Improved Genetic Operators for the Multiobjective Generalized Assignment Problem

Robert F. Subtil[1], **Eduardo G. Carrano**[1], **Ricardo H. C. Takahashi**[2], **Marcone J. F. Souza**[3], **Sérgio R. de Souza**[1]

[1]Centro Federal de Educação Tecnológica de Minas Gerais.
Av. Amazonas, 7675
30510-000 – Belo Horizonte, MG – Brazil

[2]Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627
31270-901 – Belo Horizonte, MG – Brazil

[3]Universidade Federal de Ouro Preto
Campus Universitário
35400-000 – Ouro Preto, MG – Brazil

rsubtil@yahoo.com.br, egcarrano@dppg.cefetmg.br

taka@mat.ufmg.br, marcone@iceb.ufop.br, sergio@dppg.cefetmg.br

Abstract. *The original formulation of the Generalized Assignment Problem (GAP) consists in, given a set of n different tasks and m different agents, assigning each task to an agent in such a way that a cost function is minimized. A previous work introduced the Equilibrium Function as a new objective function in the problem formulation. The purpose of this second objective function is to minimize the maximum difference between the amount of work assigned to the agents. This allows better distributions of the tasks between the agents than the results found from the original problem, with a small increase in the cost. This paper proposes new crossover and mutation operators that produce improvements in the algorithm presented in [Subtil et al. 2010], leading to considerably better Pareto approximations than the ones obtained in the previous work, within the same number of function evaluations. The proposed operators exploit problem-specific information in a probabilistic way, performing operations that lead to objective function enhancement or feasibility enhancement with greater probability than operations that do not cause such enhancements. A statistical comparison procedure is employed for supporting such conclusions.*

1. Introduction

The Generalized Assignment Problem (GAP) is a classical $\mathcal{NP}$-Hard problem [Sahni and Gonzalez 1976, Chu and Beasley 1997] which is used for modeling several practical design problems, such as job assignment in computer networks [Balachandran 1976], parallel machine scheduling [Lenstra et al. 1990], multiprocessor scheduling [Turek et al. 1992], facility location [Ross and Soland 1977] and vehicle routing [Fisher and Jaikumar 2006].

In its usual form, the GAP is stated as follows: given a set $\mathcal{T}$ of tasks, a set $\mathcal{A}$ of agents, the cost of each task j when it is accomplished by each agent i and the amount of resources required by each agent i for completing each task j, find the optimal assignment X such that the sum of the costs spent for accomplishing all tasks is minimum, ensuring, on the other hand, that each task is assigned to a single agent and that the total of resources demanded from agent i is lower than its upper bound.

Therefore, if an optimization algorithm is employed to solve such problem, it should ideally find the less expensive assignment policy, regardless the distribution of the tasks amongst the agents. This characteristic of this optimal solution may render it unsuitable for several practical problems, for instance:

People management: If some people of a team receives much more work than other people, they tend to become dissatisfied and their efficiency may decline due to the excessive workload.

Computer networks or parallel computing: A task allocation policy that concentrates the processing in some few resources is more likely to cause bottlenecks than a more balanced distribution policy.

In practice, a more equitable distribution of the tasks can be helpful in any situation in which load balance can be relevant. In [Subtil et al. 2010] a multiobjective adaptation of the original GAP (referred as Multiobjective GAP or MoGAP) is proposed. In this new formulation, a second objective function, called Equilibrium Function, is introduced. The purpose of this function is to guide the search in order to reach solutions which provide better distribution of the tasks amongst the agents. Since two objective functions are considered, the solution of this problem is a set of efficient points, in which a solution can not be chosen without introducing any preference.

This paper proposes an extension of the integer NSGA-II proposed in [Subtil et al. 2010]. The new algorithm employs improved versions of the crossover and mutation operators, leading to considerably better Pareto approximations than the ones obtained by the former algorithm. In order to support such a comparison, a detailed statistical evaluation procedure is employed. Comparisons of the quality of the approximations achieved and the quality of a deterministically chosen solution are also performed.

