A Comparative Study of Evolutionary Computation Techniques for the Generalized Assignment Problem

Roni F. Banaszewski, Heitor S. Lopes, Cesar A. Tacla, Jean M. Simão

Graduate Program in Electrical Engineering and Industrial Informatics (CPGEI)

Federal University of Technology-Paraná (UTFPR)

Av. 7 de setembro, 3165 - Curitiba (PR), Brazil

ronifabio@gmail.com;{hslopes, tacla, jeansimao}@utfpr.edu.br

Abstract. *This paper presents a comparative study among default implementations of Genetic Algorithm, Differential Evolution and Particle Swarm Optimization for load balancing, a specific case of the Generalized Assignment Problem (GAP). DE was found the best algorithm, and it was further tested with harder GAP instances to determine its best configuration, so as to be applied to other GAP-like problems.*

Resumo. *Este trabalho apresenta um estudo comparativo entre implementações padrão do Algoritmo Genético, Evolução Diferencial e Otimização por Enxame de Partículas no problema de balanceamento de carga, um caso específico do Problema Generalizado de Atribuição (GAP). Neste estudo comparativo, o algoritmo de Evolução Diferencial apresentou o melhor desempenho. Por isso, o algoritmo foi também experimentado em instâncias GAP mais complexas a fim de determinar a sua melhor configuração para execução deste tipo de problema.*

1. Introduction

Over the last decades, Evolutionary Computation (EC) techniques have been successfully applied to many complex optimization problems. Among the EC techniques, Genetic Algorithms (GA) (Holland, 1975) and, more recently, Differential Evolution (DE) (Price, 2005) and Particle Swarm Optimization (PSO) (Kennedy, 2001) are those which efficiency has been frequently pointed out in the recent literature. However, few works focus on the performance comparison between EC techniques for a given problem. This work aims at establishing an unbiased comparison between GA, DE and PSO for a relevant and well-known engineering problem, the Generalized Assignment Problem (GAP) (Osman, 1995),(Chu, 2007).

The specific instance of the problem addressed in this work is load balancing, that is, how to distribute work between computers so as to minimize the overall processing time and resources usage. The use of EC techniques for load balancing is suitable once new alternatives should be presented against the existing ones (e.g. Random Allocation, Round-Robin, Weighted Round Robin, and Fewest connections with limits), to be applied to static or dynamic load balancing problems. In the static mode, the balancing takes place before the execution of any process, whereas in the dynamic mode, it occurs at the execution time. This paper focuses the static mode, to assure conformity with GAP.

The paper is organized as follows: section 2 shortly describes the algorithms used in this study. Section 3 presents the formulation and approaches for modeling the problem. Section 4 shows graphically the results of experiments using GAP benchmarks and suggests suitable values for the running parameters of the best EC algorithm. Finally, section 5 discusses results and presents conclusions.

2. EC Techniques

This section shortly introduces the EC techniques used in this work. GA, DE, and PSO are somewhat similar in the sense of that they are population-based metaheuristics that can be used for optimization problems, but they differ significantly in their specific biological inspiration.

2.1. Genetic Algorithm (GA)

GA was proposed by John Holland in the 60's (Holland, 1975), and was inspired in the Darwin study about the evolution of living beings, including the principle of naturals election, hereditariness and genetic drift.

GA works with an initial population of randomly generated solutions. Each individual encodes the parameters of the problem in its chromosomes. Thus, the genetic material of an individual ultimately represents a possible solution to the optimization problem.

To each individual, a given value (i.e. fitness) is assigned, which represents how good it is to the specific problem in hand. This value is obtained by a mathematical function, called fitness function. Individuals of the population are submitted to a selection procedure, where the natural selection principle comes up.

Throughout generations, the average fitness of the population is expected to increase, since the best individuals (that is, those with the highest fitness values) of a population have an increasing probability to be selected for generating descendants.This is done by a selection method that maintains good individuals and rejects bad ones. Selected individuals undergo the effect of genetic operators, usually crossover and mutation. Algorithmically, this selection process occurs until a stop criterion is reached.

In this work, GA is configured with mutation and crossover rates of 0.01 and 0.7, respectively. Selection of individuals was accomplished by the tournament selection method, which selects $t \geq 2$ individuals to compare their fitness values; t means the tourney size, corresponding to 3% of the population size.

