IJMSOR Vol. 1, Issue. 1
January - June, 2016

A BI-Criteria Optimization Model for Parallel Machine Scheduling: Game Theoretic vs Genetic Algorithms

Modelo de Optimizacion Bi-Objetivo para la Programacion de Maquinas en Paralelo: Teoria de Juegos Vs Algoritmos Geneticos

dx.doi.org/10.17981/ijmsor.01.01.03

Research Article - Reception Date: February 20, 2016 - Acceptance Date: March 18, 2016

Diana G. Ramirez-Rios
Rensselaer Polytechnic Institute
ramird2@rpi.edu, New York, United States

Claudia Marcela Rodriguez Pinto
Transport for NSW
rod.claudiamarcela@gmail.com, Sydney, Australia

Javier Visbal Martinez
Universidad del Norte
javierv@uninorte.edu.co, Barranquilla, Colombia

Fabio Andrés Monroy Silvera
Fundación Centro de Investigación en Modelación Empresarial del Caribe FCIMEC
fabiomonroy80@gmail.com, Barranquilla, Colombia

Jair José De la Cruz Hernández
Universidad del Norte
jair.delacruz@gmail.com, Barranquilla, Colombia

Yezid Donoso Meisel
Universidad de los Andes
ydonoso@uniandes.edu.co, Bogotá, Colombia

Carlos D. Paternina Arboleda
Universidad del Norte
cpaterni@uninorte.edu.co, Barranquilla, Colombia

To reference this paper:
D. G. Ramirez-Rios, C. M. Rodriguez Pinto, J. Visbal Martinez, F. A. Monroy Silvera, J. J. De la Cruz Hernández, Y. Donoso Meisel and C. D. Paternina Arboleda "A Bi-Criteria Optimization Model for Parallel Machine Scheduling: Game Theoretic vs Genetic Algorithms", *IJMSOR*, vol. 1, no. 1, pp. 20-30, 2016. DOI: dx.doi.org/10.17981/ijmsor.01.01.03

Abstract-- This paper considers a problem for scheduling jobs on two identical parallel machines, the aim was to minimize two criteria in particular, makespan and total flow time. In order to solve this problem, two approaches were considered. A mechanism was proposed as an approach to solve this type of problem with a setting of a 2-player non-cooperative game, under the framework of a 2x2 non-sum zero matrix; each player looking after one of the criteria suggested in the scheduling problem. On the other hand, a Genetic Algorithm, known as Strength Pareto Evolutionary Algorithm (SPEA), was applied to the problem. The comparison between both approaches suggests a complementarity among rational agents approach models and machine enforced solution approaches. The resulting Pareto Front set of points were plotted and curves were compared, showing promising results for game theoretic applications to scheduling under multiple objectives.

Keywords- identical parallel machines, makespan, total flow time, non-cooperative game, SPEA, Pareto Front.

Resumen-- Este articulo contempla el problema de programación de la producción en una configuracion de maquinas en paralelo con el objetivo de minimizar dos criterios en particular: el lapso y el tiempo total de flujo. En este problema en particular, el incremento de uno de estos objetivos resulta en la reduccion del otro, por lo que se propone su solucion bajo enfoques metaheuristicos. Dos tipos de algoritmos fueron considerados: uno basado en la teoria de juegos y el otro en los algoritmos genéticos. Para el primero se diseña un mecanismo de juego no cooperativo entre dos jugadores, en donde cada jugador busca optimizar cada criterio de programación de las máquinas. Para el segundo enfoque se implementa el algortimo genético SPEA, en donde se seleccionan aquellas soluciones dominantes en ambos objetivos. Resultados de ambos enfoques resultan en un Frente de Pareto, las cuales representan las soluciones dominantes para ambos objetivos. Estos resultados demuestran que ambos enfoques son complementarios: SPEA arroja resultados que cubren todo el frente de Pareto, mientras que el algoritmo de Juegos No Cooperativo indica la programación mas conveniente para cada agente en particular.

