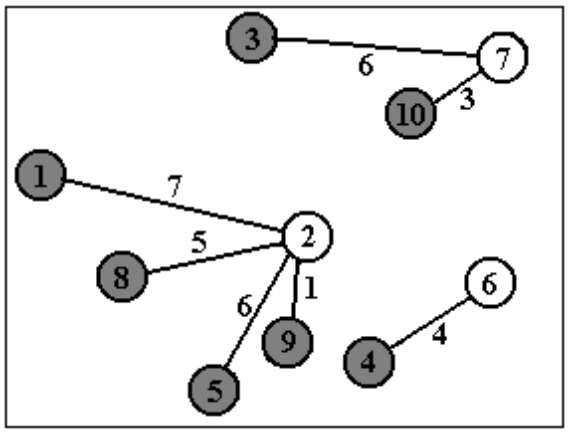

Figure 3: A random structure (0,1,0,0,0,1,1,0,0,0)

2.3 The evolution process

The evolution process in *CGA* is conducted to accomplish the objectives (*interval minimization and g maximization*) of the *BOP*. At the beginning of the process, the following two *expected values* are given to these objectives:

1. (*expected value to g maximization*)

   A non-negative real number $g_{max} > Max_{s_k \in X} g(s_k)$, that is an upper bound to $g(s_k)$, for each $s_k \in X$.

2. (*expected value to interval minimization*)

   The interval length dg_{max}, obtained from g_{max} using a real number $0 \leq d \leq 1$.

For the clustering problems, the overall bound g_{max} is obtained at the beginning of the *CGA*, by generating a random complete structure and making g_{max} receive the g evaluation for that structure.

For the 3-median example of *section*2.1., the random structure of *figure 3* gives $g_{max} = g(0,1,0,0,0,1,1,0,0,0) = 32$, and for $d = 0.1$, the expected interval length is $dg_{max} = (0.1).(32) = 3.2$. In order to ensure that g_{max} is always an upper bound, after recombination, each new structure generated s_{new} is rejected if $g_{max} \leq g(s_{new})$.

The evolution process is then conducted considering an adaptive rejection threshold, which contemplates both objectives in *BOP*. Given a parameter $\alpha \geq 0$,the expression

$$g(s_k) - f(s_k) \geq dg_{max} - \alpha.d[g_{max} - g(s_k)] \qquad 2.3.1$$

presents a condition for rejection from the current population of a schema or structure s_k.

The right hand side of (2.3.1) is the threshold, composed of the expected value to the interval minimization dg_{max}, and the measure $[g_{max} - g(s_k)]$, that shows the difference of $g(s_k)$ and g_{max} evaluations.

For $\alpha = 0$, (2.3.1) is equivalent to comparing the interval length obtained by s_k and the expected length dg_{max}. Schemata or structures are discarded if expression

(2.3.1) is satisfied. When $\alpha > 0$, schemata are penalized and have higher possibility of being discarded than structures, as structures present, in general, smaller differences $[g_{max} - g(s_k)]$ than schemata.

Parameter α is related to time in the evolution process. Considering that the good schemata need to be preserved for recombination, the *evolution parameter* α starts from 0, and then increases slowly, in small time intervals, from generation to generation. The population at the evolution time α, denoted by P_α, is dynamic in size accordingly the value of the adaptive parameter α, and can be emptied during the process.

The parameter α is now isolated in expression (2.3.1), thus yielding the following expression and corresponding rank to s_k:

$$\alpha \geq \frac{dg_{max} - [g(s_k) - f(s_k)]}{d[g_{max} - g(s_k)]} \qquad 2.3.2$$

1. (*rank*)

   The right hand side of expression (2.3.2) gives a *rank* value to s_k:

   $$\delta(s_k) = \frac{dg_{max} - [g(s_k) - f(s_k)]}{d[g_{max} - g(s_k)]}$$

At the time they are created, structures and/or schemata receive their corresponding rank value $\delta(s_k)$. The *rank* of each schema or structure is compared with the current evolution parameter α. At the moment a structure or schema is created, it is then possible to have some figure of its survivability. The higher the value of $\delta(s_k)$, and better is the structure or schema to the *BOP*, and they also have more surviving and recombination time.