2.2. Differential Evolution (DE)

The Differential Evolution is a heuristic approach for minimizing nonlinear and non-differentiable continuous space problems (Price, 2005). DE works with a population of individuals represented by vectors, where four are randomly selected. One of them is chosen to be substituted (target vector) in the population and the other three are used as parents. One of the parents is chosen as the main vector (i.e. the best vector of the population ($\vec{\boldsymbol{X}}^{(t)}_{best,j}$) or a randomly one ($\vec{\boldsymbol{X}}^{(t)}_{rand,j}$)), and the remaining are chosen as secondary parents($\vec{\boldsymbol{X}}^{(t)}_{\alpha,j}, \vec{\boldsymbol{X}}^{(t)}_{\beta,j}$).

The mutation operation occurs when a scalar number F weighs the difference between the secondary parents and add it to the main parent, producing yet another vector called donor vector ($\vec{\boldsymbol{V}}_{i,j}^{(t+1)}$). Equation 1 presents the mutation process for the *j*th component of each vector according to the specific mutation strategy known as DE/Best/1/Bin:

$$\vec{\boldsymbol{V}}_{i,j}^{(t+1)} = \vec{\boldsymbol{X}}_{best,j}^{(t)} + \boldsymbol{F}\left(\vec{\boldsymbol{X}}_{\alpha,j}^{(t)} + \vec{\boldsymbol{X}}_{\beta,j}^{(t)}\right) \quad (1)$$

In order to increase the potential diversity of the population, a crossover operator mix values of the donor vector with the values of the target vector ($\vec{\boldsymbol{X}}_{target,j}$), resulting in a trial vector ($\vec{\boldsymbol{U}}_{i,j}^{(t)}$). In DE, the crossover can be performed by two different strategies: binomial and exponential.

Binomial and exponential crossovers are performed, respectively, on each variable according to the scheme outlined in Equations 2 and 3, where CR is a crossover constant. In the exponential crossover, an integer n is randomly chosen within the vector dimension (D) to act as starting point in the crossover process with the donor vector. Another integer (L) is chosen to set the number of components that the donor vector contributes to the target one. To keep the population size constant over subsequent generations, the selection operator determines if the target or trial vector will survive in the next generation.

$$\vec{\boldsymbol{U}}_{i,j}^{(t)} = \begin{cases} \boldsymbol{if\ rand}(\boldsymbol{0},\boldsymbol{1}) \leq \boldsymbol{CR},\ \vec{\boldsymbol{V}}_{i,j}^{(t)} \\ \boldsymbol{else}\ ,\vec{\boldsymbol{X}}_{target,j} \end{cases} \quad (2) \qquad \vec{\boldsymbol{U}}_{i,j}^{(t)} = \begin{cases} \vec{\boldsymbol{V}}_{i,j}^{(t)} \quad \boldsymbol{for\ j} = < n >_{\boldsymbol{D}}, \ldots < n - L + 1 >_{\boldsymbol{D}} \\ \vec{\boldsymbol{X}}_{target,j} \end{cases} \quad (3)$$

The control parameters for the experiments are F=0.5 and CR=0.1 with the classical Binomial DE/Best/1 strategy. The population size (NP) corresponds to 10 times the D value.

2.3. Particle Swarm Optimization (PSO)

PSO is based on the observation of the social behavior of groups of individuals (in this case, particles) in the nature. These individuals move in a search space where each position represents a possible solution to a problem. The movement of a particle is influenced by its own acquired experience and the experience of other neighbor particles (or the whole population).The PSO population consists of particles generated randomly where each one is evaluated by a pre-established fitness function. The particles move in the *n*-dimensional search space until a stop criterion is reached, such as a maximum number of iterations. The movement is guided by the Equations 4 and 5.

$$v_{new} = v_{old} * \omega + c1 * r * (p_{best} - p_{current}) + c2 * r * (g_{best} - p_{current}) \quad (4)$$

$$p_{new} = p_{current} + v_{new} \quad (5)$$

where,

v_{new}	: new velocity used to adjust the new position(p_{new})
v_{old}	: current velocity of each particle
ω	: inertia operator, controls the momentum of the particle
p_{best}	: keep the best position reached by own particle
$p_{current}$	: current position of the particle
g_{best}	: keep the swarm best position
r	: random number in the interval [0,1]
$c1$ *and* $c2$	: indicate, respectively, the influence of the own experience or of the swarm's. Controls the amount of local and global search

In the experiments, we used the default values of the literature for the control parameters (Price, 2005), as follows: c1=c2=2 and w=0.5.