Palabras clave- máquinas paralelas idénticas, tiempo de flujo total, makespan, juego no cooperativo, SPEA, frente de pareto

Address correspondence to Diana G. Ramirez-Rios Graduate Research Assistant, Department of Civil and Environmental Engineering Rensselaer Polytechnic Institute, 110 Eighth Street, Troy, New York 12180, USA Email: ramird2@rpi.edu

I. Introduction

Production Scheduling is a wide and extensive line of research, which has been developing over the century and continues to bring solutions to applications in the industry. Its complexity is also evident, given that everyday hundreds of tasks are produced in a production line and a diversity of settings are considered for less and more specialized products. Machine learning is, as well, a line of research that has considerable applicability and is well-known for approaching complex problems, such as the ones mentioned below. When tackling multiple objectives, decisions become even more complex and decisions to complete several criteria usually involve the decision of single experienced person in a facility. Several heuristic approaches have been employed for multi-objective optimization in scheduling problems [1], [2], [3], [4], but metaheuristics are considered to be the best in terms of efficiency and robustness [5], [6], [7] and [8].

Game theoretic approaches, on the other hand, allow negotiation processes to be mathematically modeled through a simplified structure, establishing the interactions between each player's options to accept conditions set up by them. Once an agreement between players is achieved, it is said to have come to equilibrium. Traditional negotiation theory analyzes single criterion games in which each agent appraises one single objective, hindering more realistic negotiation processes that may involve more than one criterion. When analyzing such processes an additional difficulty arises: The pay-off matrix within the game can not be considered zero-sum, thus the costs implied for these players are given in a different scale, depending on the criterion assumed for each player. At the same time the equation must consider how each agent is affected by the decision taken.

When making decisions in a firm, there are at least two conflicting objectives faced by the person in charge. On one hand, the production must be completed as fast as possible, while the customer representatives strive to get their own client's job done as fast as possible. While maximizing overall utility seems like a reasonable answer it is not likely to represent the real outcome since efficiency implies sacrificing client's satisfaction, which may lead to the loss of market and prestige. Agent-based models, such as the game theoretic approaches, can more realistically represent a decision maker in a facility, which satisfies the overall efficiency of the production line versus satisfying the customer's needs.

Extant Work

Literature reviews on scheduling theory, game theory and multi-criteria scheduling theory, has shown the area has been broadly studied. Specifically in parallel machine scheduling under multiple objectives, heuristic approaches such as the GAP/EDD algorithm were combined with metaheuristics, such as Tabu Search, to solve the bicriteria Cmax and Max T problem [1]. In the recent years, new metaheuristic approaches have been proposed to solve the different multi-objective problems in parallel machine scheduling, such as the Nondominated Sorting Genetic Algorithm (NSGA) [9], the Particle Swarm Optimization Problem (PSO) [10], among others. On the other hand, agent-based approaches have also been approached for machine scheduling. One of the most important approaches of game theory in scheduling have been obtained in the area of mechanism design [11], Auction Theoretic Modeling [12], Worst case Equilibra [13], and Kutanoglu's incentive design scheduling. One of the contributions is observed in Even-Dar's work [14], which contemplated a setting composed of n jobs with an associated agent, over m machines. Jobs were allowed to select a machine to minimize their own cost, this cost was determined by the load on the machine, which was the sum of the weights of the jobs running on it. It was stated that at least one job was willing to change to another machine, until Nash Equilibrium was reached.

More specifically, in the area of computer science, a concept known as agent-based simulation has been of great influence to today's research on game theoretic approaches to scheduling. Archer [13] and, Kutanoglu and Wu's work [12] stated that jobs are considered agents; while Nisan [15] also states the usefulness of a mechanism design for selfish agents. Recent work has been applied to the allocation of deteriorating jobs in parallel machines [16].

Decision making situations always involve a component of strategy, which has been traditionally analysed in Game Theory as well as the Economic Theory, but, as observed, has also been adopted in Computer Science and Operations Research studies as well, motivated by the rising theories on auctions and internet-based transactions. From the application perspective on Production Scheduling, it has been found that its focus may be quite narrow, yet, this is indeed one of the approaches that explain this type of optimization problem in a decentralized perspective, where the centralized decision traditionally made is partially replaced by decisions and actions taken by agents. These are assumed to act rationally, based on their own self-interest and it is assumed that this selfish behaviour leads to some sort of system equilibrium, but it can also be true that the design of this decentralized setting may lead behaviour of selfish agents to act towards global system performance [17].

This Work.

This research was performed in a production programming environment sketch in order to obtain solutions for scheduling on a particular problem un-

der parallel machine configuration: ($Pm||C_{max}$, $\sum C_i$), where C_{max} represents maximum completion time and $\sum C_i$, total flow time. Usually, the schedule for this type of configuration results from the arrangement of each one of the n jobs, assigned to m number of machines; this depends on the availability of the machine (First Available Machine), or; in the case of non-identical machines, it can depend on the velocity of the machine (Fastest Machine First), but this was not considered in this research. To minimize flow time, an algorithm that constructs a list in order of non-decreasing processing time (SPT) is widely used. On the other hand, in order to minimize makespan (C_{max}), a list in the order of decreasing processing time is constructed (LPT).