Analyzing the ratio $\delta(s_k)$ it is possible to isolate the effects of d and g_{max} in population size. It is clear that, for small d, the population presents slow increase and maintains of small dimension. A large d may give the storage problems of a large population, but eventually with good structures. An undesired effect of g_{max} is that, when $g_{max} \gg g(s_k)$ for a large number of $s_k \in X$, they all have $\delta(s_k) \cong 1$, thus entailing the possibility of being eliminated, from a generation to the next (after $\alpha = 1$). Naturally, the effect can be an enormous reduction in population size in a few generations. Then the definition of parameters d and g_{max} must be the result of a careful study.

3 The CGA operators

In this section is described the operators that works on the evolution process in the *CGA*. The *CGA* initial population will be formed only by schemata. Recombination will create new schemata and structures.

Recombination can produce more adapted structures and/or schemata. The best structure needs to be kept in the process, generating a best feasible solution to the problem under consideration. Structures can be improved by mutation. After some generations the population is composed of good schemata and probably good structures representing improved feasible solutions.

3.1 Initial population

The initial population is composed exclusively of schemata, and for a sequence of generations, the population can increase by addition of new offspring generated out of the combination of two schemata.

Let P_0 be the initial population. For each schema, a proportion of random positions receives label 0, and p (number of clusters) random positions receive label 1. The remaining positions receive label #.

The schemata in P_0 will be (re)combined to produce offspring with some of the labels # substituted by labels 1 or 0 , seeking for structures. For the case of # to 1 substitution, some original position with label 1 are modified to 0, maintaining the pre-fixed number of clusters.

The proportion of labels # have direct influence on the evolution process. Schemata with a small number of labels # have higher possibility of generating structures as offspring than those with large number of labels #, but it is difficult to have good schemata generated at random. On the other side, schemata with large number of labels # have small possibility of generating a structure as an offspring.

For the computational tests of section four, 20% of n random positions received label 0, where n is the size of P_0. The following can be an initial population for the 3-median example ($n = 10$):

$s_1 = (1, 0, \#, 1, 1, \#, 0, \#, \#, \#)$, $s_2 = (\#, 1, 0, 0, \#, 1, \#, \#, \#, 1)$,
$s_3 = (\#, \#, 1, 0, \#, 1, 1, \#, \#, 0)$, $s_4 = (0, \#, \#, 1, 0, \#, \#, 1, \#, 1)$,
$s_5 = (1, \#, 0, \#, \#, 1, 0, 1, \#, \#)$, $s_6 = (\#, 0, \#, \#, 1, 1, 1, 0, \#, \#)$,
$s_7 = (\#, \#, \#, 0, 1, 0, 1, \#, \#, 1)$, $s_8 = (\#, 1, 0, 0, \#, \#, 1, 1, \#, \#)$,
$s_9 = (0, 0, \#, \#, \#, 1, \#, 1, 1, \#)$, $s_{10} = (0, 1, 0, 1, \#, \#, \#, 1, \#, \#)$.

3.2 Recombination

We have two purposes on the evolution process: to obtain structures (good solutions to the *g maximization* objective on the *BOP*), and that these structures be good ones (best solutions to the *interval minimization* problem on the *BOP*). The selection of structures and/or schemata for recombination will be conducted to attain these two objectives. The first one is attained by selecting for recombination schemata with small number of labels #, and the second considering structures or schemata with small $d_k = \frac{g(s_k) - f(s_k)}{g(s_k)}$.

The structures and schemata in population P_α are maintained on ascending order, according to the key $\Delta(s_k) = \frac{1+d_k}{|V_1(s_k)|+|V_0(s_k)|}$. Structures ($V_\#(s_k) = \emptyset$), schemata with $|V_\#(s_k)|$ small, and structures and/or schemata presenting small d_k are better and appear in the initial positions.

Two structures and/or schemata are selected for combination. The first is called the *base* (s_{base}) and is randomly selected out of the n first positions in P_α; in general it is a good structure or a good schema (d_{base} is small). If it turns out to be a schema, the following recombination process tries to preserve labels 1 or 0 already assigned in s_{base}. The second structure or schema is called the *guide* (s_{guide}) and is randomly selected out of the total population. The objective of the s_{guide} selection is the conduction of a guided modification on s_{base}. The current labels in corresponding positions are compared. Let s_{new} be the new structure or schema (offspring) after recombination.