3. Load Balancing (GAP-like) Formulations

This section introduces and mathematically describes the NP-Hard load balancing combinatorial problem (GAP -like) (Fisher, 1986) to be solved by the EC techniques. In fact, the EC techniques presented consider the problem as maximization, such that the balancing profit should be increased with respect to the problem restrictions.

The balancing profit computation is influenced by the proximity between the node that contains a process and those that contains most of the necessary resources for its execution. For instance, an optimal assignment takes place when a process *j* is attributed to a processor *i* that offer all the necessary resources for the execution of *j*. Thus, the processing capacity consumed by this assignment will be ideal, since no communication with remote nodes will be necessary to negotiate resources usage.

The objective function is presented in the Equation 6, where $p_{ij} \in ZZ+$ represents the profit associated with the assignment of the process *j* to processor *i*. Variable x_{ij} indicates whether ($x_{ij} = 1$) or not ($x_{ij} = 0$) process *j* is assigned to processor *i* .

Equation 7 represents two constraints. The first one refers to a processor capacity, where $w_{ij} \in ZZ+$ is the capacity demand of processor *i* by a process *j* (if *j* is assigned to *i*) and $b_i \in ZZ+$ is the capacity of processor *i*. The second constraint warrants that process *i* is assigned to only one processor.

$$\boldsymbol{maximize\ z = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\, x_{ij}} \qquad (6)$$

subjected to

$$\sum_{j=1}^{n} w_{ij}x_{ij} \leq b_i, \quad i \in M = \{1,2,\ldots,m\}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j \in N = \{1,2,\ldots,n\} \qquad (7)$$

$$\textbf{where } x_{ij} = \begin{cases} 1 & if\ process\ j\ assigned\ to\ processor\ i \\ 0 & otherwise \end{cases}$$

The fitness function is presented in Equation 8, with respect to the objective function and the constraints of the problem. This function penalizes unfeasible solutions with the normalized value returned by Equation 9 (i.e. penalty ϵ [0…1]). This value is multiplied by a penalty weight (default is 5).

$$\boldsymbol{fitness = \max(0, objectiveFunction - (penalty * weight))} \qquad (8)$$

The solution is penalized only if one or more processors *i* load more processes than they can execute in an acceptable performance. This penalty function solves the first constraint imposed by Equation 7.

$$\boldsymbol{penalty = \frac{\sum_{i=0}^{i} \max\left(0, \frac{consumedCapacity_i - totalProcessorCapacity_i}{totalProcessorCapacity_i}\right)}{amountOfProcessors}} \qquad (9)$$

The second constraint is solved directly by the way individuals are structured (i.e. GA chromosome, DE vector, and PSO particle), represented as an array of integers. It imposes that each array position *k* represents a process *j* to be assigned to only one processor *i*, identified by the value stored in the array position *k*.

4. Results

4.1. Comparative Study on EC Techniques

In order to compare the EC algorithms, we used the standard implementations with default parameters proposed in the literature. For each test instance we used the same number of fitness evaluations as a common ground for comparison of algorithms. The test instances shown in Table 1 were found in the ORLib (ORLib, 2012). Table 2 shows the general parameters settings used to execute each test instance with each algorithm.

Table 1. Test Instances

Test	Processes	Processors	Best Profit[1]	Best Fitness
a	15	5	336	0,954
b	20	5	434	0,947
c	30	5	656	0,968

Table 2. Parameters Settings

		Test a	Test b	Test c
PSO	Pop.	100	150	200
	Gen.	500	667	1000
AG	Pop.	200	200	250
	Gen.	250	500	800
DE	Pop.	150	200	250
	Gen.	334	500	800
Evaluations		50000	100000	200000

Each algorithm was run independently 20 times for each test instance, and the average performances of them are shown in Figure 1, where the horizontal axis represents the number of fitness evaluations and the vertical axis represents the fitness value, normalized between 0.5 and 1 to enlarge the visualization area of the plots.