This paper has the following structure: the formulation of the MoGAP is shown in section 2; the improved algorithm, which is proposed for dealing with MoGAP, is presented in section 3; finally, a statistical comparison of the two algorithms which have been considered is performed on section 4.

2. The Multiobjective Generalized Assignment Problem

Let $\mathcal{A}$ be the set of agents, $\mathcal{T}$ be the set of tasks, $c_{i,j}$ be the cost of assigning the task $j \in \mathcal{T}$ to the agent $i \in \mathcal{A}$, $a_{i,j}$ be the amount of resources required by the agent $i \in \mathcal{A}$ for performing the task $j \in \mathcal{T}$, b_i be the total resource capacity of agent $i \in \mathcal{A}$ and $x_{i,j}$ be a binary decision variable, which assumes value "1" if the task $j \in \mathcal{T}$ is assigned to agent $i \in \mathcal{A}$ or value "0", otherwise. The MoGAP can be stated as [Subtil et al. 2010, Yagiura et al. 2006]:

$$\mathcal{X}^* = \arg\min_x \begin{bmatrix} f_C(x) \\ f_E(x) \end{bmatrix} \tag{1}$$

$$\text{subject to :} \begin{cases} (G1): \sum_{j \in \mathcal{T}} a_{i,j} \cdot x_{i,j} - b_i \leq 0 & , \forall i \in \mathcal{A} \\ (G2): \sum_{i \in \mathcal{A}} x_{i,j} - 1 = 0 & , \forall j \in \mathcal{T} \\ (G3): x_{i,j} \in \{0,1\} & , \forall i \in \mathcal{A},\ \forall j \in \mathcal{T} \end{cases} \tag{2}$$

in which:

$$f_C(x) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{T}} c_{i,j} \cdot x_{i,j} \tag{3}$$

$$f_E(x) = \max_{i \in \mathcal{A}} \left(\sum_{j \in \mathcal{T}} a_{i,j} \cdot x_{i,j} \right) - \min_{i \in \mathcal{A}} \left(\sum_{j \in \mathcal{T}} a_{i,j} \cdot x_{i,j} \right) \tag{4}$$

In this formulation, the equations (3) and (4) are the cost and equilibrium functions, respectively. The equilibrium is modeled as the difference between the most busy agent and the most vacant one. This difference should be minimized. In (2), the constraint $g_1(x)$ ensures that the maximum resource capacity of each agent is not violated, the constraint $g_2(x)$ ensures that each task is assigned to a single agent and the constraint $g_3(x)$ defines that each variable $x_{i,j}$ is binary. An important feature of this formulation is that it cannot be solved by an integer linear programming method, since $f_E(\cdot)$ is not linear.

3. Proposed Algorithm

The characteristics of the problem, which is combinatorial, multiobjective and has a non-linear objective function, preclude the employment of most part of the deterministic optimization methods. The genetic algorithms are particularly adequate for dealing with the MoGAP, since they are suitable for handling with multiobjective optimization problems with rather arbitrary objective and constraint functions [Coello 2000, Fonseca and Fleming 1995]. The algorithm proposed here is an improved version of the one described in [Subtil et al. 2010]. It is based on the NSGA-II [Deb et al. 2002], and includes also some problem-specific solution encoding, crossover and mutation mechanisms. A general scheme for this algorithm can be seen in Algorithm 1.

In this algorithm:

- $P \leftarrow$ **new_population**(N) generates a random population with N feasible individuals;
- $A \leftarrow$ **non_dominated**(P) returns the individuals which lie in the first front of the population P;
- $\mathcal{F} \leftarrow$ **fast_non_dominated_sorting**(P) employs *fast non-dominated sorting* [Deb et al. 2002] to find the front of each solution in the population P;
- $\mathcal{C}_i \leftarrow$ **crowding_distance_assignment**$(\mathcal{F}_i)$ employs *crowding distance assignment* [Deb et al. 2002] to estimate how the solutions of front i are spread in the objective space;
- $\mathcal{F}_i \leftarrow$ **sort**$(\mathcal{F}_i, \mathcal{C}_i,$ 'descending') sorts the solutions of front i in descending order of $\mathcal{C}_i$;

