12-1-2010

A Genetic Algorithm for Multiobjective Hard Scheduling Optimization

Elías Niño

Carlos Ardila

Alfredo J. Perez

Yexid Donoso

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 5, pp. 825-836

A Genetic Algorithm for Multiobjective Hard Scheduling Optimization

E. Niño, C. Ardila, A. Perez, Y. Donoso

Elías Niño, Carlos Ardila
Department of Computer Science
Universidad del Norte
Km 5, Via Pto Colombia. Barranquilla, Colombia
E-mail: {enino,cardila}@uninorte.edu.co

Alfredo Perez
Department of Computer Science and Engineering
University South Florida
4202 E. Fowler Ave. Tampa, Florida
E-mail: ajperez4@cse.usf.edu

Yezid Donoso
Department of Computing and Systems Engineering
Universidad de los Andes
Cra 1 No. 18A-12. Bogotá, Colombia
E-mail: ydonoso@uniandes.edu.co

Abstract: This paper proposes a genetic algorithm for multiobjective scheduling optimization based in the object oriented design with constrains on delivery times, process precedence and resource availability.

Initially, the programming algorithm (PA) was designed and implemented, taking into account all constraints mentioned. This algorithm's main objective is, given a sequence of production orders, products and processes, calculate its total programming cost and time.

Once the programming algorithm was defined, the genetic algorithm (GA) was developed for minimizing two objectives: delivery times and total programming cost. The stages defined for this algorithm were: selection, crossover and mutation. During the first stage, the individuals composing the next generation are selected using a strong dominance test. Given the strong restrictions on the model, the crossover stage utilizes a process level structure (PLS) where processes are grouped by its levels in the product tree. Finally during the mutation stage, the solutions are modified in two different ways (selected in a random fashion): changing the selection of the resources of one process and organizing the processes by its execution time by level.

In order to obtain more variability in the found solutions, the production orders and the products are organized with activity planning rules such as EDD, SPT and LPT. For each level of processes, the processes are organized by its processing time from lower to higher (PLU), from higher to lower (PUL), randomly (PR), and by local search (LS). As strategies for local search, three algorithms were implemented: Tabu Search (TS), Simulated Annealing (SA) and Exchange Deterministic Algorithm (EDA). The purpose of the local search is to organize the processes in such a way that minimizes the total execution time of the level.

Finally, Pareto fronts are used to show the obtained results of applying each of the specified strategies. Results are analyzed and compared.

Keywords: Scheduling, Process, Genetic Algorithm, Local search, Pareto Front.

1 Introduction

The Genetics Algorithms (GA) are a powerful tool for solving combinatorial problems. Nowadays, it exits a lot of algorithms inspired in GA for solving real problems such as design of vehicle suspensions [1], product deployment in telecom services [2], design of the flexible multi-body model vehicle suspensions based on skeletons implementing [3], job-shop scheduling [4], economic dispatch of generators with prohibited operating zones [5], multi-project scheduling [6], inversion analysis of permeability coefficients [7], path planning in unstructured mobile robot environments [8], rough mill component scheduling [9] and power plant control system design [10].

In productive systems is very critical the assignments of resources, for instance, a product has process and the process requires resources. The programming of the execution of the processes affects the overall cost and time of the products. Due to this, it is very important try to do the planning and scheduling in the best way. It can be accomplished with a Genetic Algorithm.

2 Preliminaries

2.1 Local Search

Local search are techniques that allows finding solutions in a set of solutions. It always tries to improve the actual solution through perturbations. A perturbation is a simple way for changing a solution. The perturbation depends of the way for representing the solutions, for instance in figure 1 can be seen a binary representation of a solution, in this case the perturbation can be done changing ones (1) by zeros (0). On the other hand, in figure 2 can be seen a no-binary representation of a solution (tour of the Traveling Salesman Problem [11] for example), in this case the perturbation can be done swapping two elements of the tour.

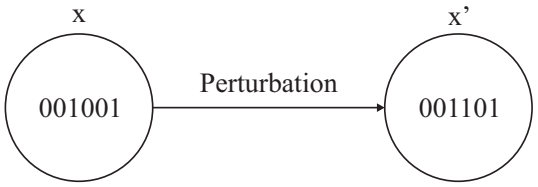

Figure 1: A binary representation of solutions. In this case x is the representation of the decimal number 9. The perturbation was done changing the 4th 0 to 1. Due to this, it creates a new solution x' that is the representation of the decimal number 13.

