Sitting Guests at a Wedding Party: Experiments on Genetic and Evolutionary Constrained Optimization

Miguel Rocha
Departamento de Informática
Universidade do Minho
Braga - PORTUGAL
mrocha@di.uminho.pt

Rui Mendes
Departamento de Informática
Universidade do Minho
Braga - PORTUGAL
azuki@di.uminho.pt

Paulo Cortez
Departamento de Informática
Universidade do Minho
Braga - PORTUGAL
pcortez@di.uminho.pt

José Neves
Departamento de Informática
Universidade do Minho
Braga - PORTUGAL
jneves@di.uminho.pt

Abstract- The complex task of giving out tables to guests, according to their preferences, at a wedding party, instantiates a broader class of clustering problems, whose purpose is to group a number of entities into a number of clusters, according to a set of hard constraints, and optimizing an objective function.

In order to study the application of *Genetic and Evolutionary Algorithms (GEAs)* to these class of problems, some experiments were conducted. These contemplated different approaches to constraint handling, namely the use of penalty functions and decoders. The encoding issue was also studied, being compared direct and indirect representations of the problem's solutions in the chromosomes. The development of hybrid genetic operators, that combine the synergies of the *GEAs* paradigm with those of problem dependent heuristics, were also taken into account.

The overall result is a study on the performance of several approaches to constrained optimization by *GEAs*, that can be used to guide the application of the paradigm in real-world problems, in the *Combinatorial Optimization* arena.

Keywords: Hybrid Genetic and Evolutionary Algorithms, Direct and indirect representations, Combinatorial Optimization, Grouping problems, Minimum/Maximum k-clustering sum problem.

1 Introduction

Only those who have not been called upon to organize a wedding party are unfamiliar with the complexity of the underlying table's assignment task. In fact, assigning the proper table to each guest, considering the network of complicities or preferences among the entities involved, is typically very hard. In fact, a number of constraints have to be taken into account, dictated by the set of, more or less strict, social rules. For example, husband and wife must sit together and a guest should not be isolated from his/her relatives and/or friends.

The problem can be viewed as an optimization process, in terms of a solution's space S and an objective function f, that assigns numeric values to any $s \in S$. Being possible to define S and f, the solution to the problem is given by the *maximum* of f. It is obvious that, in this case, S should be the set of all possible configurations of guests per table. The function f is harder to define, once it is a non-trivial task to evaluate a particular solution, given the diversity of criteria available.

At a glance, the problem may seem to be of little interest, from an optimization point of view. A closer look, however, reveals a different insight. In fact, the problem represents an instance of a broader class of *Combinatorial Optimization (CO)* problems, where the aim is to group a number of entities into a number of classes, complying with a pre-defined set of constraints and optimizing a given objective function, defined over the set of allowed configurations. One can also view these problems as consisting in the search for a given partition of the set of entities (V) into a collection of mutually disjoint subsets V_i, where $V_i \bigcap V_j = \emptyset, \forall i \neq j$ and $\bigcup_i V_i = V$. These are named *grouping*, *partitioning* or *clustering* problems.

These class includes a number of well known *CO* problems, such as the *Bin Packing*, the *Graph Coloring*, the *Graph Partitioning* or the *Three Matching*, just to name a few. In each case, the set of constraints and the objective function diverge. For example, in the *Graph Coloring* problem, the constraint is that no connected entities are allowed in the same cluster, and the objective function is the number of different colors needed for a given graph. On the other hand, the *Bin Packing* problem imposes a maximum capacity for each bin, but optimizes a similar function.

The application of *Genetic and Evolutionary Algorithms (GEAs)* to this kind of problems is strongly motivated by the success on the paradigm's application to solve complex problems in the *CO* arena. In fact, although some skepticism from researchers in the traditional field of *Operations Research*, the use of *GEAs* has been growing and the results obtained in some tasks are quite impressive.

Good examples are the applications on the eternal *Traveling Salesman Problem* [11], on the complex *Quadratic Assignment Problem* [15] and on several types of *Scheduling* [6] and *Timetabling* [3] tasks. Therefore, it is not a surprise to find applications of *GEAs* to the mentioned grouping tasks. Some of the earlier approaches are summarized in Table 1.