Algorithm 1 Pseudocode for NSGAII

procedure NSGAII(N,N_A)
  $t \leftarrow 0$
  $P_t \leftarrow$ **new_population**(N)
  $Q_t \leftarrow \emptyset$
  $A \leftarrow$ **non_dominated**(P_t)
  while not stop criterion **do**
    $R_t \leftarrow P_t \cup Q_t$
    $\mathcal{F} \leftarrow$ **fast_non_dominated_sorting**(R_t)
    $P_{t+1} \leftarrow \emptyset$
    $i \leftarrow 1$
    while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**
      $\mathcal{C}_i \leftarrow$ **crowding_distance_assignment**$(\mathcal{F}_i)$
      $P_{t+1} \leftarrow P_t \cup \mathcal{F}_i$
      $i \leftarrow i + 1$
    end while
    $\mathcal{F}_i \leftarrow$ **sort**$(\mathcal{F}_i, \mathcal{C}_i,$ 'descending')
    $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
    $Q_{t+1} \leftarrow$ **selection**(P_{t+1}, N)
    $Q_{t+1} \leftarrow$ **GAPCrossover**(Q_{t+1})
    $Q_{t+1} \leftarrow$ **GAPMutation**(Q_{t+1})
    $t \leftarrow t + 1$
    $A \leftarrow$ **non_dominated**$(A \cup Q_t)$
  end while
end procedure

- $Q \leftarrow$ **selection**(P, N) uses binary stochastic tournaments for performing selection of the population P. The outcome of this procedure is a population Q, with N individuals which have been selected from P with replacement.

The selection, fitness assignment, niche and elitism procedures employed in the proposed algorithm are identical to the canonical ones of NSGA-II, presented in [Deb et al. 2002]. The encoding scheme, crossover and mutation operators have been chosen in order to adapt to the problem structure. These components are described in the sequel.

3	4	4	2	1	4	3	2	2	4

Figure 1. Example of the encoding scheme. In this candidate solution, the task 5 is assigned to agent $A1$, the tasks 4, 8 and 9 are assigned to agent $A2$, the tasks 1 and 7 are assigned to agent $A3$ and the tasks 2, 3, 6 and 10 are assigned to agent $A4$.

3.1. Encoding Scheme

The encoding scheme adopted here was the same proposed in [Subtil et al. 2010]. In this representation, each candidate solution is represent by a $1 \times |\mathcal{T}|$ integer vector. Each position of this vector can assume any integer value in $\{1, |\mathcal{A}|\}$. An example of encoding for an instance with 4 agents and 10 tasks is shown in Figure 1.

3.2. GAPCrossover

The crossover operator introduced in this paper, called GAPCrossover, is a modification of the original uniform crossover, commonly employed in binary GAs. However, it carries some knowledge about the problem, performing the steps described in Algorithm 2.

Algorithm 2 GAPCrossover

Input: parent solutions p_1 and p_2

Output: offspring solutions o_1 and o_2

generate a random binary word (ref) of length $|\mathcal{T}|$;
generate a scalar d_T, at random, following an uniform distribution;
$o_1 \leftarrow p_1$
$o_2 \leftarrow p_2$
for each position j in which $ref \neq 0$ **do**
    if $d_T < 0.50$ **then**
        swap $o_1(i)$ and $o_2(i)$;
    else
        $o'_1 \leftarrow o_1$
        $o'_2 \leftarrow o_2$
        swap $o_1(i)'$ and $o_2(i)'$;
        if $g_1(o'_1) \leq 0$ and $g_1(o'_2) \leq 0$ **then**
            if $c_{o_1(i)',j} < c_{o_2(i)',j}$ **then**
                $o_1 \leftarrow o'_1$
                $o_2 \leftarrow o'_2$
            end if
        else if $g_1(o'_1) \leq g_1(p_1)$ and $g_1(o'_2) \leq g_1(p_2)$ **then**
            $o_1 \leftarrow o'_1$
            $o_2 \leftarrow o'_2$
        end if
    end if
end for

From this scheme, it is possible to note that the algorithm has 50% of probability of performing the recombination, and 50% of probability of swapping the selected positions only in the case of a improvement condition is verified. In the first case, the operator is identical to the original uniform crossover, while, in the second case, the operator only swaps the values which provide one of the following improvements:

- the solution o_1 improves the cost of parent p_1; or
- the two new solutions inflict the constraint $g_1(\cdot)$ less than their respective parents.