The game theoretic approach proposes a mechanism that considered two types of agents, an agent that represented the jobs (job agent) and the system controller: agent 0, which regulated the jobs' allocation on the machines. Each agent responded to its corresponding criteria, agent 0 aimed to minimize the makespan, while each job agent, the total flow time. A 2-player non-cooperative game was designed, where each job agent played against the controlling agent, whose decisions: to move or not, were based on some associated costs, that were measured in units of time and depended on each of the criteria measured [18]. The metaheuristic approach, on the other hand, was proposes a nice configuration of a scheduling allocation represented as chromosomes. These chromosomes interact with each other and reproduce other good and less than good solutions. A fitness function selects the fittest, mutation and reproduction functions provide another generation of solutions. On the long run, these solutions lead to globally optimal solutions to multi-objective problems, such as the one shown below. Among the different algorithms developed, the Strength Pareto Evolutionary Algorithm (SPEA), proposed by [19], proved to be most efficient when tackling multiple objectives in different engineering applications [20], [21].

According to Game Theory a simulation based on agents mechanism was designed in order to achieve a set of efficient schedules whose outcomes were sketched in a C_{max} vs. $\sum C_i$ graph that later transformed into a Pareto Front of efficient solutions for our problem. When analysing the trade-offs and movements among agents it was observed that in the long run they sometimes tend to choose a set of strategies or have a tendency to choose; when this happens, the model gets to an equilibrium in the long run, it can be Nash Equilibrium or Mixed Equilibrium, in the case, it involves choosing a set of strategies with a certain probability. On the other hand, metaheuristic approaches do not take into account human behaviour, but may lead to more efficient solutions.

Hence, this investigation is composed by the integration of both of these topics of study and intends to bring out solutions that complement each other. This paper is organized as follows: In section 2, the proposed game theoretic model is described; in section 3 a numerical example is provided where results to game theoretic model were analysed; in section 4, the metaheuristic SPEA is described; in section 5, multiple scenarios are provided and results for both approaches are obtained; ending in section 6 where the conclusions and further analysis are provided.

II. Proposed Game Theoretic Model.

Consider once again the scheduling problem configuration for identical parallel machines, where the objectives were makespan and total completion time ($Pm||C_{max}$, $\sum C_i$). A proposed model developed by Ramirez-Rios, Rodriguez and Paternina-Arboleda [18], involve a combination of machine learning with repeated non-zero sum games. Since both variables are conflictive: C_{max} and $\sum C_i$, a scheduling problem considering both of these criteria was solved by using the mechanism design approach using incentives and penalties in the agent's payoffs. During the simulation, each job agent plays a game with a central agent in a sequential order. Each type of agent has an objective that can interact rationally with its opponent and lead to overall system efficiency, in order to meet both criteria through tradeoffs within a payoff matrix.

A. Definition of the Game

Consider a non-cooperative repeated game of two players, where player one represents a job agent and player 2, agent 0. The job agent represents a product or customer and the agent 0 represents the production plant supervisor or manager. Each one has two strategies that correspond to their own interests. For the job agent these are to seize his job first, which leads to an improvement of his *associated partial flow time*, and for agent 0, to reach a better C_{max}. Regarding the system mechanism to reduce not only C_{max} but also total flow time, it is assumed that each player within the game will play by assuming some costs imposed by the system, in order to allow the conditions acquainted to be reached. A description of the strategies for job agent and agent 0 are S_1 and S_2, respectively:

Strategies for job agent, S_1 = {A, B}:
A= Stay in the current position (time slot).
B= Move to the previous available position.
Strategies for agent 0, S_2 = {C, D}:
C= Leave job on the current machine.
D= Move the job to another machine.

The payoffs for the players have been represented as costs, which are expressed in units of time, which give a proxy to associated costs that players could incur. The cost functions are given in terms of the makespan and flow time. Each pair of strategies involves a cost function, which has two components that illustrate the performance cost and the incentive for each player when choosing the strategies des-

cribed above, providing what is known as the outcome cost (OC). There are four costs for each agent, which are further evaluated in a non-zero sum game matrix.