Structure or schema s_{new} is obtained by applying the following operations (in this order):

{Recombination}
(i) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = \#$ and $s_{guidej} = \#$ set $s_{newj} = \#$;
(ii) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = 1$ and $s_{guidej} = 1$ set $s_{newj} = 1$;
(iii) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = 0$ and $s_{guidej} = 0$ set $s_{newj} = 0$;
(iv) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = 1$ and $s_{guidej} = \#$ set $s_{newj} = 1$;

(v) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = 0$ and $s_{guidej} = \#$ set $s_{newj} = 0$;
(vi) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = \#$ and $s_{guidej} = 0$ set $s_{newj} = 0$;
(vii) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = \#$ or 0 and $s_{guidej} = 1$ then two cases are possible:

1. set $s_{newj} = 1$, also if $|V_1(s_{new})| = p$, set $s_{newl} = 0$ for $l \in \{1, ..., n\}$ presenting $s_{newl} = 1$ (randomly selected),

2. set $s_{newj} = 1$, also if $|V_1(s_{new})| = p$, set $s_{newl} = 0$ for each $l \in \{1, ..., n\}$ presenting $s_{newl} = 1$ generating p new structures and/or schemata,

(viii) For each $j \in \{1, ..., n\}$ presenting $s_{basej} = 1$ and $s_{guidej} = 0$ then two cases are possible:

1. set $s_{newj} = 0$ and $s_{newl} = 1$ for $l \in \{1, ..., n\}$ presenting $s_{newl} = 0$ (randomly selected),

2. set $s_{newj} = 0$ and $s_{newl} = 1$ for each $l \in \{1, ..., n\}$ presenting $s_{newl} = 0$ generating $|V_0(s_{base})|$ new structures and/or schemata.

For each offspring generated, operations (i)-(v) preserve in s_{new} the labels which are present in s_{base} (schema information). Operations (vi)-(viii) produce different offspring. Operation (vi) produces one s_{new} that is s_{base} with some additional labels 0. Operations (vii) and (viii) can be seen as a mutation operator generating, in the second case, p or $|V_0(s_{base})|$ new offspring. These mutation like phases on recombination (operations (vii) and (viii)) are imposed by the number of clusters in s_{new} (the number of 1's is fixed on schemata and structures).

When s_{base} is a structure ($V_{\#}(s_{base}) = \emptyset$), the following interchange heuristic is performed as a local search mutation:

```
IH {Interchange Heuristic}
For each j ∈ V1(s_base) do
    For each l ∈ V0(s_base) do
        Interchange j and l and generating an offspring s_new;
            {offspring generation}
        Interchange l and j;
    End_for
End_for
```

The application of the local search mutation *IH* produces $p.|V_0(s_{base})|$ new structures. For each α value, n new structures and/or schemata are generated, their ranks computed and compared with α, and included or not in the new population.

3.3 The algorithm

The Constructive Genetic Algorithm can be summed up by the pseudo-code:

```
CGA {Constructive Genetic Algorithm }
Given g_max and d;
```

```
α:=0;
ε := 0.05;                                   { time interval }
Initialize P_α;                              { initial population }
Evaluate P_α;                                { fg-fitness }
For all s_k ∈ P_α compute δ(s_k)             { rank computation }
end_for
While (not stop condition) do
    For all s_k ∈ P_α satisfying α < δ(s_k) do   { evolution test }
        α := α + ε;
        Select P_α from P_{α−ε};             { reproduction operator }
        Recombine P_α;                       { recombination operators }
        Evaluate P_α;                        { fg-fitness }
    end_for
    For all new s_k ∈ P_α compute δ(s_k)     { rank computation }
    end_for
end_while
```

The *CGA* algorithm begins with the recombination procedures (schemata) and no local search mutation. After a sequence of generations, the number of structures increases and so does the consequent application of *IH* mutations. At final generations only *IH* mutations are performed.

Our stop conditions occur with an emptied population (assured by a sufficiently higher α) or at a predefined number of generations. The population increases, after the initial generations, reaching an upper limit (in general controlled by storage conditions), and decreases for higher values of the evolution parameter α (see *figure 4* in *section 4*.). The structure corresponding to the best problem solution must be kept in the process.