If none of these conditions are not observed, then the positions are kept unchanged. An example of this operator, when it reduces to the uniform crossover case, can be seen in Figure 2.

It should be noticed that this operator generates less unfeasible individuals than the original uniform crossover, since a "feasibility-oriented" mechanism is included in some conditions. In some tests, this mechanism was executed in all situations, in order to completely avoid the generation of unfeasible solutions. This version of the operator was

p_1	3	4	4	2	1	4	3	2	2	4
p_2	1	3	4	1	1	2	2	3	2	3
ref	0	1	0	0	1	1	1	0	0	1
o_1	3	3	4	2	1	2	2	2	2	3
o_2	1	4	4	1	1	4	3	3	2	4

Figure 2. Example of uniform crossover: Let $d_T < 0.50$. p_1 is copied to o_1 and p_2 is copied to o_2. Then, the values of o_1 and o_2 are swapped in the positions in which the $ref(i) = 1$ (gray columns).

discarded, because it presented lower performance than the one shown here. The authors believe that the degradation of performance can be credited to the loss of population diversity caused by such an operator.

3.3. GAPMutation2

The GAPMutation2 proposed in this paper is a small variation of the GAPMutation proposed in [Subtil et al. 2010]. Both operators are based on the flip mutation, and employ some kind of additional procedure for incorporating problem knowledge, as shown in Algorithm 3.

Algorithm 3 GAPMutation2

Input: parent solution p

Output: offspring solution o

choose a position $j \in \{1, |\mathcal{T}|\}$, at random;
generate a scalar r_D, at random, following uniform distribution;
$o \leftarrow p$
if $r_D \leq 0.20$ **then**
    replace $o(j)$ by a new agent $a \neq p(j)$, at random;
else if $r_D > 0.20$ and $r_D \leq 0.50$ **then**
    replace $o(j)$ by the agent a with the highest contribution $f_E(m)$ (least required agent) and $a^* \neq p(j)$;
else
    replace $o(j)$ by the agent a^* in which $a^* = \arg\min_a c_{a,j}$ with $a^* \neq p(j)$, ensuring that $g_1(o) \leq 0$;
end if

p	3	4	4	2	1	4	3	2	2	4
						⇓				
o	3	4	4	2	1	1	3	2	2	4

Figure 3. Example of GAPMutation2. Let $j = 6$ and $0.20 < r_D < 0.50$ and assume that all tasks demand the same amount of resources of all agents. The task 6 has been re-assigned to the least required agent, which is $A1$.

Such as for GAPCrossover, a version of GAPMutation which ensures the feasibility of all obtained solutions had been tested. This version was not considered on further

implementations of the algorithm since it reduced the global efficiency of the method. An example of the GAPMutation2 can be found in Figure 3.

3.4. Constraint Handling

The solution encoding and the operators described earlier in this section ensure that $(G2)$ and $(G3)$ constraints (see equation 2) are always valid. However, some additional mechanism is necessary to employ for handling $(G1)$, since it is not possible to *a priori* ensure that this constraint is valid for all individuals of the algorithm. Following the same methodology proposed in [Subtil et al. 2010], an exponential penalty function was employed in order to penalize the solutions which do not comply with such a constraint. The main difference between this approach and the previous one is that, here, the penalty function is applied to both objective functions simultaneously. These modified objective functions are given by:

$$f_C^m(x) = f_C(x) \cdot \epsilon^{r_{ind}} \tag{5}$$

$$f_E^m(x) = f_E(x) \cdot \epsilon^{r_{ind}} \tag{6}$$

in which

$$r_{ind} = \sum_{i \in \mathcal{A}} \max \left\{ 0, \sum_{j \in \mathcal{T}} (a_{ij} \cdot x_{ij}) - b_i \right\}$$