The cost function for the job agents is provided in Equation 1. The first term corresponds to the total completion or flow time of the system (ΣC_i). The second term is known as a payment incentive for the job agent in order to encourage it to sacrifice some time in their production, guaranteeing overall efficiency.

$$Job_i\ Cost = TotFT - \beta_i(F_m) \quad (1)$$

This incentive is defined as the fraction of the total cost represented for the job agent in the production line. If it is has a longer waiting time, this cost will be lower than if it were in a better position. Equation 2 provides how this function was calculated.

$$\beta_i = \frac{Pi(job\_selected)}{PartialFT_i(on\_machine\_selected\_by\_agent0)} \quad (2)$$

The cost function for agent 0 is provided in Equation 3. The first term in the equation (C_{max}) is referred to as the maximum completion time measured in the system. The second term is similar to the opportunity cost given to the controlling agent when deciding to move a job or not. It includes a β_0, which is a value based on the weight that the completion time of the current machine (the one being analyzed) has over the other machine. A similar calculation is derived as in Equation 2. The second part of this term (C_i) of the equation corresponds to the completion time of the job agent in its current position.

$$Agent0\ Cost = C_{max} + \beta 0 C_i \quad (3)$$

B. General Structure of the Model

PROPOSED MODEL - SCHEDULING GAME MECHANISM
1 Load Balancing for the initial schedule generated. 2 Do 3 Ai randomly chosen from the machine with the highest load, current machine. 4 A second machine is randomly chosen. 5 Ai and A0 choses among strategies. 6 According to payoff matrix, a strategy is chosen. 7 If rand() < prob_move Then 8 Ai is reallocated in the second machine at the end of the schedule. 9 Else 10 Movement corresponding to the chosen strategy results in a new schecule. 11 End if 12 Schedule is registered and so are the values for Cmax and Flow Time. 13 Loop Until "System Reaches Equilibrium" 14 Pareto Front is generated based on dominance between the registered values for Cmax and Flow Time

The "Scheduling Game" Algorithm has two termination conditions providing an option to the user: by using a number of iterations or by reaching equilibrium in the system. In order to establish the number of iterations permitted, a limit had to be determined. This limit depends on the number of jobs (n) and the number of parallel machines (m) in the system. Given that each iteration allows one movement, then the limit was determined by: $n * n * (m - 1)$. Given this limit, computational complexity for this algorithm is approximated to $O[LOG(N^2*(M-1))]$.

III. A Numerical Example: Solving the Scheduling Game

A. Numerical Example

Let's consider a bi-criteria scheduling problem composed of 10 jobs, Ji={8,46,30,19,4,36,21,23,6,17} that need to be allocated in 2 parallel machines in order to minimize makespan (C_{max}) and total flow time (ΣCj). There is no information on release dates or setups considered and preemption is not allowed. This problem is solved through the proposed game theoretic model to illustrate its applicability.

The game theoretic approach described previously starts by generating an initial solution, as shown in Table 1, based on Load Balancing Heuristic. This heuristic uses the rule First Come, First Served, yet the allocation of jobs in machines is done to guarantee balance in terms of completion times.

Table 1. Initial Allocation of jobs in 2 machines by Load Balancing

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	Partial Fi
1	8	8	8	2	46	46	46
3	30	38	46	5	4	50	96
4	19	57	103	6	36	86	182
7	21	78	181	9	6	92	274
8	23	101	282	10	17	109	383

On each iteration, a random selection occurs in the machine with the maximum makespan and a job is selected to be the job agent for the iteration. In this example, job 5 was selected as the job agent. After choosing the job agent, payoffs for both the controlling agent and the job agent are calculated according to the equations explained above. These payoffs as observed depend on the position of the job and the machine it will be allocated to. As observed in table 2, the job agent 5, has only two strategies: (A) Stay in its actual position (timeslot 2), or (B) move to a previous position (timeslot 1). Likewise, the controlling agent, has to choose between two alternatives: (C) the choice to move the job to machine 1 or (D) leave it in machine 2.

TABLE 2. JOB AGENT SELECTED FOR THE GAME (A5) AND THE POSSIBLE MOVEMENTS IT CAN MAKE

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	Partial Fi
1	8	8	8	2	46	46	46
3	30	38	46	5	4	50	96
4	19	57	103	6	36	86	182
7	21	78	181	9	6	92	274
8	23	101	282	10	17	109	383

Table 3 presents the resulting payoff matrix for all possible scenarios on the players' decisions, which make the first iteration of the game designed. When solving the matrix, each player tries to minimize their corresponding cost. Given that, in this case, it reached Nash Equilibrium, it was observed that job 5 was preferred in the current machine but in a previous position.