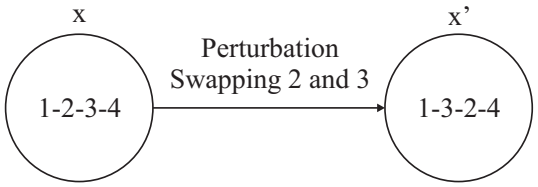

Figure 2: A no-binary representation of solutions. In this case x is the representation of a tour in TSP. The perturbation was done changing the 2 and 3 in the string. Due to this, it creates a new solution x'.

There exist a many local search algorithms such as Tabu Search (TS) [12], Simulated Annealing (SA) [13] and Exchange Deterministic Algorithm (EDA) [14]. Due to this, it is necessary to find a good representation of the solutions. Obviously, it depends of the problem to solve.

2.2 Genetic Algorithm

Genetic Algorithm (GA) is a search technique used for solving optimizations problem. The most important in GA is the design of the chromosome. It is the representation of the feasible solutions. Consequently, the behavior of the GA depends of the chromosome. Due to this, a bad chromosome implies a bad behavior of the GA. On the other hand, a good chromosome may imply a good behavior of the GA. The framework of GA can be seen in figure 3.

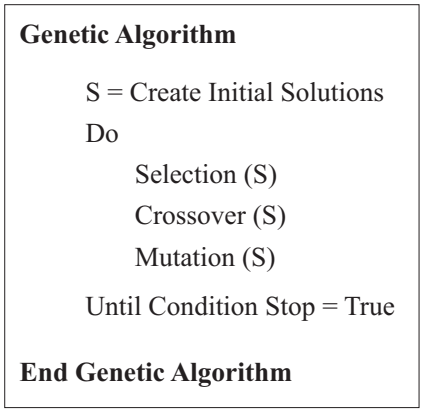

Figure 3: The Framework of a Genetic Algorithm.

GA has three important steps. First, it selects the solutions for the crossover and mutation step. This selection can be done using a metric, for example the Inverse Generational Distance (IGD) [15]. Second, it takes pairs of solutions for crossing. It can be done at random. Crossover consists in creates new solutions with parts of two solutions. The two original solutions are named parents (father and mother) and the two new solutions sons. It is created with half from father and half from mother. Lastly, it takes some sons for mutating it. The mutation is a step that allows creating new solutions. It can be done using a perturbation or a local search. The three steps of GA can be seen in figure 4.

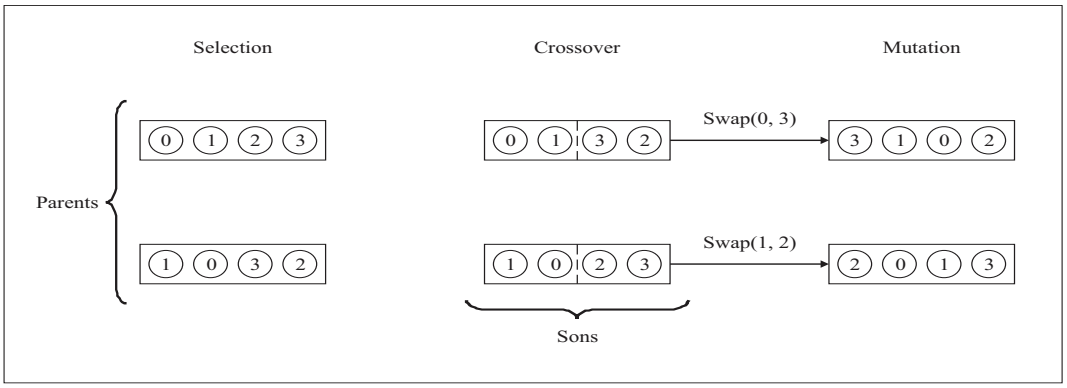

Figure 4: Steps of a Genetic Algorithm.

3 A Genetic Algorithm for Multiobjective Hard Scheduling Optimization

We state a new Genetic Algorithm for Multiobjective Hard Scheduling Optimization (GAMHSO). This algorithm works in scenarios with the following characteristics: there are production orders that are composed by a set of products. Each of the products is described by a product tree that contains all the processes needed to build such product.