4 Computational results

The *CGA* was initially tested using a partial sample of the p-median data drawn from the OR-library (Beasley,1990). The sizes are 100, 200, 300, 400, and 500 vertices. The number of medians varies from 5 to 67.

The computational tests are reported in *table 1*. The *Lagrangean* heuristic results appeared in the paper (Senne and Lorena, 2000); it uses local search and an iterative location-allocation procedure to obtain feasible solutions. It was programmed in C, running on an IBM Risc/6000 model 3AT workstation (compiled using xlc compiler with -O2 optimization option). The *CGA* feasible solution is the $g(s_k)$ evaluation for the best structure kept, using the simple *AH1* heuristic for the assignments. The *gap* is calculated as [(Feasible Solution - Optimal solution)*100]/(Feasible Solution).

The *CGA* results are slightly worse than the Lagrangean ones, but all gaps are inferior to 0.73% and null for nine instances. For all the *CGA* results presented in this section, the algorithm was coded in C, running on a Pentium 166 Mhz. The computational times (*table 1*) for both algorithms are not comparable due to the use of different machines, although the IBM Risc/6000 could be considered faster than a Pentium 166 Mhz.

It can be seen in *table 1* that a large number of medians increase the *CGA* computational times. This is easy to explain as the local search mutation is used more

intensively on these cases. For all *CGA* results presented in this section $d = 0.1$, and the evolution control (time interval) used is: $\epsilon = 0.05$ for $0 \leq \alpha \leq 1$ and $\epsilon = 0.025$ for $\alpha > 1$.

We examine in the following graphical representations of the running test with problem *pmed1*. *Figure 4* shows the population evolution by generation. A maximum population size is obtained at generation 25 ($P_{1.25}$) with approximately 2000 structures and/or schemata. The population decreases after $\alpha = 1.25$, showing that the number of new structures and/or schemata by generation is smaller than the number that do not pass the evolution test. The procedure stopped with an empty population at $\alpha = 4.5$.

Table 1: Computational results (OR-library instances)

Problem	Vertices (n)	Medians (p)	Optimal Solution	Lagrangean heuristic gap(%)	Lagrangean heuristic times(sec.)	CGA gap (%)	CGA times (sec.)
pmed1	100	5	5819	0	0.93	0	28
pmed2	100	10	4093	0	1.34	0	37
pmed3	100	10	4250	0	1.79	0	34
pmed4	100	20	3034	0	1.58	0	230
pmed5	100	33	1355	0	2.07	0.36	375
pmed6	200	5	7824	0	4.75	0	172
pmed7	200	10	5631	0	5.45	0	238
pmed8	200	20	4445	0	5.03	0.20	1055
pmed9	200	40	2734	0	10.6	0.73	3331
pmed10	200	67	1255	0	17.4	0.15	4325
pmed11	300	5	7696	0	10.0	0	369
pmed12	300	10	6634	0	11.7	0.04	677
pmed13	400	5	8162	0.012	19.11	0	555
pmed14	500	5	9138	0	20.60	0	1875

Figure 5 shows the maximum number of vertices for each structure or schema generated out of recombination and local search mutation. Only after generation 13 a structure is obtained. Even with a population mostly with structures, schemata form a representative part of it; after all, a schema may receive a better *fg-fitness* than a structure.

Figure 6 shows the improvements on the best d_k by generation and finally *figure 7* shows the best g function evaluation for a structure (*p-median solution*) at each generation.

A classical mathematical programming formulation for p-median problems was considered in the papers (Beasley, 1985,1993; Christofides and Beasley, 1982; Senne and Lorena, 2000). Considering a primal-dual approach, the set of OR-Library p-median instances can be considered that is composed of easy problems in the sense of duality gaps. The gaps can all be closed using a *Lagrangean* (or *Lagrangean/surrogate*) approach (Beasley, 1993; Senne and Lorena, 2000). Based on that, another test suit was