In this work, the task of assigning tables to guests in a wedding party is a starting point for a voyage through several alternative approaches for the application of *GEAs* to clustering problems. The comparison of direct and indirect representations of the problem's solutions into the genomes is con-

0-7803-6657-3/01/$10.00 ©2001 IEEE

Table 1: A Summary of Approaches to Grouping Problems by *GEAs*

Problem	Authors
Graph Coloring	Galinier and Hao [12] Fleurent and Ferland [10]
Graph Partitioning	Talbi and Bessiére [25] Mühlenbein [19]
Bin Packing	Reeves [22] Falkenauer [8][9]
Three Matching	Magyar, Johnson and Nevalainen [16]

sidered, as well as the definition of operators that combine the strengths of the genetic and evolutionary search, with those of heuristics tailored for specific situations.

In what follows, the problem is firstly presented and formulated; then, *GEA* approaches which make use of direct representations and penalties are presented; next, the introduction of hybrid genetic operators is considered and evaluated; the following section is devoted to a description and an evaluation of the use of indirect representations; finally, some conclusions are drawn and prospective future work is presented.

2 The problem's formulation

The afore mentioned task, in what follows designated by *Table Assignment Problem (TAP)*, can be viewed as a *maximum k-clustering sum problem* [1], proven to be NP-complete. The problem is defined in terms of:

- a set V of entities;
- an *attraction* function $a : V \times V \mapsto \Re$, defined for each pair of entities in V; and
- a number of m clusters $(C_1, C_2, \ldots, C_m)$,

aiming at the partition of V into mutually disjoint clusters, of equal cardinality (S), that maximizes the sum of the *attraction function* values, for all pairs of entities assigned to the same cluster. A similar problem is given by the corresponding minimization task.

Thus, the problem can be postulated as:

$$\textit{Maximize/ Minimize} \quad \sum_{k=1}^{m} \sum_{i,j \in C_k} a(i,j) \qquad (1)$$

$$\textit{subject to:} \quad |C_k| = S, \forall k \in \{1, \ldots, m\} \qquad (2)$$

$$C_k \cap C_l = \emptyset, \forall k, l \in \{1, \ldots, m\} \qquad (3)$$

In the *TAP* case, the clusters are the tables and its size is given by the maximum number of guests allowed to sit in each table. The function a represents the strength of the relationship for every pair of guests; i.e., $a(i,j)$ is the added value given by sitting guests i and j in the same table. The objective function can be defined as the sum of the attraction values for every pair of guests in the same table. In this case, the function $a()$ can be represented by a symmetric matrix A, being $a(i,j)$ given by a_{ij}, if $i < j$ or by a_{ji}, otherwise.

In practice, the construction of matrix A by hand is onerous. So, a process to achieve the automation of this task was devised. The logic programming paradigm was chosen to provide for the computation framework, since it presents a simple syntax and allows for the generation of new knowledge from previous one, with minimum effort.

A set of logical predicates was developed to define the relationships between the guests (e.g., parents, couples, friends). Each kind of relationship was given a numerical weight, with closer links being rewarded with higher values. A convenient scheme was developed to uncover all the relationships between every pair of guests, weighting it and putting it into A.

The particular instance used in the experiments is based on a real case, with a total of 160 guests to be assigned to 20 clusters, with a maximum size of 8. The relationships between the guests were defined in a set of logical predicates and the matrix A built according to the above defined methodology.

A similar problem was considered, where the entities to be grouped are scattered in a bi-dimensional space, being therefore characterized by its x and y coordinates. The *attraction function* is, in this case, given by the *Euclidean* distance between the pairs of entities. The purpose is to group them in a pre-defined number of clusters, with equal cardinalities, in order to minimize the sum of the distances between all pairs of entities assigned to the same cluster. This could be used as the first stage to a *Vehicle Routing* problem, by partitioning the clients to be served by the different vehicles, and then solving a *TSP* instance for each cluster. These are named *two-phase heuristics* [2].