For a product to be ready, it is required the execution of all processes that belongs to its tree. A process may need the execution of another process (precedence) or other subpart before it can be executed. The execution of some processes can be only done in certain times (schedules). To execute a process, a group of resources is required. The defined resources are: machinery, employees, and vehicles. It is not necessary for a group of resources to contain all types of resources.

Finally, if a subcomponent is required, it has to be constructed by a set of processes or it can be modeled by a process that indicates idle state (the subcomponent has not arrived yet to the system). This scenario can be seen in a domain model in figure 5.

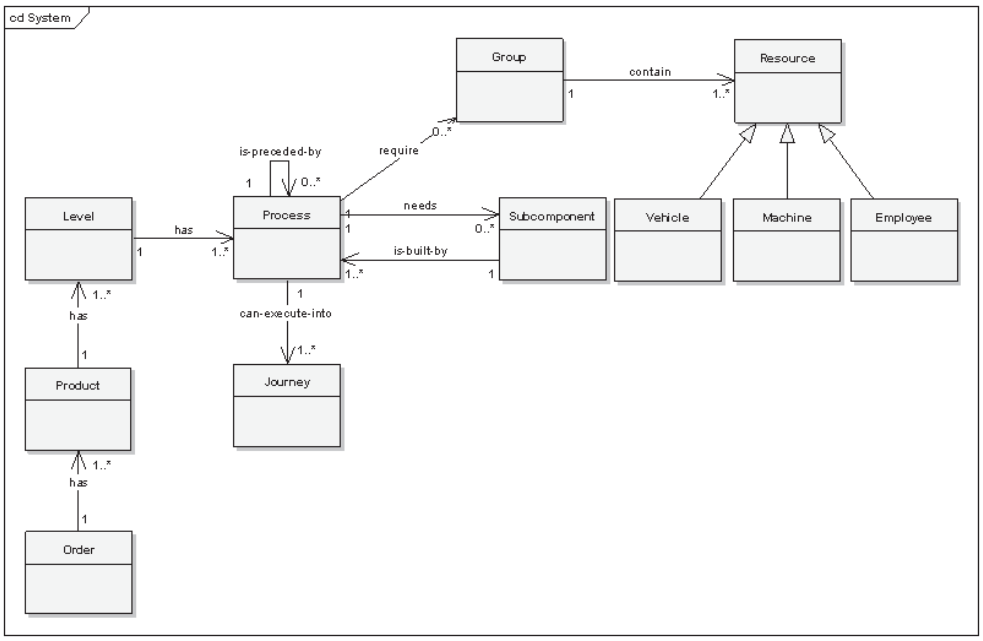

Figure 5: Domain model for scenario of GAMHSO

Formally GAHMSO is defined in figure 6. The two objectives of GAMHSO is found a set of solutions nondominated of the programming of the processes minimizing the overall time and cost.

The initial solutions are created with some heuristics. The heuristics organizes the production orders with rules. GAHMSO use three rules for creating the initial solutions. It selects the heuristic in at random. The available heuristics for creating the initial solutions are Early Due Date (EDD), Long Process Time (LPT) and Short Process Time (SPT). EDD organizes the production orders from lower to upper respect to the due date. SPT organizes the production orders from lower to upper respect to the summation of the processes time of the products. LPT is the contrary to SPT.

Once the initial solutions are created, the selection step selects solutions from the overall set of solutions. The selected solutions are named candidates. The candidates are the solutions that can be crossed for creating new solutions. The amount of candidates is defined by the candidate rate C_R.

It is very important the definition of the chromosome for the crossover step. For instance, consider the product of the figure 7. The representation of two feasible solutions could be to program the execution of processes in the order 8 - 11 - 9 - 10 - 6 - 7 - 5 - 4 - 1 - 2 - 3 and 11 - 10 - 6 - 5 - 4 - 2 - 8 - 9 - 7 - 1 - 3. The program indicates a feasible solution for programming the processes. Some processes can be executed parallel, so the order indicates the priority in the utilization of the resources. For instance, processes 8 and 9 can be executed at the same time, but in the first solution, if 8 uses a machine that 9 requires, 9 could not executed until 8 free the resource. The problem with this representation is that the crossover step could create unfeasible solutions. For example, if we split the two solutions mentioned in the middle and later we cross those solutions, we will obtain the solutions 8 - 11 - 9 - 10 - 6 - 2 - 8 - 9 - 7 - 1 - 3 and 11 - 10 - 6 - 5 - 4- 7 - 5 - 4 - 1 - 2 - 3. Obviously, those are unfeasible solutions.

```
Inputs: C_R, ρ_R
Output: A set (S) of nondominated solutions.

GAMHSO
    S = Create Initial Solutions(EDD, LPT, SPT)
    Do
        S' = Selection of Candidates (S, C_R)
        C = Crossover of Candidates (S', ρ_R)
        x = generated a integer in [0,2] with a uniform distribution.
        switch(x)
            case 0: C = TabuSearch(C)
            case 1: C = SimulatedAnnealing(C)
            case 2: C = Exchange Deterministic Algorithm(C)
        end
        S = S U C
        S = Remove Domiated Solutions(S)
    Until Condition Stop = True
End GAMHSO
```