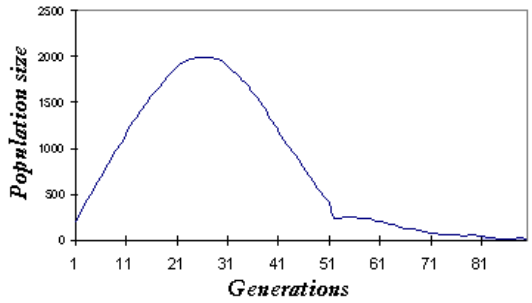

Figure 4: Population size by generation

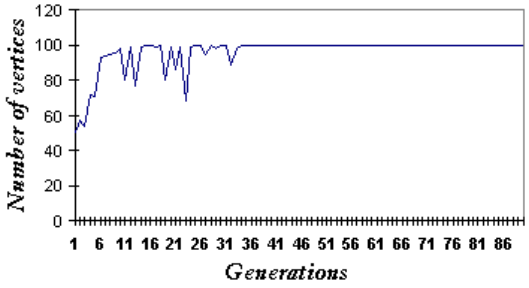

Figure 5: Maximum number of vertices by generation

performed, now with another set of instances (Galvão and ReVelle, 1996; Senne and Lorena, 2000), that presents large duality gaps for some values of p. A comparison was also made with the direct application of the interchange heuristic IH on a set of structures.

Table 2 shows the computational experiments. The mean *CGA* gaps for 5 replications are lower than 0.547% for all instances. The *CGA* results compare favorably with the Lagrangean ones. The Lagrangean heuristic's worst result is 3.595%, and, for five instances, the Lagrangean heuristic gap is greater than 0.547%. To apply the IH alone, we start with an initial set of n structures, then IH is applied on each structure. The process is replicated with 5 initial randomly generated sets, and the best solution reported in *table 2*. The direct application of IH yields gaps varying from 3.57% to 14.88%. The comparison is included to accentuate the concept that *CGA* works with a population that retains adapted structures after the constructive phase.

The *CGA* was also tested on instances of the *CPMP*. Two sets of 10 instances are

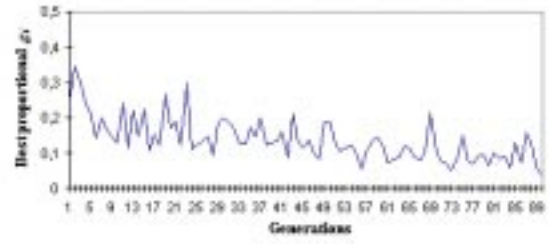

Figure 6: Best d_k by generation

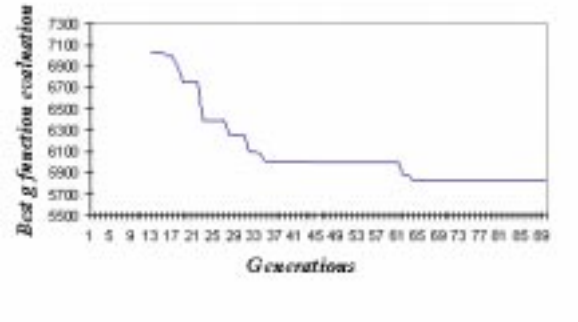

Figure 7: Best g evaluation by generation

considered, with (50x5) and (100x10) vertices and medians, respectively. They appeared in the paper (Osman and Christofides, 1994). The results are shown in *table 3*, which columns are composed of the problem identification, the best known solution, results for some heuristics (H.OC, H1+F1, H1+B1, HSS.OC, HSS.C and TS1+FBA) used in (Osman and Christofides, 1994), and at the last column, the *CGA* solution.

Heuristic H.OC is a simple constructive heuristic, while H1+F1 and H1+B1 begin with the H.OC solution and make some permutations using "first improve" and "best improve" strategies. Algorithm TS1+FBA is a *tabu search* implementation. *Simulated annealing* is implemented with algorithms HSS.OC and HSS.C, which use distinct cooling schedules. The best results are obtained using the algorithm HSS.OC, where the simulated annealing probabilistic acceptance was improved in three ways. It makes use of a non-monotonic cooling schedule, a systematic neighborhood search, and a termination condition based on the number of temperature resets performed without improving the best solution.

Table 4 presents the gaps to the best solution reached by the 7 heuristics, as well as the times for heuristic HSS.OC and the mean times for 5 replications of the *CGA*. The times for HSS.OC are reported in the paper (Maniezzo et al., 1998), where the authors have implemented the heuristic in Fortran 77, also running on a Pentium 166

Mhz. *Tables 3* and *4* show that the *CGA* results were as good as TS1+FBA, and little worse than HSS.OC and HSS.C (i. e., their best results are comparable).

Table 2: Problems presenting large duality gaps.