4. Numerical Results

In reference [Subtil et al. 2010], it has been proposed an integer enhanced NSGA-II for solving the MoGAP. That algorithm was compared with a basic NSGA-II and with an exact method for the mono-objective problem. The comparison, in [Subtil et al. 2010], was guided by visual inspection, based on five independent runs of each algorithm. It was noticed that the algorithm proposed there clearly dominated the basic one, and was able to provide good approximations to the exact optimal cost solution in five of six instances. Besides, it was always possible to choose a solution which provided favorable ratio *% gain in equilibrium* vs. *% loss in cost*. The algorithm of [Subtil et al. 2010] will be referred in the remainder of this paper as $\mathcal{A}_1$.

The current work proposes an enhancement of that previous algorithm, in order to improve its convergence. This new algorithm, denoted by $\mathcal{A}_2$, is compared with the previous version, $\mathcal{A}_1$. The comparison with the basic GA is not performed here, since this algorithm is clearly dominated by $\mathcal{A}_1$. Before presenting the results, it is important to explain how the comparisons were conducted.

Comparison scheme

The algorithm comparison methodology which is considered here is inspired on the evaluation schemes proposed in [Carrano et al. 2008, Carrano et al. 2011]. It can be summarized as follows:

- Each algorithm is executed k times, for a fixed number of function evaluations (stop criterion).

- For each algorithm i:
  - For each run j:
    - Evaluate a merit criterion, $m_{crit}(i,j)$, of the final archive achieved in the run j of the algorithm i.
  - Perform bootstrapping [Efron 1979] with the values $m_{crit}(i,:)$. Each bootstrapping iteration delivers a sample of the mean of the merit function for the algorithm i. The whole set provided by the bootstrapping is an empirical approximation of the probability distribution function (PDF) of such a mean.
- Compare the empirical PDF's of the algorithms using One-Way ANOVA [Lindman 1974], for a significance level $\alpha = 0.05$.

Comparison criteria

Two independent merit criteria have been considered in the comparison:

- Hyper-volume indicator: the hyper-volume indicator [Zitzler 1999] is used to estimate the quality of the final Pareto approximation achieved in each run;
- Decision-making: a solution is automatically chosen for each algorithm run. Such a solution is the one which maximizes:

$$i^* = \arg \min_i \frac{f_E^{*C} - f_E^i}{f_C^i - f_C^{*C}} \cdot \frac{f_C^{*C}}{f_E^{*C}}$$

$$\text{subject to: } \begin{cases} f_C^i \neq f_C^{*C} \\ f_E^i \neq f_E^{*C} \end{cases}$$

  in which $[f_C^i \;\; f_E^i]'$ is the function vector assigned to the solution i and $[f_C^{*C} \;\; f_E^{*C}]'$ is the function vector of the optimal cost solution (known a priori).

The rule for decision-making has been chosen arbitrarily and, therefore, it could be replaced by any other reasonable rule. The main goal of considering these two criteria is to estimate the quality of the global Pareto approximation found and the quality of a particular solution picked from such approximations.

Computational procedures

The algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ have been tested in six instances, which have been taken from the OR-Library [Beasley 2010]:

- $A05 \times 100$: 5 agents and 100 tasks;
- $A05 \times 200$: 5 agents and 200 tasks;
- $A10 \times 100$: 10 agents and 100 tasks;
- $A10 \times 200$: 10 agents and 200 tasks;
- $A20 \times 100$: 20 agents and 100 tasks;
- $A20 \times 200$: 20 agents and 200 tasks.

Both algorithms have been set with the following parameters:

- Number of runs: 100 runs per algorithm;
- Population size: 50 individuals;
- Stop criterion: maximum number of generations;
- Maximum number of generations: 300 generations;
- Crossover probability: 0.90 per pair;
- Mutation probability: 0.03 per integer.