TABLE 3. PAYOFF MATRIX FOR JOB AGENT 5 IN THE FIRST ITERATION.

		AGENT 0			
	ROUND 1	C		D	
JOB AGENT 5	A	649,04	252,42	629,40	610,00
	B	332,52	163,50	454,16	1.228,50

The allocation of jobs, as observed in Table 4, change according to Nash Equilibrium, which results in a makespan, C_{max}, of 109 and a total flow time, ΣC_i, of 623 units of time.

TABLE 4. PROPOSED SCHEDULE FOR FIRST ITERATION.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	Partial Fi
1	8	8	8	2	4	4	4
3	30	38	46	5	46	50	54
4	19	57	103	6	36	86	140
7	21	78	181	9	6	92	232
8	23	101	282	10	17	109	341

B. Results from Numerical Example

After 127 iterations, the algorithm reached an equilibrium, which implies that job agents in the current machine were no longer motivated to move to other positions. The equilibrium reached corresponds to a mixed equilibrium, that is, on the long run, no strategy is dominated but a percentage of 62.7% was given to choosing strategy AD. The results to all these schedules were plotted as observed in figure 1, where the Pareto Chart is constructed and table 5 shows the values for the Strict Pareto Solutions.

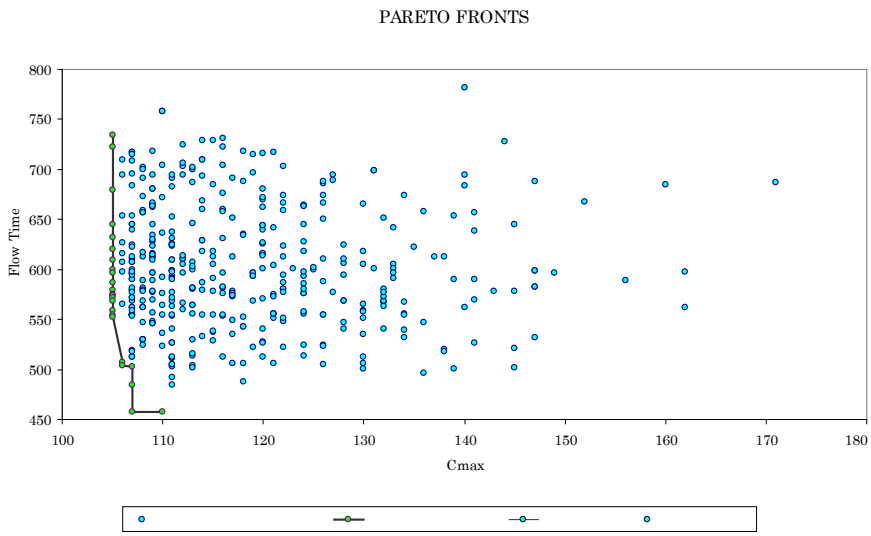

Fig. 1. Pareto Front obtained from the Scheduling Game for this example.

Table 5. Strict Pareto Solutions of the Scheduling Game for this example.

PARETO SOLUTIONS	
Cmax	Flow Time
105	552
106	504
107	458

Pareto Front shown displays the solutions that are considered non-dominated by both types of agents in the Scheduling Game. These solutions represent the best allocation of jobs for both job agents and the central agent, after their interaction on the simulation game. The solutions shown are clearly conservative, due to the nature of game theory, where solutions might avoid getting a better one; which is evidently selfish. It is important to note that the values are also a result of some randomness that took place during the game.

The values that have constructed the Pareto Front are efficient schedules that have been generated throughout the game, for the reported schedules. All these solution sets are strict Pareto solutions because there is at least one strict inequality between them, supporting the classical multi-criteria decision making theory (MCDM). In table 6, the schedules for the strict Pareto solutions are specified with their values, Z.

Table 6. Schedules generated that belong to the strict Pareto Front solution set.