Figure 6: The framework of GAMHSO

4 Definition of the Chromosome

The objectives of GAMSHO are the optimization of overall cost and time. It plays with the programming order of the execution of the processes. So it is very important to provide a good representation of the solutions.

Consider again the product of the figure 7. It has subparts because it is modeling the reality. But, żwhat does a subpart mean? It means that the process that contains the subpart cannot be executed until the processes that build it have been executed. In other words, there exists a precedence constrain between the process that contain the subpart and the processes

that build the subpart. If we applied this to the figure 7, we are going to obtain the tree of the figure 8. In this tree does not exist subpart, we replace the subpart for the precedence between the processes. Once the tree is ready, we need to create a chromosome that allows the representation of the programming.

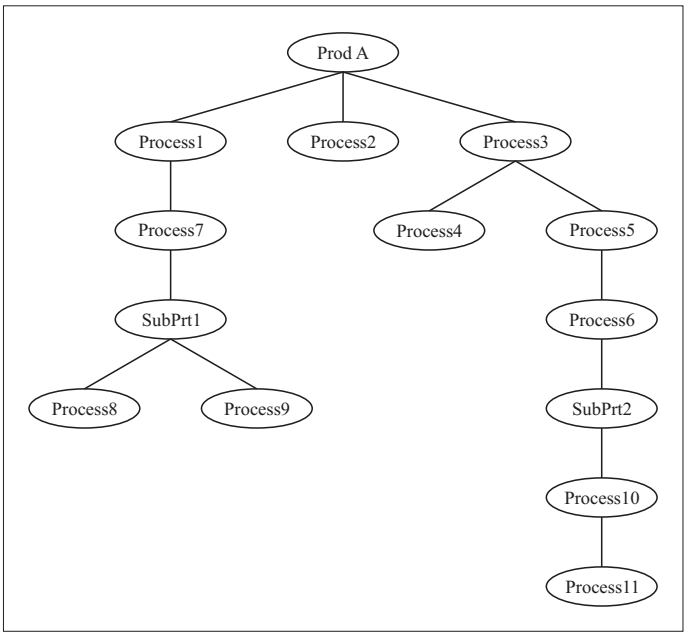

Figure 7: Tree Product A. Product A requires of the execution of the processes 1, 2 and 3. Process 1 requires the execution of the process 7. Process 7 requires the subpart1. SubPart1 requires the execution of the processes 8 and 9. Process 2 does not require processes. Process 3 require the execution of the processes 4 and 5.Process 5 requires the execution of the process 6. Process 6 requires the SubPart 2. SubPart 2 requires the execution of process 10 and process 10 requires the execution of process 11.

First we group the processes by level. It means that the time execution of a process in a superior level depends on the finalization execution time of the parents in a low level. Formally:

$$p.start\_time = max(parent_i.finalization\_time) \tag{4.1}$$

For instance, if process 5 finishes its execution in time 20 and the process 4 finishes its execution in time 40, process 3 will start its execution in time 40. On the other hand, the process 2 can be executed since time 0.

Once the levels of the processes have been identified, those are grouped in a level structure. It can be seen in figure 9. Each process knows its children in the superior level. Due to this, the chromosome for GAMHSO can be seen in figure 10. The production orders are executed from left to the right (from up to down). The products are built from left to the right (from up to down). The processes are executed from right to the left (from down to up). The processes are processed from left to the right (from up to down).