As observed in the figure, DE presented the best results in the tests. DE presents a similar curve in all three experiments, with gradual growth towards the optimum. The curves of PSO were also similar in all the experiments. However, they represent the worst results, due to a premature convergence to a local maximum. It reasonable to infer that PSO performance could be improved by fine-tuning their control parameters, *c1*, *c2*, and *w*, during the execution. Finally, GA presented different form of curves though the tests. Its curves are identified for the high jumps in direction to the optimum. The basic GA performed reasonably well, therefore it is possible that improving GA with some hybrid functionality can improve the results, however this is out of the scope of this work.

[1] Sometimes the algorithms in experiment reached the best profit result

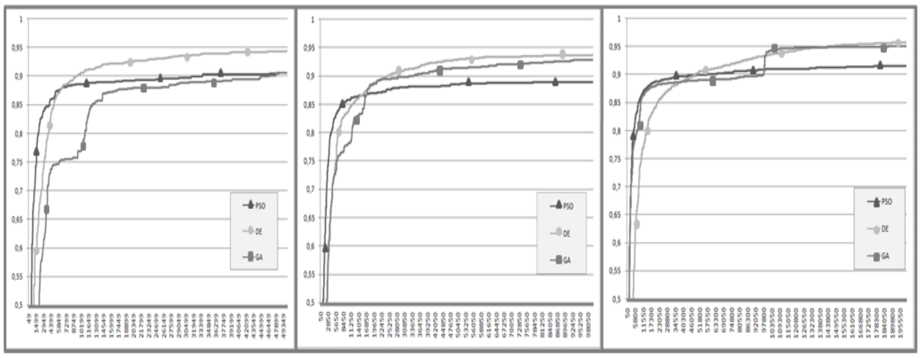

Fig. 1. Performance comparison of PSO, DE and GA, for the a, b and c test instances

Due to the superiority of DE in the experiments, further experiment were done to explore the effectiveness of this algorithm for the problem, now using more complex instances of test.

5. Experiments Using DE

In (Price, 2005), authors suggested 10 different strategies for DE and explain their applicability to optimization problems. These binomial and exponential strategies and their respective mutation scheme are presented in the Table 3. They are identified by the notation DE/x/y, where x specifies whether the vector will be disturbed by a vector randomly chosen (Rand), by the best vector in the population (Best), or even by the interaction between the target and the best vectors (RandToBest), and *y* is the number of difference vectors considered for the perturbation of x.

The experiments using DE for the GAP were subdivided in three steps, as follows:

a. A comparative study among the five binomial strategies;

b. A comparative study among the five exponential strategies;

c. A comparative study among the best strategies of the previous steps for determining the best one to be applied to GAP-like. In this process, the test instances used were more complex than those of the previous section, and are shown in Table 4.

Table 3. DE Strategies

Bin/Exp	Mutation scheme
DE/Best/1	$\vec{v}_i^{(t+1)} = \vec{X}_{best}^{(t)} + F\left(\vec{X}_\alpha^{(t)} - \vec{X}_\beta^{(t)}\right)$
DE/Rand/1	$\vec{v}_i^{(t+1)} = \vec{X}_{rand}^{(t)} + F\left(\vec{X}_\alpha^{(t)} - \vec{X}_\beta^{(t)}\right)$
DE/Best/2	$\vec{v}_i^{(t+1)} = \vec{X}_{best}^{(t)} + F\left(\vec{X}_\alpha^{(t)} - X_\beta^{(t)} + X_\gamma^{(t)} - X_\delta^{(t)}\right)$
DE/Best/2	$\vec{v}_i^{(t+1)} = \vec{X}_{rand}^{(t)} + F\left(\vec{X}_\alpha^{(t)} - X_\beta^{(t)} + X_\gamma^{(t)} - X_\delta^{(t)}\right)$
DE/RandToBest/1	$\vec{v}_i^{(t+1)} = \vec{X}_{target}^{(t)} + F.\left(\vec{X}_{best}^{(t)} - \vec{X}_{target}^{(t)} + \vec{X}_\alpha^{(t)} - \vec{X}_\beta^{(t)}\right)$

Table 4. Complex Test Instances Approach

Test	Processes	Processors	Best Profit	Best Fitness	DE Profit	DE Fitness	DE Profit	DE Fitness
d	30	10	709	0,967	703	0,959	706	0,963
e	40	10	958	0,981	952	0,975	955	0,978
f	50	10	1139	0,980	1114	0,958	1132	0,974
Amount of Fitness Evaluations			Not knew		500.000		1.000.000	

Table 4 presents the best profit and fitness values obtained by (ORLib, 2012), together with those obtained by the application of the best DE strategy indicated in this paper. The experiments consider, for the 3 tests (d, e and f), a total of 500.000 and 1.000.000 fitness evaluations, using a population of 300 vectors and the default parameters shown above.