The analysis of the results observed in the six instances has been divided in two sections, one for each merit criterion.

Table 1. Hyper-volume indicator - 95% confidence interval

	95% confidence interval	
instance	$\mathcal{A}_1$	$\mathcal{A}_2$
$A05 \times 100$	$[32333\ ;\ 33711]$	$[32788\ ;\ 34105]$
$A05 \times 200$	$[295322\ ;\ 305575]$	$[307332\ ;\ 311554]$
$A10 \times 100$	$[103116\ ;\ 111013]$	$[106169\ ;\ 112955]$
$A10 \times 200$	$[279897\ ;\ 301474]$	$[308347\ ;\ 322609]$
$A20 \times 100$	$[88885\ ;\ 100839]$	$[95353\ ;\ 106252]$
$A20 \times 200$	$[226007\ ;\ 254017]$	$[263840\ ;\ 290605]$

4.1. Hyper-volume indicator

After running the algorithms 100 times each, the Pareto approximations have been used for finding the values of the hyper-volume associated to each algorithm. These values have been used for finding 95% confidence intervals, which are shown in Table 1.

From Table 1, the bounds of the intervals observed for $\mathcal{A}_2$ are always higher than the ones noticed for $\mathcal{A}_1$. Since higher values of hyper-volume indicate better Pareto approximations, this analysis suggests that the algorithm $\mathcal{A}_2$ has found better results than $\mathcal{A}_1$. This hypothesis is supported by the ANOVA tests, which have indicated that there are significant differences between $\mathcal{A}_1$ and $\mathcal{A}_2$ in all instances[1].

In order to perform an evaluation of the amount of the difference that was found, a visual comparison of the Pareto approximations has been performed. Three Pareto approximations have been chosen: the approximation with lower hyper-volume, the median and the approximation with higher hyper-volume. These comparisons have shown that the Pareto approximations achieved by $\mathcal{A}_2$ always dominate most part of the $\mathcal{A}_1$ solutions. This can be noticed in Figure 4, which illustrates such a comparison for a median run.

4.2. Decision-making

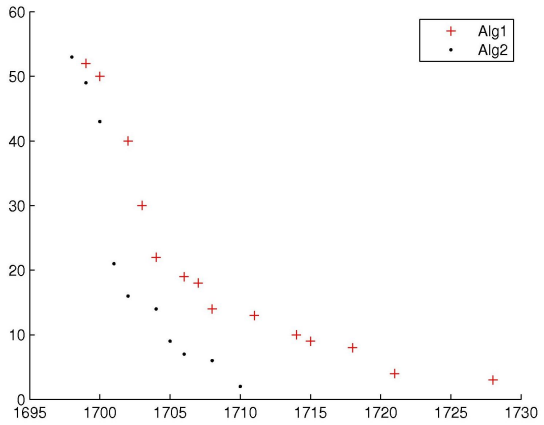
A deterministic rule for choosing a solution from each Pareto approximation has been adopted: the solution which provides better ratio *gain in equilibrium* vs. *loss in cost*. Therefore, a value of maximum ratio is assigned to each algorithm run and these values are used in the statistical comparison scheme discussed earlier in this section. The 95% confidence intervals for these ratios are shown in Table 2.

These intervals indicate that the algorithm $\mathcal{A}_2$ is considerably better than $\mathcal{A}_1$ in this merit criterion, since it is able to provide solutions with better ratios in all instances considered here. Such a conclusion was corroborated by the ANOVA tests, which detected significant differences between the algorithms in the six instances[2].