In the experiments presented below, an instance of the problem with 280 entities and a cluster size of 20 was used, being the coordinates of the problem taken from a *TSP* instance from TSPLIB [23], named *a280*. In what follows, this will be named the *Euclidean Clustering Problem (ECP)*.

3 Direct representations with penalty functions

The term *Genetic and Evolutionary Algorithm (GEA)* is used to name a family of computational procedures, where a number of potential solutions to a problem is evolving simultaneously within a *population*. Each individual codes a solution in a string (*chromosome*) of symbols (*genes*), being assigned to each a numerical value (*fitness*), that stands for the solution's quality.

New solutions are created through the application of genetic operators (typically crossover or mutation). The whole process evolves via a process of stochastic selection biased to favor the strongest individuals (the ones with higher fitnesses). Traditional *GEAs* use a binary representation in the codification of solutions to the target problem. More recently, some researchers found advantages in the use of representation schemes closer to the solution's space [18].

A solution to the *k-clustering sum* problem can be given in terms of n tuples $\langle v, c \rangle$, $v \in V$ and $c \in \{C_1, \ldots, C_m\}$ (being n the cardinality of V and m the number of clusters). Assuming that the entities are sequentially numbered, with an integer value from the set $\{1, \ldots, n\}$, a solution s can be represented as a sequence of integers $s[1], \ldots, s[n]$, where the index i represents the entity's number and $s[i] \in \{1, \ldots, m\}$ the number of the cluster assigned to it. A straightforward encoding scheme is to consider one gene per object, coding the cluster assigned to it. Thus, each gene will be given in the alphabet $\{1, \ldots, m\}$.

Under this coding scheme, there is a number of genomes that code for solutions that are not feasible, since they do not comply with the problem's restriction regarding the equal clusters' cardinalities. Thus, there is a constraint handling problem to be faced.

This problem may be handled by the introduction of penalties to constraint violations. In this case, the evaluation function for a solution s turns out to be given by

$$Eval(s) = F(s) - P(s) \tag{4}$$

where $f(s)$ is the objective function, and $P(s)$ is the penalty function. In (4), and in the discussion that follows, it is assumed a maximization problem, although the conversion for a minimization one tends to be straightforward.

It is important, when defining a penalty function, to have some measure of the degree of the constraints' violation, by the proposed solution. In this way it is possible to assign higher penalties to larger violations. In the *k-clustering sum* problem, it is logical that this measure should be the total number of entities in excess, considering all the clusters. This measure for a given cluster k is given by the expression:

$$exc_k = \begin{cases} 0 & \text{if } |\{x : x \in C_k\}| \leq 0 \\ |\{x : x \in C_k\}| & \text{otherwise} \end{cases} \tag{5}$$

The simplest penalty function, in terms of the degree of violation, is given in the form:

$$P(s) = \rho \times \sum_{k=1}^{m} exc_k \tag{6}$$

where ρ denotes a constant. In order to find a proper value for ρ, a compromise has to be found. If the penalty is too hard the *GEA* will have difficulties in finding near-optimal solutions. On the other hand, if it is too light, the *GEA* will tend to keep invalid solutions. In this case, it was intended to find a value that would denote the maximum profit an entity can achieve by being placed improperly; i.e., by being assigned to a cluster with a cardinality above the limit.

In order to get such a measure, the sum of the attractions, for each entity, was calculated. The average of this value over all the entities was taken, as the measure looked for. Since the matrix is symmetric, such value can be calculated as:

$$\rho = 2 \cdot \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} A_{ij}}{N} \tag{7}$$

Under this representation, three different crossover operators were used, namely the *one-point*, the *two-point* and the *uniform* ones, whose implementation is similar to those of their binary counterparts [13][24]. It must be noticed that these operators are general-purpose, and that no guarantee is granted that the offspring will code a feasible solution, even when both ancestors are valid individuals.

In what mutation is concerned, it was considered a *non-adjacent swap* operator, that exchanges the value of the genes in two randomly selected positions in the chromosome. This operator is also general purpose, but it is guaranteed that only feasible offspring is created with feasible ancestors.