In the first evaluation step, strategy DE/Best/1 was superior over the other binomial strategies during the optimization progress. In the second step, the strategies obtained similar results, differently from those from the binomial experiment. In test (d), the strategies DE/Best/1 and DE/Rand/1 did not present significant differences by the end of the evaluations cycle. However, when considering the whole cycle, the strategy DE/Best/1 presented better results. Moreover, as complexity of the instance increased (tests e and f), the DE/Best/1 was confirmed as the best exponential strategy for GAP.

Finally, Figure 2 presents the final comparison of performance among the Binomial DE/Best/1 and Exponential DE/Best/1. Analyzing the plot, a significant difference among the solutions is observed, such that the Exponential DE/Best/1 was found the best one for the GAP.

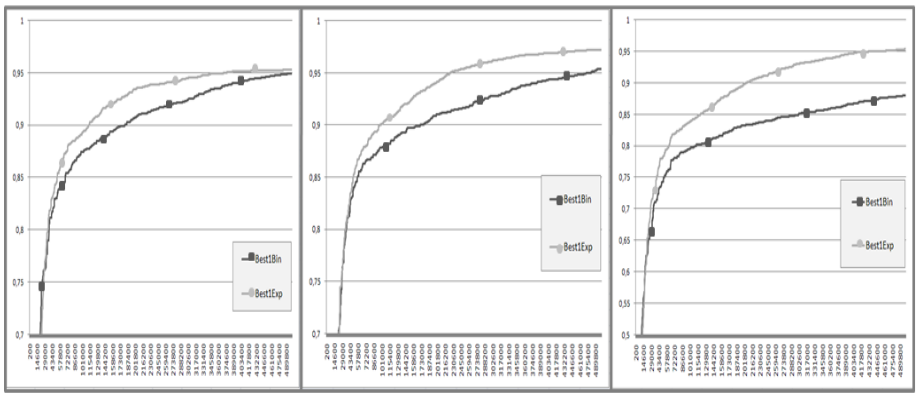

Fig. 1. Comparison on Better Binomial and Exponential Strategies

6. Conclusion

In this paper we compared three EC algorithms for the GAP. DE presented the best performance for the instances testes, since it converges quickly to high-quality results. Moreover, DE is a simple and robust algorithm that performed consistently in different runs and for different instances of the problem. The paper contribution is the comparison of some of the most popular EC algorithms and the definition of the best applicable parameters GAP instances of load balancing. Also, the paper presented the best DE strategy to deal with GAP. As conclusion, the DE exponential strategies found better results than the binomial ones, for instances tested, since in some experiments, the worst exponential strategy presented better results than the best binomial one. It is fair to conjecture that the particular DE strategy will perform similarly well for other instances of GAP. However, one cannot generalize our conclusions to other classes of problems.

References

Chu, P.C. and Beasley, J.E., (1997), “A genetic algorithm for the generalized assignment problem”, Computers and Operations Research, 24:17-23.

Fisher, M., Jaikumar, R. and Van Wassenhove, L., (1986), “A multiplier adjustment method for the generalized assignment problem”. Management Science, 32:1095-1103.

Holland, J.H., (1975),“Adaptation in Natural and Artificial Systems”, Ann Arbor: University of Michigan Press.

Kennedy, J., Eberhart, R. and Shi, Y., (2001), “Swarm Intelligence”, Waltham: Morgan Kaufmann.

ORLib GAP instances, (2012), Available in the internet at: http://people.brunel.ac.uk/~mastjjb/ jeb/orlib gapinfo.html . Accessed at 27/04/2012.

Osman, I.H., (1995), “Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches”. OR Spektrum, 17, 211-225.

Price, K., Storn, R. and Lampinen, J., (2005), “Differential Evolution – A Practical Approach to Global Optimization”, Berlin: Springer-Verlag.