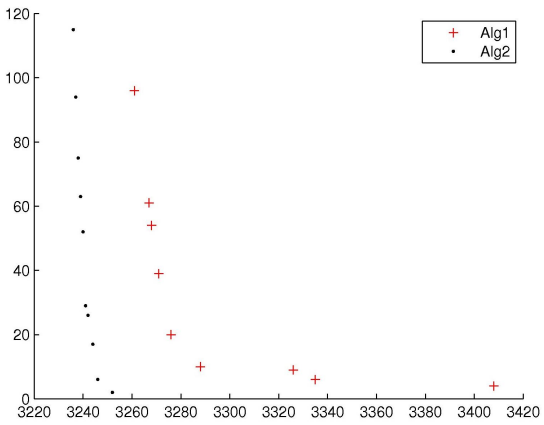
Considering both the Pareto approximation quality and the decision-making step, it can be concluded that the new proposed algorithm is better than the former one, at least for the instances considered.

[1] The higher p-value observed was lower than 1×10^{-16}.

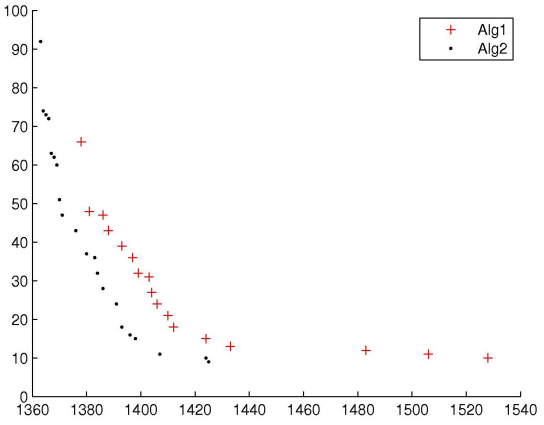
[2] All p-values were lower than 1×10^{-16}.



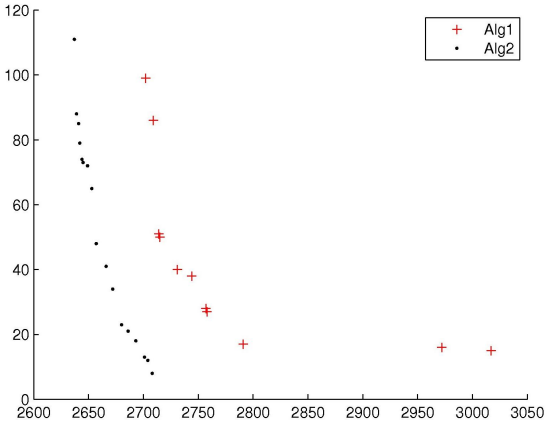
(a) $A05 \times 100$



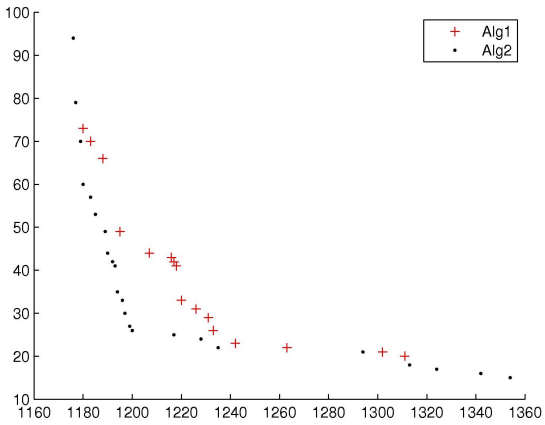
(b) $A05 \times 200$



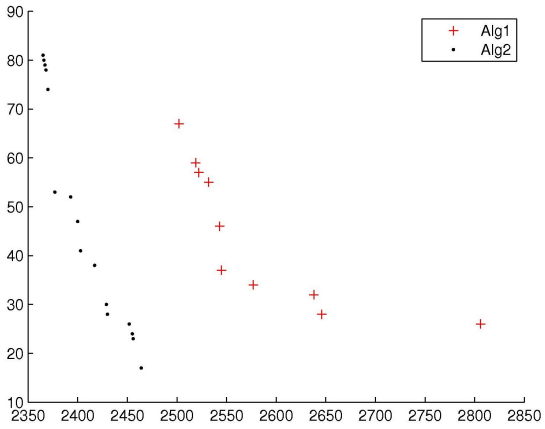
(c) $A10 \times 100$



(d) $A10 \times 200$



(e) $A20 \times 100$



(f) $A20 \times 200$

Figure 4. Comparison of the Pareto approximations (for medians)