The crossover step consists in select two solutions, split in the middle and cross. It can be done by levels. An example can be seen in figure 11. The number of solution that can be crossed

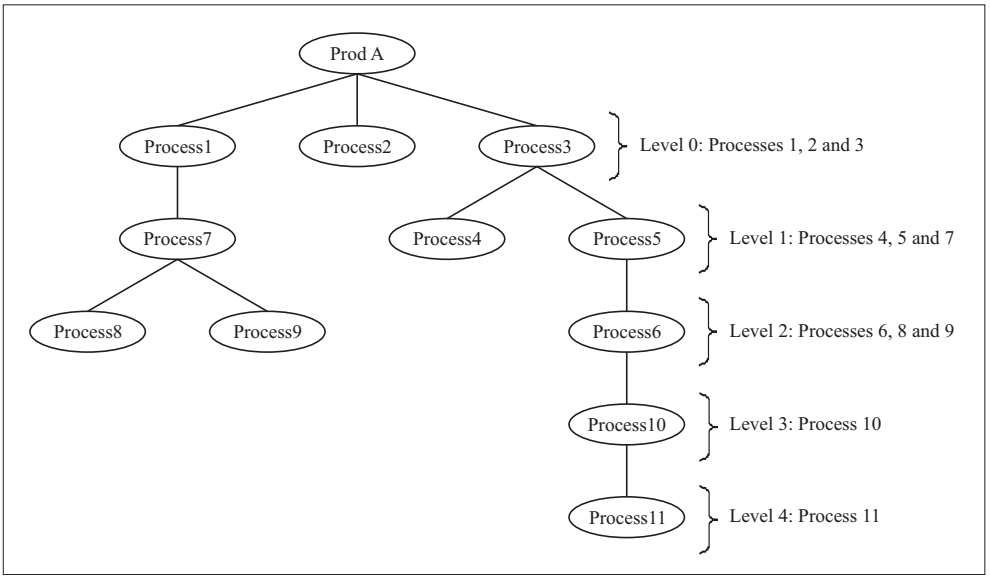

Figure 8: A view of the precedence tree of Product A group by level.

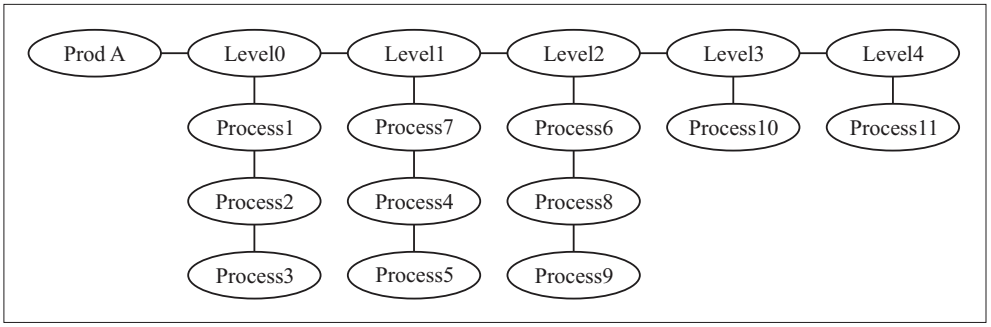

Figure 9: A representation of the Product A in a level structure.

is specify by the crossover rate (ρ_R). Once the solution is created, it is necessary to schedule the processes of the solution for obtaining the time and cost of the solution. The Programming Algorithm (PA) is an algorithm that requires a solution for programming all the processes for all the products of the production orders. It verifies is a process can be executed with three validations: First, it verifies the process precedence. Second, it verifies if the resource are available for the execution of the process. Lastly, it verifies is the process can be executed in the journey. Once a process completes its execution, it set the initial time to his children to his finalization time. PA can be seen in figure 12.

The mutation step of GAMHSO consists in the improvement of the solutions through Local Search (LS). GAMHSO works with three LS: Tabu Search (TS), Simulated Annealing (SA) and Exchange Deterministic Algorithm (EDA).

5 Experimental Settings

We tested GAMHSO in a computer AMD Turion 64, 2 GB of RAM and a Hard Disk of 120 GB.