Number of vertices (n)	Number of medians (p)	Duality gap (%)	Optimal solution	Lagrangean heuristic gap(%)	IH gap (%)	CGA gap (%)
100	5	0.346	5703	0	6.36	0
100	10	3.728	4426	2.553	10.72	0
100	15	0.895	3893	0.745	8.20	0.136
100	20	0.093	3565	0.084	6.07	0.299
100	25	0.067	3291	0	4.10	0.243
100	30	0.056	3032	0.066	3.73	0.425
100	40	0	2542	0	3.57	0.432
100	50	0	2083	0	4.11	0.096
150	5	1.404	10839	0	4.24	0
150	10	3.158	8729	0.252	8.21	0
150	15	4.906	7390	0.731	12.15	0.184
150	20	2.975	6454	3.595	14.88	0.547
150	25	1.009	5875	2.060	13.20	0.204
150	30	0.208	5495	0.564	12.59	0.345
150	40	0.068	4907	0.143	8.40	0.346
150	50	0.062	4374	0	6.24	0.137

The *CGA* results can be considered good, spending small computer times. Considering effective implementations of Lagrangean relaxation heuristics and Simulated Annealing, the *CGA* results are comparable, and even better in some cases.

5 Conclusion

This work describes a constructive approach to genetic algorithms and an application to some clustering problems in graphs.

The *Constructive Genetic Algorithm - CGA* provides some new features to genetic algorithms. The principal *CGA* characteristics can be summarized as:

1. the direct work and evaluation of schemata,
2. the population is dynamic in size,
3. the initial population is formed by schemata,
4. the new *fg-fitness* provides double evaluation of schemata and structures,
5. at the creation time schemata and structures receive a rank,
6. good schemata and structures have high ranks and greater surviving and recombination times,
7. recombination preserves good schemata,

8. local search mutation to structures provides population diversification,

9. one objective of the CGA is the construction of a population with good structures after some generations.

For the clustering problems studied (*PMP* and *CPMP*) assignment heuristics decode common binary representations. The (good) effects of the heuristics are reflected to the whole population. The same *CGA* representation (composed with an assignment heuristic) was also tested (and described in an earlier version of this paper) on the *Min Cut Clustering Problem* considered in the paper of Johnson and Mehrotra (Johnson and Mehrotra,1992). It is the problem of partitioning the vertex set of a given graph into a pre-fixed number of clusters, such that the sum of the cluster vertex weights have inferior and superior limits, while the sum of clusters edge weights is maximized (or, alternatively, the sum of edge weights outside the clusters is minimized).

These representations and assignment decoders can be used in other clustering problems, but it is important to note that the *CGA* concepts and properties are independent on the representation and decoders used. Some early results used different representations to the *CGA* application to 2D-cutting (Lorena and Lopes, 1996) and k-coloring (Ribeiro and Lorena, 1997) problems.

Some complementary research can be made on experimental parameter design (ϵ and d), additional representation decoders (heuristics) and definitions of functions f and g and of parameter g_{max}. It is expected that the good behavior obtained with the clustering problems studied here can be repeated with other difficult optimization problems.

Table 3: Problem CPMP-CGA and Osman & Christofides results.

Problem	Vertices	Medians	Best solution	H.OC	H1+F1	H1+B1	HSS.OC	HSS.C	TS1+ FBA	CGA
1	50	5	713	786	780	818	713	734	734	713
2	50	5	740	816	762	778	740	740	740	740
3	50	5	751	972	811	816	751	751	751	751
4	50	5	651	891	651	652	651	651	651	651
5	50	5	664	804	746	677	664	664	664	664
6	50	5	778	882	841	847	778	778	778	778
7	50	5	787	968	852	824	787	805	787	787
8	50	5	820	945	834	837	820	820	821	826
9	50	5	715	752	735	734	715	715	715	715
10	50	5	829	1017	844	891	829	829	829	834
11	100	10	1006	1761	1020	1019	1006	1006	1009	1014
12	100	10	966	1567	1004	974	966	966	968	969
13	100	10	1026	1847	1144	1053	1026	1026	1026	1026
14	100	10	982	1635	998	1054	985	982	985	987
15	100	10	1091	1517	1098	1138	1091	1091	1096	1091
16	100	10	954	1780	1063	993	954	954	957	955
17	100	10	1034	1665	1104	1092	1039	1037	1040	1034
18	100	10	1043	1345	1089	1136	1045	1045	1045	1045
19	100	10	1031	1634	1105	1125	1031	1032	1034	1032
20	100	10	1005	1872	1036	1030	1005	1019	1005	1039