Some experiments were conducted under this approach, taking into consideration the problem instances for the *k-clustering sum* problem, referred in Section 2, namely the *TAP* and the *ECP*. The purpose was to observe whether the constraint handling procedure was capable of producing feasible solutions, and also to study the behavior of the different genetic operators.

Selection is performed by ordering the individuals in a linear ranking and then performing a *roulette wheel* sampling. In every generation, 40% of the individuals are kept to the next generation and the remaining are created through the application of genetic operators.

In each run a crossover and a mutation operator were chosen, being the former responsible for breeding 75% of the offspring, and the latter the remaining ones. A population size of 200 individuals was used, being the process terminated after a fixed computation time. This termination criteria makes sure that all the approaches presented in this work have a fair comparison. The maximum time was set to 300 seconds in the *TAP*, and to 2000 in the *Euclidean* one.

The implementation of the different *GEAs* described in this work took advantage of a development environment, the *Genetic and Evolutionary Programming Environment (GEPE)* [20], that runs under the *Linux* operating system, being programmed in the C++ language. The common background gives an extra assurance of the fair comparison between the different alternatives. All the experiments were conducted in a PC with a *Pentium II 350 MHz* processor.

Each experiment was repeated 20 times, with randomly generated initial populations, being the results given in Table 2. The first column shows the crossover operator used; the second shows the objective function of the best solution found (the average of the 20 runs) for the *TAP*; the last one shows the same results for the *ECP* (keep in mind that the former is a maximization problem and that the last is a minimization one).

Prior to an analysis of the table, it is important to emphasize that the penalty methodology used allowed to obtain feasible solutions in all the runs. An analysis of the table shows larger differences between the operators in the *EP* case, where the *one-point crossover* performs better.

Table 2: Experimental results for the GEA with direct representations and penalties

Crossover operator	TAP	ECP
One-point	682.8	**8531**
Two-point	**690.5**	9336
Uniform	679.9	16157

4 Order-based representations

In an *Order Based Representation (OBR)*, the chromosome is a permutation of the symbols from a given alphabet. Without loss of generality, the genes in the alphabet can be considered as the integers in the set $\{1, 2, \ldots, N\}$. The genetic operators designed to work with this kind of genomes are necessarily different, since they have to maintain valid permutations in the genotypes.

Therefore, the mutation operators are typically based on a reordering of the chromosome. One common attitude is to consider an operator that swaps the value of two genes in random positions in the chromosome.

For the crossover operation a number of alternatives have been developed. Historically, the problem was approached using the *Partially Matched Crossover (PMX)*[14], where two crossing points are randomly chosen, defining a *matching section* on the string used to effect a cross between the two ancestors, through position-to-position exchange operations.

A different family of operators is the *Order Preserving* one, where the emphasis is put on the relative order of the genes from both parent [4]. The process may be based in the random selection of one (*OPX1*) or two (*OPX2*) cutting points, or in the generation of a binary mask, thus defining the *Uniform Order Preserving Crossover (UOPX)* [5].

Another alternative is found with the *Cycle Crossover* [21], that performs recombination under the constraint that each node (gene) in a certain position must come from one parent or the other.

In the *k-clustering sum* problem, the fact that the size of the clusters must be constant has worked to make the problem more complex, under the previous approach. However, this situation can also work on one's behalf, by using a different representation scheme, taking advantage of *OBR*'s. In this case, it is considered that the first S entities will be on cluster C_1, the next S entities on cluster C_2, and so on, being S the cluster size. Using this approach every permutation codes for a valid solution.

Some experiments were conducted under this new framework, by considering the afore mentioned crossover and mutation operators and keeping the settings of previous experiments.

The results are presented in Table 3, and seem to be quite better than those obtained by the previous approach. In terms of the crossover operator, the differences are not quite significative, although the *PMX* offers the best overall behavior.

Table 3: Experimental results for the *Order Based Representation* approach.

Crossover operator	TAP	ECP
OPX1	674.2	6867
OPX2	**701.8**	5813
UOPX	482.3	6014
PMX	701.5	**5727**
Cycle	693.3	5812

5 Hybrid approaches

5.1 The group based crossover

Both approaches devised so far are based on general purpose representations and operators. Furthermore, the representations are redundant, since the same solution can be represented by a number of different chromosomes. It is also arguable that the crossover operators do not recombine the information of the ancestors in a meaningful way. These facts lead some researchers to propose a different representation scheme to grouping problems, and to devise new genetic operators [19][7].