SCHEDULE	MACHINE 1	MACHINE 2	Z(Cmax, ∑Cj)
1	{J1, J3, J7, J2}	{J5, J8, J10, J4, J6, J9}	(105, 552)
2	{J8, J10, J3, J6}	{J9, J7, J5, J1, J4, J2}	(106, 504)
3	{J5, J1, J1, J3, J2}	{J9, J10, J7, J8, J6}	(107, 458)

C. Comparison of the results to the Classical Multi-criteria Techniques

A solution to this same scheduling problem was obtained by Gupta and Ho in their paper "Minimizing Flow Time subject to Optimal Makespan on Two Identical Parallel Machines", the difference being the fact that their solution approach was based on the hierarchical rule in MCDM. This procedure is known as the lexicographic search base algorithm [21], [23]. Other results were obtained from the LEKIN SCHEDULING SYSTEM Software, using the rules already known in scheduling like SPT, LPT, FCFS (first come first served) and a heuristic used by this software to solve multi-criteria scheduling problem, known as the Shifting Bottleneck Heuristic.

Table 7. Results from the Classical Multi-criteria Techniques

RESULTS FROM OTHER ALGORITHMS		
ALGORITHM/HEURISTIC	Cmax	Flow Time
General SB Routine/sumC	110	458
LPT	106	800
SPT	120	458
FCFS	109	665
Lexicographical Search Base*	105	460

Table 7 shows the solutions for heuristics in multi-criteria scheduling. These solutions were compared to the ones obtained from the Scheduling Game proposed. Given the set of solutions for the Scheduling Game as P^{SG} and the set of solutions found by MCS as P^{MC}, an interpretation can be established with respect to the Pareto points graphed in figure 2, where the values seem to be close enough to the ones found by classical multi-criteria techniques, as shown.

P^{MC} = {**(110, 458)**; (106, 800); (120, 458); (109, 665); **(105, 460)**}

P^{SG} = {**(105, 552)**; **(106, 504)**; **(107, 458)**}

When comparing the solutions to the results for the Lexicographical Search Base algorithm and the Shifting Bottleneck Heuristic, the percent deviations indicate a level of the efficiency within the model, since they are close together and the improvement for one value, worsens the improvement for the other value.

	LSB Algoritm		SB Heuristic	
Schedule	**Flow Time**	**Cmax**	**Flow Time**	**Cmax**
1	20.00%	0.00%	20.52%	-4.55%
2	9.57%	0.95%	10.04%	-3.64%
3	-0.43%	1.90%	0.00%	-2.73%

It is observed that the results obtained using the game theoretic approach can be used as alternate schedules for certain situations where it is impossible to reach an optimal one. Basically, these solutions are rather conservative but they are flexible because they can adapt towards many situations that can become conflictive.

IV. A meta-heuristic approach: Using Evolutionary Algorithms

It has been known in literature that the many approximations that meta-heuristics have given

to solutions of different types of scheduling problems are considered NP-Hard. Moreover, scheduling problems considering multiple objectives are considerably complex, for which, most of them are NP-Hard. For these problems, which cannot be solved in polynomial computational time, solutions are approximations that can be obtained through the use of meta-heuristics and heuristics.

During this investigation, an evolutionary algorithm known as SPEA, Strength Pareto Evolutionary Algorithm, was implemented as another approach to solve the scheduling problem studied. This approach, in spite of the game theoretic one, generates elitist solutions due to the nature of the algorithm.

A. SPEA- Strength Pareto Evolutionary Algorithm

The general structure of SPEA is shown below:

SPEA - Strength Pareto Evolutionary Algortitm

```
1   Initial Population
2   For t=1 To Generations
3       Select Non-Dominated Solutions (P´)
4       While N > N´ Do
5           Clustering
6       End while
7       Fitness P and P´
8       Selection
9       Crossover
10      Mutation
11      Update Population
12  End For
```

B. Chromosome Representation

Evolutionary algorithms use the form of a chromosome to represent a solution to a given problem, which mainly contains genetic information that can be shown as a string of characters, vectors or matrices. Each unit of the chromosome is denominated an allele and the structure of the chromosome can be represented by real numbers, integers or binary digits. The chromosome representation used during this investigation was based on the sequence of jobs in each machine. Franca, et al. [24] developed this type of chromosomal representation for single machine problems with sequence dependent setup times, with alleles assuming integer values from 1 to n, where n is the number of jobs. For the specific problem studied, the alleles for each machine are separated by a character that contains the letter M and the corresponding number of the machine. For example:

M0 4 9 6 **M1** 2 8 5 10 …
M1
M2
Job 4, Job 9, Job 6
Job 2, Job 8, Job 5, Job 10 …..

Fig. 2. Chromosome Representation for parallel machine problems.