The test consisted in build 1000 products of type A (Figure 7). The parameters were $C_R = 0.4$,

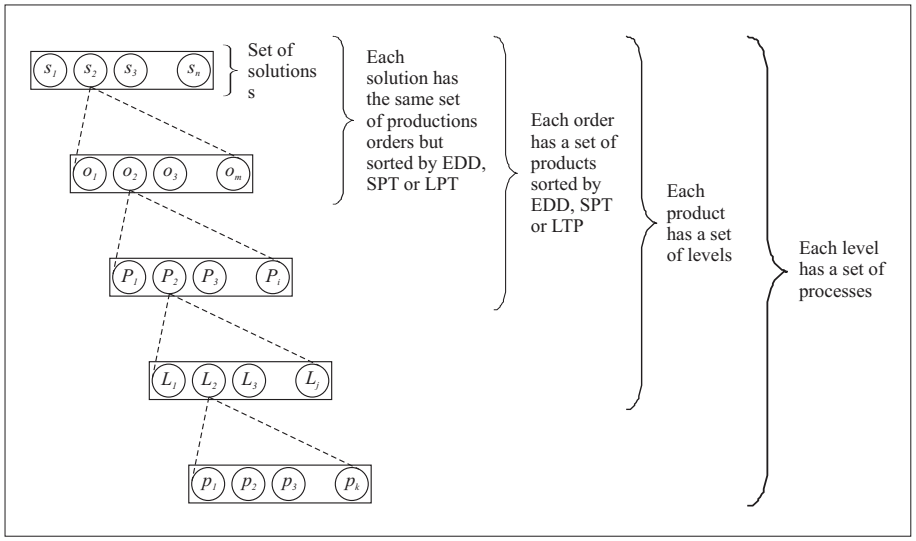

Figure 10: The Chromosome for GAMHSO.

$\rho_R = 0.5$. We tested the performance of GOMHSA with EDA, TS and SA. For making a real comparison, we use the Inverted Generational Distance (IGD) metric [15]. It is defined as follows:

Given a reference set A^*, the IGD value of a set $A \subset R^m$ is defined as:

$$IGD(A, A^*) = \frac{1}{|A^*|} \sum_{v \in A^*} \{\min_{u \in A} d(u, v)\} \tag{5.1}$$

6 Results of GAMHSO

The Pareto Fronts for each LS can be seen in figure 12. The results of IGD metric between the LSs can be seen in table 1. The running times for GAMHSO for each LS can be seen table 2.

	EDA	SA	TS
EDA	0	1137.0605	7485.2654
SA	1134.2321	0	6157.3669
TS	7367.1584	6388.229	0

Table 1: The IGD-metrics values for each LS against the rest of LS.

	Running Time in seconds
EDA	32
TS	57
SA	128

Table 2: Running times of GAMHSO with each LS.

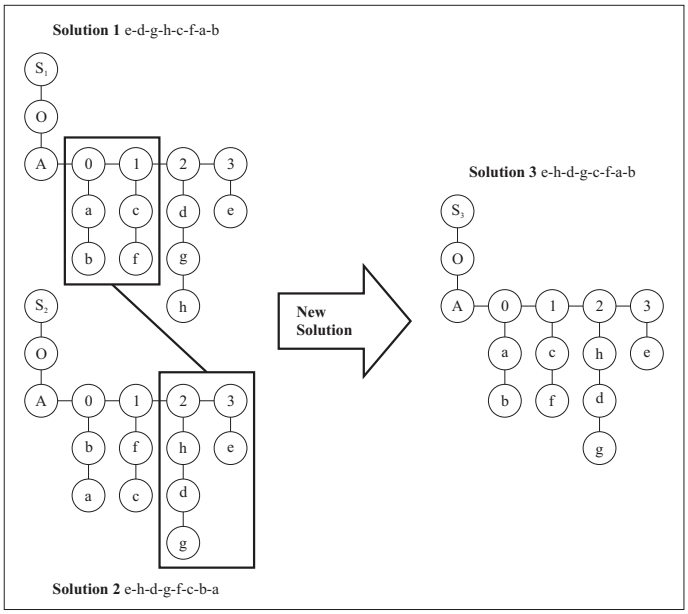

Figure 11: An example of the crossover step of GAMHSO.