This path was also followed in this work, although it was considered to be sufficient to reformulate the genetic operators, in order to take effectively into account the information in the genotypes.

The proposed operators were devised to work with both representation schemes proposed earlier, since they work on the solution itself (the phenotype) and not on the genotype. Each solution is defined by the clusters it induces. If m is the number of clusters, a solution can be given in terms of m sets, $V_1, \ldots, V_m$, where $e \in V_i$, if i is the cluster assigned to e.

The new crossover operator devised, the *Group Based Crossover (GBX)*, considers two ancestors and generates one offspring. In the first step, one of the ancestors is selected, and a randomly selected subset of its clusters is automatically passed into the offspring. The effective number of clusters is chosen randomly in the range $[m/3; 2m/3]$. The entities that are assigned a cluster in this first step are removed from the clusters in the second ancestor.

In the next stage, the clusters of the second parent are ordered by their sizes (without the removed entities). These clusters are taken in decreasing order, and completed by choosing the required number of entities, from those not yet assigned. Once this entities are assigned they are removed from the clusters in the second parent, being the process finished when all clusters are empty.

This process is executed in a greedy way; i.e., in each step it is chosen the available entity that optimizes the objective function, from a local point of view. The process is stopped when all clusters in the offspring are complete.

The *GBX* was tested in the same problems as the previous approaches, and under similar conditions. Both representations of the previous sections were used, and the results are shown in Table 4.

Table 4: Experimental results for the *Group Based Crossover*.

Representation	TAP	ECP
Direct	751.7	5598
OBR	**755.2**	**5581**

It is not surprising to verify that the results were significantly improved over the standard crossover operators. The results obtained under the *OBR* are slightly better than those of the direct representations with penalties, but the computational overhead required by the constraint handling mechanism does not seems to be overwhelming, since the results are on similar levels for both problems.

5.2 Local optimization operators

Hybrid approaches, that combine *GEAs* with local optimization heuristics have been increasingly developed to tackle *CO* problems, with considerable success. Some examples of this research trend has been materialized into applications to *Graph Coloring* [12], *Quadratic Assignment* [17], *Traveling Salesman* [11] or *Bin Packing* [9] problems.

A local optimization mutation operator was considered for the *k-clustering sum* problem, although it is based on an analogy with the *2-opt* operator, designed for the *TSP*. The proposed operator exchanges two entities in different clusters, being this exchange accepted only if the objective function of the overall solution is improved. This mechanism is repeated for all possible pairs of entities, in a one-pass strategy.

In the direct representation with penalties, when the operator is applied to an invalid solution, it first moves towards feasibility. To achieve such purpose, it first finds a cluster with an excessive number of assigned entities, and moves one of its members to another cluster, not yet completely full.

The proposed local optimization operator was tested, being a replacement for the standard mutation operators. The results obtained, with the standard and the group crossover operators, under the conditions postulated above, are shown in Tables 5 and 6, for the each of the two proposed representations.

Table 5: Experimental results for the local optimization operator (direct representations with penalties).

Crossover operator	TAP	ECP
One-point	701.7	5633
Two-point	699.5	5612
Uniform	705.3	5824
GBX	**739.9**	**5546**

A closed look at the table reveals that the best results are obtained by combining the *GBX* with a local optimization procedure, under the *OBR* scheme. The same combination, but using a direct representation, with penalties to handle constraints, leads to slightly worse results (about 2%). In general,

Table 6: Experimental results for the local optimization operator (order based representations).

Crossover operator	TAP	ECP
OPX1	723.9	5486
OPX2	722.3	5514
UOPX	720.7	5537
PMX	722.8	5522
Cycle	718.2	5560
GBX	**755.3**	**5459**

and with any crossover operator, the *2-opt* operator works significantly better than the standard mutation one.