C. Selection, Crossover and Mutation

Every type of evolutionary algorithm contains the main functions such as selection, crossover and mutation. The selection function uses a method to select a group within the population in order to perform the operations of crossover and mutation. The selection function used for this problem in particular is based on a binary tournament selection. The crossover operator used for this type of chromosomes is the Uniform order Crossover function, which is based on randomly selecting a portion of alleles from each parent and copying it to the offspring. In the example shown on figure 3, the shaded portions represent the alleles selected for Offspring A, in order of appearance the portions are selected and they are randomly placed one of the machines of the Offspring A. If a job is already repeated, the portion goes to Offspring B. The non-shaded portion of the chromosomes is placed in Offspring B in the same way that it happened to Offspring A.

Parent A: **M0** 4 9 6 **M1** 2 8 5 10 **M2** 3 1 7
Parent B: **M0** 1 3 **M1** 10 4 7 2 **M2** 8 9 6 5
Offspring A: **M0** 4 3 8 9 **M1** 6 2 **M2** 10 5 1
Offspring B: **M0** 7 3 5 **M1** 9 4 8 **M2** 1 2 10 6

Fig. 3. Uniform order crossover.

The mutation operator is based on the swapping of any two jobs chosen randomly from different machines in the chromosome. This way, the operator assures the reallocation of two different jobs, because the purpose of it is to generate new search spaces in order to avoid the premature convergence to local optima.

D. Fitness and Clustering

SPEA [26] uses the fitness and clustering functions. The fitness function is applied to both external (P') and current (P) population members and it is used as a measure of how well fitted each of the population members are with respect to each other. For the population P', a strength is determined for each member *i* by a probability of how many members in the current population are dominated by it with respect to the total:

$$S_i = \frac{n_i}{N+1} \qquad (4)$$

For the population P, the fitness is determined by the sum of all the S_i of the external population members which weakly dominate j:

$$F_j = 1 + \sum S_i \qquad (5)$$

The clustering algorithm reduces the size of the external population P' of size $\bar{N}$ to $\bar{N}'$ by calculating the average Euclidean distance:

$$d_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1,\, j \in C_2} d(i, j) \qquad (6)$$

These distances, known as cluster-distances, are computed and the clusters with the minimum cluster-distance are combined together to form the bigger cluster. This continues until the number of clusters in the external population is reduced to $\bar{N}$.

E. Parameter Selection

One of the crucial aspects in every implementation of the evolutionary algorithms is the selection of parameters. Gutierrez and Mejia [21] considered two combinations of parameters: the first one, with a small size of the population and high probabilities of mutation and crossover; and the second one, with a large size of the population but low probabilities of mutation and crossover. Other authors like [27] have concluded that the optimum combination varies from problem to problem. The parameters defined in SPEA are the number of generations, size of the population, size of the external population, probability of mutation and probability of crossover. The values for these parameters were determined experimentally, after considering the different levels for each parameter and depending on value that gave the lowest flow time in each run. The values chosen were:

Number of generations: 2000.
Size of the population: 50000.
Size of the external population: 10.
Probability of mutation: 15%.
Probability of crossover: 70%.

VI. Simulation Results

The interest in obtaining results through the use of a meta-heuristic like SPEA, is to compare both approximated solutions and observe the difference of the elitist approach with the conservative one. Since both methods generate Pareto front set of points, solutions cannot be based on any particular point, but on the whole Pareto front. The spacing and the distance between Pareto Fronts (generational distance) are considered the most important variables to measure in these experiments.

The instances used in the computational experiments were randomly generated and the ranges used for n and m were [25], [26], [15] and [13], [28], [29] respectively. The processing times were generated following a discrete uniform distribution DU(1,50) and 20 replications were considered for each configuration of the number of jobs and machines, generating a total of 120 runs.

For each configuration of jobs and machines, a Pareto front was constructed using both approaches. Figures 4, 5, 6, 7, 8 and 9 show the comparison of the strict Pareto Fronts.

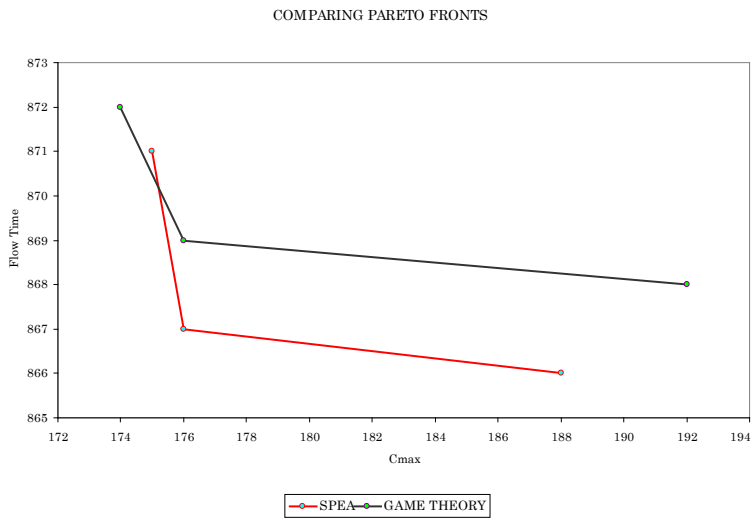

Fig. 4. Comparison of strict Pareto fronts for 10 jobs and 2 machines.