Acknowledgments:

The first author acknowledges Conselho Nacional de Desenvolvimento Cientfico e Tecnolgico -CNPq (proc. 350034/91-5, 520844/96-3, 680082/95-6) and Fundação para o Amparo a Pesquisa no Estado de S. Paulo - FAPESP (proc. 95/9522-0 e 96/04585-6) for partial financial support. We acknowledge the comments and suggestions of two anonymous referees, the suggestions for text improvement made by the editor Prof. D. Whitley and by Prof. P. P. B. Oliveira from UNIVAP-Universidade do Vale do Paraíba.

Table 4: Problem CPMP:CGA and Osman & Christofides gaps to the best solution.

Problem	H.OC	H1+FI	H1+BI	HSS.OC	HSS.C	TS1+FBA	HSS.OC times (sec.)	CGA	CGA times (sec.)
1	10.23	9.39	14.72	0	2.94	2.94	4.34	0	2
2	10.27	2.97	5.13	0	0	0	3.94	0	2
3	29.42	7.98	8.65	0	0	0	4.10	0	12
4	36.86	0	0.15	0	0	0	3.39	0	2
5	21.08	12.34	1.95	0	0	0	3.77	0	8
6	13.36	8.09	8.86	0	0	0	5.52	0	2
7	22.99	8.25	4.70	0	2.28	0	5.15	0	3
8	15.24	1.70	2.07	0	0	0.12	10.18	0.73	11
9	5.17	2.79	2.65	0	0	0	11.43	0	2
10	22.67	1.80	7.47	0	0	0	3.40	1.44	14
11	75.04	1.39	1.29	0	0	0.29	100.57	0.79	614
12	62.21	3.93	0.82	0	0	0.20	128.67	0.31	296
13	80.01	11.50	2.63	0	0	0	106.13	0	327
14	66.49	1.62	7.33	0.30	0	0.30	74.36	0.50	303
15	38.91	0.54	4.21	0	0	0.36	129.85	0	315
16	86.58	11.42	4.08	0	0	0.31	136.88	0.10	271
17	61.02	6.76	5.60	0.48	0.29	0.58	122.47	0	332
18	28.95	4.41	8.91	0.19	0.19	0.19	94.73	0.19	380
19	58.48	7.17	9.11	0	0.09	0.29	82.87	0.09	410
20	86.26	3.08	2.48	0	1.39	0	70.60	3.38	503

References

[1] Bäck, Th. and Schwefel, H.P. (1993). An Overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1:1-23.

[2] Beasley, J.E. (1985). A note on solving large p-median problems. European Journal of Operational Research, 21:270-273.

[3] Beasley, J.E. (1990). OR-Library: distribution test problems by electronic mail. Journal of Operational Research Society, 41:1069-1072.

[4] Beasley, J.E. (1993). Lagrangean heuristics for Location problems. European Journal of Operational Research, 65:383-399.

[5] Christofides, N. and Beasley, J.E.(1982). A tree search algorithm for the p-median problems. European Journal of Operational Research, 10:196-204.

[6] Davis, L.D. (1991). Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

[7] De Jong, K. (1975). An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI.

[8] Densham, P.J. and Rushton, G. (1992). A more efficient heuristic for solving large P-median problems. Papers in Regional Science, 71:307-329.

[9] Efroymson, M. A. and Ray, T. L. (1966). A branch-and-bound algorithm for plant location. Operations Research, 14:361-368.

[10] Galvão, R.D. and Raggi, L.A. (1989). A method for solving to optimality uncapacitated location problems. Annals of Operations Research, 18:225-244.

[11] Galvão, R.D. and ReVelle, C.S. (1996). A Lagrangean heuristic for the maximal covering location problem. European Journal of Operational Research, 18:114-123.

[12] Garey, M. R. and Johnson, D. S. (1979). Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, San Francisco, USA.

[13] Goldberg, D.E. (1989). Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, MA, 11-172.

[14] Goldberg, D.E., Korb, B. and Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems, 3:493-530.