6 Indirect representations

The problems of constraint handling in *GEAs* may be tackled by the use of indirect representations, where an individual does not directly encode a solution, but instead it defines a strategy to reach one, by means of an heuristic decoder, that is problem dependent. This strategy presents an obvious advantage, once every possible chromosome codes for a valid solution.

On the other hand, it requires the definition of a proper decoding procedure that can determine, in a large sense, the success of the approach, that implies extra computational resources. In the definition of a decoding scheme, a compromise has to be found between the greediness of the approach and a correct representation of the search space. A greedy approach can induce good results in a short time, but can also reduce substantially the genetic diversity of the *GEA*'s populations, by considering only local optima.

It is possible to use the previously defined *Order Based Representations*, in order to implement an indirect representation, in grouping problems. Under this scheme, the chromosome is a permutation of the entities considered, that defines the order by which the entities are assigned. In each step, an entity is assigned a cluster in a way that no invalid solution is reached, and optionally optimizing a given criterion.

In the *k-clustering sum* problem, this criterion turns out in finding the cluster which locally leads to the best objective function. While there are unoccupied clusters, each entity has to choose whether to join an existing cluster or to start a new one. This decision is based on the expectation of starting a better cluster, instead of sticking to the ones available.

In order to make such decision, the best possible cluster for the entity is calculated, with a basis on the entities still to assign. If the average attraction to an already started cluster is above a given threshold, measured as a percentage of its optimum value, it is accepted. Otherwise, the entity starts a new cluster and takes its chances. A value of 90% for the threshold was found to work well on the problems considered.

When, on the other hand, all possible clusters are already occupied with at least one entity, the decision is to choose the one that locally optimizes the objective function; i.e., the one

that maximizes (or minimizes) the average of the attraction in the chosen cluster.

The results of the application of this approach, in the instances and under the settings defined in the previous sections, is shown in Table 7.

Table 7: Experimental results for the indirect representation.

Crossover operator	TAP	ECP
OPX1	731.7	5580
OPX2	748.4	**5500**
UOPX	**755.2**	5520
PMX	670.2	5808
Cycle	736.2	5558

Looking at the table, one is faced with fair results, almost at the same level of quality of the hybrid approaches. Indeed, the best result obtained for the *TAP* is similar to the best of the previous alternatives, but in the *ECP* case it is unable to reach the same result, although being quite close (with an error of no more than 1%).

This small difference is probably explained by the fact that the indirect representation does not search the same solutions space, since the decoder makes decisions that may eliminate a number of possible solutions. Most of these solutions are of poor quality, and their removal is beneficial but, on the other hand, there are no guarantees that the optimal solution can be obtained by using a greedy decoder.

7 Overall analysis

To make a judgment of the several *GEAs* approaches it is important to assert each method's capacity of obtaining good solutions in a short period of time. In Figures 1 and 2 it is shown the evolution of the results, according to the CPU time (in seconds), for the two problems and for the best three approaches, namely:

- Direct representations with penalties, the *GBX* and local optimization;
- Direct order based representations and identical operators; and
- Indirect order based representations, the *UOPX* operator and standard swap mutation.

The first conclusion to be drawn from the figures is that all the methods good quality results in a short fraction of the total amount of time put for the comparison. Furthermore, the *hybrid* approaches are faster in their initial convergence, while the indirect approach takes more time to achieve similar behavior.

It is paramount to have a measure of the quality of these results. This is a complex task, since the optimum solution to each of the problems is difficult to find, given the NP-completeness of the tasks. Some simple heuristics were implemented to achieve a comparison, although it is known that

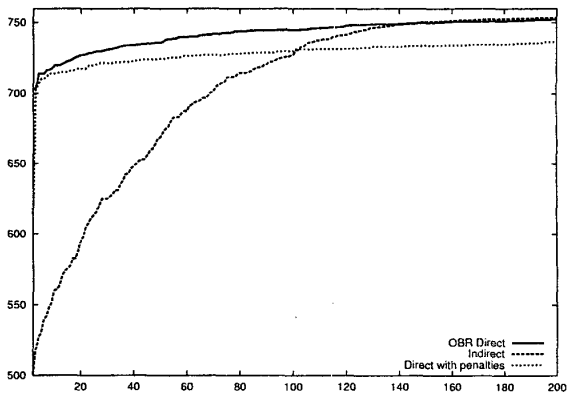

Figure 1: Evolution of the results obtained for the *TAP* per time unit (in seconds).