7 Conclusions and Future Works

We designed a new Genetic Algorithm for Hard Scheduling Optimization (GAMHSO). It works with very difficult scenarios of productive systems. Also, we define a chromosome for GAMHSO that avoids the creation of unfeasible solutions. Due to this, it is not necessary to verify if a solution (for the crossover step) is a feasible solution. Consequently, the performance of the algorithm is satisfactory in comparison with the size of the feasible solutions space. On the other hand, we state a new Programming Algorithm (PA) for scheduling of a set of production orders. PA is a flexible algorithm that allows the incorporations of new restrictions to the processes. It allows calculate the overall time and cost of a set of production orders. We made a real comparison of the GAMHSO behavior with some local search strategies such as Exchange Deterministic Algorithm (EDA), Tabu Search (TS) and Simulated Annealing (SA). The best performance of GAMHSO was using EDA and SA. Lastly, we will investigate a new chromosome that allows the crossover between production orders.

```
Programming Algoritm(O as a set of production orders)
        Cost = 0
        Time = 0
        For Each Order o in O
                For Each Product P in o
                        For Each Level l in P
                                For Each Process p in P
                                t = p.initialTime
                                        Do
                                                While p cannot be executed in time t
                                                        t = t+1
                                                End While
                                                it = t
                                                While p can be executed in time t
                                                        t = t+1
                                                End While
                                                ft = t
                                                For all Resources r used for p. r.programTask(it,ft). Cost = Cost+r.getCost
                                        Until t = 0
                                        For all Processes Children pc of p. pc.initialTime = max(pc.initialTime,t)
                                        If  t > Time Then
                                                Time = t
                                        End If
                                End For
                        End For
                End For
        End For
        Return Time,Cost
End Programming Algorithm
```

Figure 12: The Framework of the Programming Algorithm (PA) for getting the overall cost and time of the programming of a set of production orders.

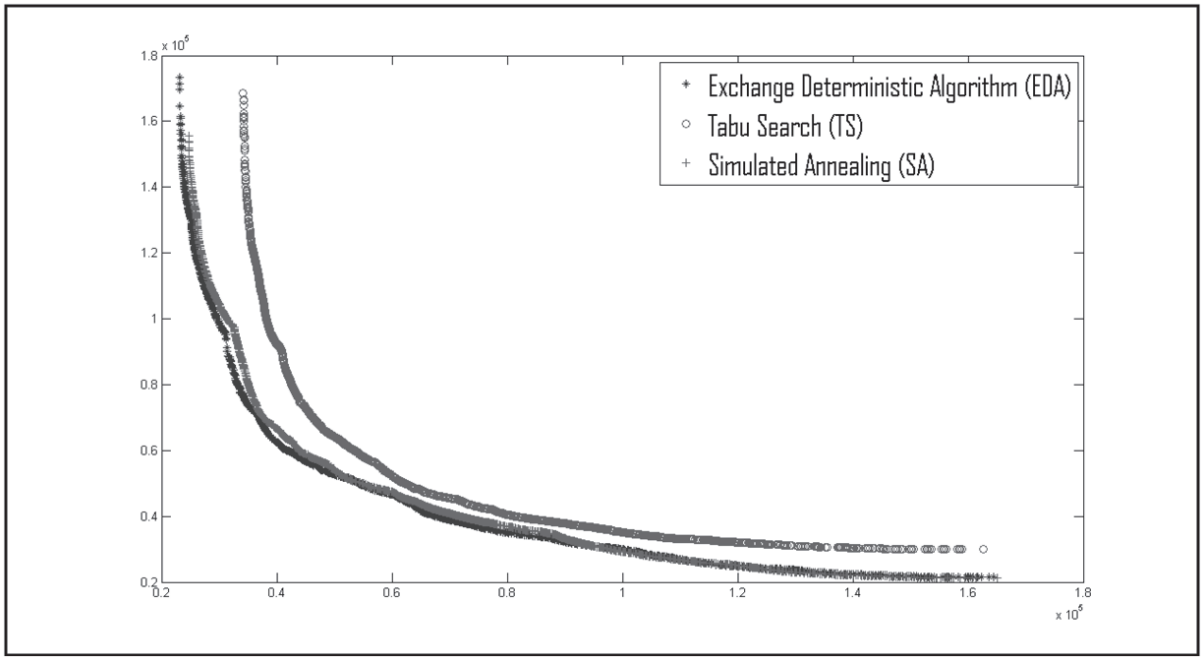

Figure 13: Pareto Fronts for GAHMSO for each LS. Notice that TS is dominated by SA and EDA. GAHMSO a similar behavior for SA and EDA

Bibliography

[1] Jingjun Zhang; Yanhong Zhang; Ruizhen Gao, "Genetic Algorithms for Optimal Design of Vehicle Suspensions", Engineering of Intelligent Systems, 2006 IEEE International Conference on , vol., no., pp.1-6, 0-0 0.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1703182&isnumber=35938