[15] Goldberg, D.E., Deb, K., Kargupta, H. and Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. IlliGAL Report No. 93004, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois, Urbana.

[16] Goodchild, M. F. and Noronha, V. (1983). Location-allocation for small computers. Monograph No. 8, Department of Geography, University of Iowa, Iowa city.

[17] Grefenstette, J.J. (1987). Proceedings of the second international conference on genetic algorithms and their applications. Hillsdale, NJ, Lawrence Erlbaum.

[18] Hakimi, S.L.(1964). Optimum distribution of switching centers and the absolute centers and the medians of a graph. Operations Research, 12:450-459.

[19] Hakimi, S.L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Operations Research, 13:462-475.

[20] Holland, J.H. (1975). Adaptation in natural and artificial systems. MIT Press,11-147.

[21] Jarvinen, P., Rajala, J. and Sinervo, H. (1972). A branch and bound algorithm for seeking the p-median. Operations Research, 20:173-178.

[22] Johnson, E.L. and Mehrotra, A. (1992). Min-Cut Clustering. Working paper. School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta.

[23] Kargupta, H. (1995). Search, Polynomial Complexity, and The Fast Messy Genetic Algorithm, Ph.D. thesis, IlliGAL Report No. 95008, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois, Urbana.

[24] Klein, K. and Aronson, J.E. (1991). Optimal clustering: a model and method. Naval Research Logistics, 38:447-461.

[25] Lorena, L.A.N. and Lopes, F.B. (1996). A dynamic list heuristic for 2D-cutting. In J. Dolezal and J. Fidler, editors, System Modeling and Optimization, Chapman-hall, London, pages 481-488.

[26] Lorena, L.A.N. and Lopes, L.S. (1996). Computational experiments with genetic algorithms applied to set covering problems. Pesquisa Operacional, 16:41-53.

[27] Lorena, L.A.N. and Lopes, L.S. (1997). Genetic algorithms applied to computationally difficult set covering problems. Journal of the Operational Research Society, 48:440-445.

[28] Maniezzo, V., Mingozzi, A. and Baldaci, R. (1998). A bionomic approach to the capacitated p-median problem. Technical report - University of Bologna.

[29] Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin.

[30] Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press, Cambridge, England.

[31] Mulvey, J.M. and Beck, M.P. (1984). Solving capacitated clustering problems. European Journal of Operational Research, 18:339-348.

[32] Neebe, A.W. (1978). A branch and bound algorithm for the p-median transportation problem. Journal of the Operational Research Society, 29:989-995.

[33] Osman, I. H. and Christofides, N. (1994). Capacitated clustering problems by hybrid simulated annealing and tabu search. International Transactions in Operational Research, 1(3):317-336.

[34] Pearl, J. (1985). Heuristics - Intelligent search strategies for computer problem solving, Addison-Wesley Publishing Company, MA.

[35] Ribeiro Filho, G. and Lorena, L. A. N. (1997). A constructive genetic algorithm for graph coloring. Presented at Asia-Pacific Operations Research Scocieties'97 conference, Melbourne, Australia.

[36] Rolland, E., Schilling, D. A. and Current, J. R. (1997). An efficient Tabu search procedure for the p-median problem. European Journal of Operational Research ,96:329-342.

[37] Rosing, K. E.. and ReVelle, C. S. (1997). Heuristic concentration: Two stage solution construction. European Journal of Operational Research, 97:75-86.

[38] Rosing, K. E., ReVelle, C. S., Schilling D. A. and Current, J. R. (1998). Heuristic concentration and Tabu search: A head to head comparison. European Journal of Operational Research, 93-99.

[39] Senne, L. F. and Lorena L. A. N. (2000). Lagrangean/Surrogate Heuristics for p-Median Problems. In Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, M. Laguna and J. L. Gonzalez-Velarde (Eds.), Kluwer Academic Publishers, London, pp. 115-130.

[40] Spath, H. (1980). Cluster Analysis Algorithms for data reduction and classification of objects. Ellis Horwood Publishers, New York.

[41] Teitz, M.B. and Bart, P. (1968). Heuristic methods for estimating the vertex median of a weighted graph. Operations Research, 16:955-961.

[42] Zupan, J. (1982). Clustering of Large Data Sets. John Wiley & Sons, New York.