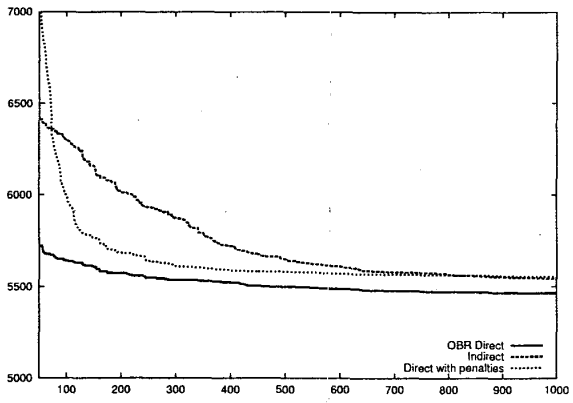

Figure 2: Evolution of the results obtained for the *ECP* per time unit (in seconds).

these are typically less demanding in terms of computational resources.

The first heuristic implemented is a greedy one, where the entities are considered in order and assigned the best cluster, by locally optimizing the objective function. The other heuristic implemented was the *complete 2-opt*, where the one-pass *2-opt*, described above, is applied until no improvement is possible. One last comparison was made with a strategy that uses a model similar to the *GEA* one, but considering only a local optimization operator in the generation of new individuals. A similar computation time was given to all the methods equal to that of the *GEA*'s experiments. In all cases the results show the average of the best solution over 20 runs of each method.

The best results of the *GEA* based approaches are better than those obtained with these heuristics. Furthermore, in both problems, the best solution was repeatedly found with different approaches, and in several runs, which leads to the

Table 8: Experimental results for the heuristic procedures.

Heuristic	TAP	ECP
Greedy	731.3	5907
Complete 2-opt	714.9	5532
Local optimization only	717.2	5563

belief that the optimum solution was found. The value of the objective function for this solutions is of 755.5 for the *TAP*, and of 5452 for the *ECP*.

8 Conclusions and future work

It has not been the purpose of this work to compare *GEAs* with other problem solving paradigms in a specific task. In fact, although the results were considered satisfactory, tests with more challenging instances and comparisons with other meta-heuristics would be necessary.

The purpose of the exercise was, rather, to compare some genetic and evolutionary approaches in a given constrained optimization problem. The lessons learned from this work can help in finding more interesting problem solving strategies to apply to real world problems, where the constraints are typically numerous, and most traditional methods fail. A travel through constraint handling techniques was endeavored, and the conclusions reached are indeed logical, yet curious.

The use of indirect representations, based on a well built decoder, showed some interesting results, that are expected to be confirmed in other studies. This approach is a good alternative to be considered in constrained problems and is of a generic nature. On the other hand, the good results claimed with the increasing use of *hybrid GEAs* was also confirmed in this study. This was, indeed, the best method.

The use of penalties and direct representations was the approach that presented the less competitive results. However, when considering the use of hybrid operators, under this representation the results improved significantly, reaching a level of quality closed to that of its competitors.

It must be remembered that the use of order based representations, to a direct encoding of the problem's solutions, was only allowed by the circumstance of the clusters being of equal sizes. In a more general case, this representation would not be possible, and the consideration of penalties would be inevitable. Under this scenario, the comparison of this approach with indirect representations is relevant, and the advantage of the latter is not definitive, since both showed competitive behavior.

This work showed that it is possible to obtain good results with general purpose encodings, but that they should be used by the genetic operators having into consideration the meaning of the information they encode. These simple representations have some advantages in terms of the computational implementation of *GEAs*, since they make use of fixed size chromosomes. Indeed, in Nature, the diversity of living beings is handled by one single genetic representation.

In terms of future work, it is intended to apply the proposed approaches to other problems, namely to those in the same class, such as *Bin Packing*, *Graph Coloring* or *Three-Matching* ones, and to other constrained optimization tasks, such *Scheduling* and *Vehicle Routing*. In all cases, a comparison with other approaches (e.g., *Simulated Annealing*, *Tabu Search*, *Ant Colony Optimization*) should be performed to evaluate its merits.