Table 2. Decision-making - 95% confidence interval

	95% confidence interval	
instance	$\mathcal{A}_1$	$\mathcal{A}_2$
$A05 \times 100$	$[80.1\ ;\ 341.7]$	$[245.6\ ;\ 400.5]$
$A05 \times 200$	$[43.2\ ;\ 2280.0]$	$[518.4\ ;\ 3135.0]$
$A10 \times 100$	$[23.5\ ;\ 70.9]$	$[50.6\ ;\ 115.7]$
$A10 \times 200$	$[9.5\ ;\ 25.6]$	$[43.3\ ;\ 275.4]$
$A20 \times 100$	$[5.3\ ;\ 21.2]$	$[18.2\ ;\ 61.6]$
$A20 \times 200$	$[4.0\ ;\ 9.9]$	$[18.2\ ;\ 81.1]$

5. Conclusion

An enhanced version of an integer NSGA-II algorithm, specialized for the multiobjective generalized assignment problem (MoGAP), is proposed in this paper. This algorithm introduces two new specific genetic operators for improving the search for efficient solutions which perform with a higher probability operations that enhance the cost function or which enhance the solution feasibility. It is interesting to notice that these operators still perform stochastic operations that may still perform operations that do not immediately enhance the solutions. This has been found to be much better than deterministic local searches.

A statistical comparison of the proposed algorithm and the original version indicates that the new algorithm is more efficient for solving the considered problem. With identical computational cost, the new algorithm was able to find Pareto approximations which dominate the ones achieved by the former algorithm. Besides, the employment of a decision-making procedure has shown that the new approach delivers solutions with more favorable trade-offs, which allow significant gains in the task homogeneity assignment with small losses in the cost.

Acknowledgments

The authors acknowledge the support from the Brazilian agencies FAPEMIG, CAPES and CNPq.

References

Balachandran, V. (1976). An integer generalized transportation model for optimal job assignment in computer networks. *Operations Research*, 24:742–759.

Beasley, J. E. (2010). Or-library. `http://www.brunel.ac.uk/depts/ma/research/jeb/orlib/gapinfo.html`. Last accessed in 2010-Feb-01.

Carrano, E. G., Takahashi, R. H. C., and Wanner, E. F. (2008). An enhanced statistical approach for evolutionary algorithm comparison. In *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'08*, Atlanta, USA.

Carrano, E. G., Wanner, E. F., and Takahashi, R. H. C. (2011). A multi-criteria statistical based comparison methodology for evaluating evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. to appear.

Chu, P. C. and Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers and Operations Research*, 24:17–23.

Coello, C. A. C. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2):109–143.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7:1–26.

Fisher, M. L. and Jaikumar, R. (2006). A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124.

Fonseca, C. M. and Fleming, P. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

Lenstra, J. K., Shmoys, D. B., and Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming: Series A and B*, 46:259–271.

Lindman, H. R. (1974). *Analysis of Variance in Complex Experimental Designs*. W. H. Freeman & Co., San Francisco, USA.

Ross, G. T. and Soland, R. M. (1977). Modeling facility location problems as generalized assignment problems. *Management Science*, 24:345–357.

Sahni, S. and Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM*, 23:555–565.

Subtil, R. F., Carrano, E. G., Souza, M. J. F., and Takahashi, R. H. C. (2010). Using an enhanced integer NSGA-II for solving the Multiobjective Generalized Assignment Problem. In *Proc. IEEE World Congress on Computational Intelligence*, Barcelona.

Turek, J., Wolf, J. L., and Yu, P. S. (1992). Approximate algorithms scheduling parallelizable tasks. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, San Diego, USA.

Yagiura, M., Glover, F., and Ibaraki, T. (2006). A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, 169:548–569.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD thesis, Computer Engineering and Networks Laboratory - Swiss Federal Institute of Technology, Zurich.