[2] Murphy, L.; Abdel-Aty-Zohdy, H.S.; Hashem-Sherif, M., "A genetic algorithm tracking model for product deployment in telecom services", Circuits and Systems, 2005. 48th Midwest Symposium on , vol., no., pp.1729-1732 Vol. 2, 7-10 Aug. 2005.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1594454&isnumber=33557

[3] Guangyuan Liu; Jingjun Zhang; Ruizhen Gao; Yang Sun, "A Coarse-Grained Genetic Algorithm for the Optimal Design of the Flexible Multi-Body Model Vehicle Suspensions Based on Skeletons Implementing", Intelligent Networks and Intelligent Systems, 2008. ICINIS '08. First International Conference on , vol., no., pp.139-142, 1-3 Nov. 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4683187&isnumber=4683146

[4] Wu Ying; Li Bin, "Job-shop scheduling using genetic algorithm", Systems, Man, and Cybernetics, 1996., IEEE International Conference on , vol.3, no., pp.1994-1999 vol.3, 14-17 Oct 1996.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=565434&isnumber=12283

[5] Orero, S.O.; Irving, M.R., "Economic dispatch of generators with prohibited operating zones: a genetic algorithm approach", Generation, Transmission and Distribution, IEE Proceedings-, vol.143, no.6, pp.529-534, Nov 1996.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=556730&isnumber=12146

[6] Zhao Man; Tan Wei; Li Xiang; Kang Lishan, "Research on Multi-project Scheduling Problem Based on Hybrid Genetic Algorithm", Computer Science and Software Engineering, 2008 International Conference on , vol.1, no., pp.390-394, 12-14 Dec. 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4721769&isnumber=4721668

[7] Xianghui Deng, "Application of Adaptive Genetic Algorithm in Inversion Analysis of Permeability Coefficients", Genetic and Evolutionary Computing, 2008. WGEC '08. Second International Conference on , vol., no., pp.61-65, 25-26 Sept. 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4637395&isnumber=4637374

[8] Yanrong Hu; Yang, S.X.; Li-Zhong Xu; Meng, Q.-H., "A Knowledge Based Genetic Algorithm for Path Planning in Unstructured Mobile Robot Environments", Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on , vol., no., pp.767-772, 22-26 Aug. 2004.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1521879&isnumber=32545

[9] Siu, N.; Elghoneimy, E.; Yunli Wang; Gruver, W.A.; Fleetwood, M.; Kotak, D.B., "Rough mill component scheduling: heuristic search versus genetic algorithms" Systems, Man and Cybernetics, 2004 IEEE International Conference on , vol.5, no., pp. 4226-4231 vol.5, 10-13 Oct. 2004.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1401194&isnumber=30426

[10] Lee, K.Y.; Mohamed, P.S., "A real-coded genetic algorithm involving a hybrid crossover method for power plant control system design", Evolutionary Computation, 2002. CEC '02.

Proceedings of the 2002 Congress on , vol.2, no., pp.1069-1074, 2002.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10043914&isnumber=21687

[11] Pepper, J.W.; Golden, B.L.; Wasil, E.A., "Solving the traveling salesman problem with annealing-based heuristics: a computational study", Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.32, no.1, pp.72-77, Jan 2002.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=995530&isnumber=21479

[12] Blesa, M.J.; Hernandez, L.; Xhafa, F., "Parallel skeletons for tabu search method", Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference on , vol., no., pp.23-28, 2001.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=934797&isnumber=20222

[13] Rose, J.; Klebsch, W.; Wolf, J., "Temperature measurement and equilibrium dynamics of simulated annealing placements", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.9, no.3, pp.253-259, Mar 1990.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=46801&isnumber=1771

[14] Niño, E.;Ardila, J. , Algoritmo Basado en Automatas Finitos Deterministas para la obtención de óptimos globales en problemas de naturaleza combinatoria. Revista de Ingeniería y Desarrollo. No 25. pp 100 - 114. ISSN 0122 - 3461.

[15] Minzhong Liu; Xiufen Zou; Yu Chen; Zhijian Wu, "Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances", Evolutionary Computation, 2009. CEC '09. IEEE Congress on , vol., no., pp.2913-2918, 18-21 May 2009.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4983309&isnumber=4982922

[16] H. Li and J.D. Landa-Silva, Evolutionary Multi-objective Simulated Annealing with Adaptive and Competitive Search Direction, Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), IEEE Press, pp. 3310-3317, 01-06 June, 2008, Hong Kong.