Acknowledgements

The work of Paulo Cortez was supported by the portuguese *Foundation of Science & Technology* through the PRAXIS XXI/BD/13793/97 grant. The work of José Neves was supported by the PRAXIS' project PRAXIS/P/EEI/13096/98.

Bibliography

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spacamella, and M. Protasi. *Complexity and Approximation - Combinatorial optimization problems and their approximability properties*. Springer Verlag, November 1999.

[2] Alex Van Breedam. A parametric analysis of heuristics for the vehicle routing problem with side-constraints. RUCA Working Paper 96/13, University of Antwerp, 1996.

[3] Dave Corne, Peter Ross, and Hsiao-Lan Fang. Evolutionary Timetabling: Practice, Prospects and Work in Progress. Technical report, University of Edinburgh, Edinburgh, UK, 1994.

[4] L. Davis. Job-shop scheduling with genetic algorithms. In *J. Grefenstette, editor, Proc. Int. conf. on GAs*. Lawrence Erlbaum, 1985.

[5] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[6] C. Dimopoulos and A. Zalzala. Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions and comparisons. *IEEE Transactions on Evolutionary Computation*, 4(2), July 2000.

[7] Emanuel Falkenauer. A new representation and operators for gas applied to grouping problems. *Evolutionary Computation*, 2(2):123–144, 1994.

[8] Emanuel Falkenauer. Setting new limits in bin packing with a grouping ga using reduction. Technical Report RO108, CRIF-Research Center for Belgian Metalworking Industry, March 1994.

[9] Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. *J.Heuristics*, 2(1):5–30, 1996.

[10] C. Fleurent and J. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique and satisfiability. *DIMACS series in Discrete Mathematics*, 1994.

[11] Bernd Freisleben and Peter Merz. New Genetic Local Search Operators for the Travelling Salesman Problem. In *Proceedings of the 4th. Parallel Problem Solving From Nature (PPSN IV)*, Berlin, September 1996. Springer Verlag.

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

[13] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., NY, USA, 1989.

[14] D. Goldberg and R. Lingle. Alleles, Loci and the Traveling Salesman Problem. In J.Grenfenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Hillsdale, New Jersey, 1985. Lawrence Erlbaum Associates.

[15] M.H. Lim, Y. Yuan, and S. Omatu. Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications*, (15):249–268, 2000.

[16] G. Magyar, M. Johnsson, and O. Nevalainen. An adapative hybrid genetic algorithm for the three-matching problem. *IEEE Transactions on Evolutionary Computation*, 4(2):135–146, July 2000.

[17] Peter Merz and Bernd Freisleben. A genetic local search approach to the quadratic assignment problem. In *Proc. of the 7th International Conference on Genetic Algorithms(ICGA'97)*, 1997.

[18] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition, 1996.

[19] H. Mühlenbein. Parallel genetic algorithm in combinatorial optimization. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research - New Developments in Their Interfaces*. Pergamon Press, 1992.

[20] J. Neves, M. Rocha, H. Rodrigues, M. Biscaia, and J. Alves. Adaptive Strategies and the Design of Evolutionary Applications. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99)*, Orlando, Florida, USA, 1999.

[21] I.M. Oliver, D.J. Smith, and J. Holland. A Study of Permutation Crossover Operators on the Travelling Salesman Problem. In J.Grenfenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*. Lawrence Erlbaum Associates, July 1987.

[22] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. In *Proceedings of the Annual Conference of Operational Research Society*, York, September 1993.

[23] G. Reinelt. Tsplib'95. Technical report, Universitat Heidelberg, 1995.

[24] Gilbert Syswerda. Schedule optimization using genetic algorithms. In L.Davis, editor, *Handbook of Genetic Algorithms*. Van Nostrand, 1991.

[25] E-G. Talbi and P.Bessiere. A parallel genetic algorithm for the graph partitioning problem. In *Proceedings of the International Conference on Supercomputing*, 1991.