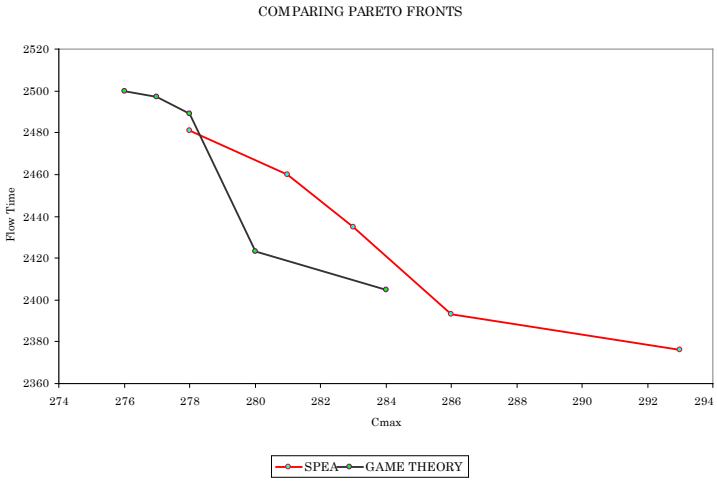

Fig. 5. Comparison of strict Pareto fronts for 20 jobs and 2 machines.

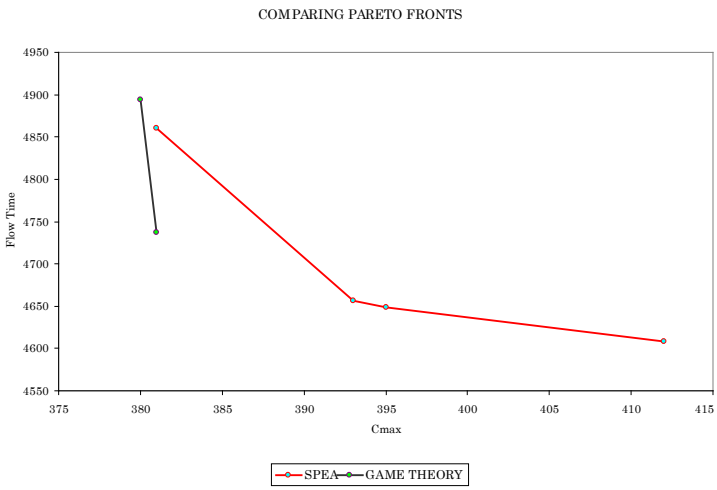

Fig. 6. Comparison of strict Pareto fronts for 30 jobs and 2 machines.

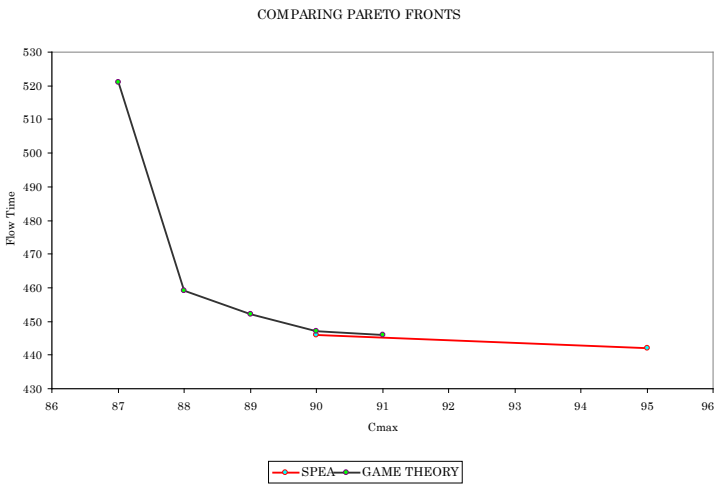

Fig. 7. Comparison of strict Pareto fronts for 10 jobs and 3 machines.

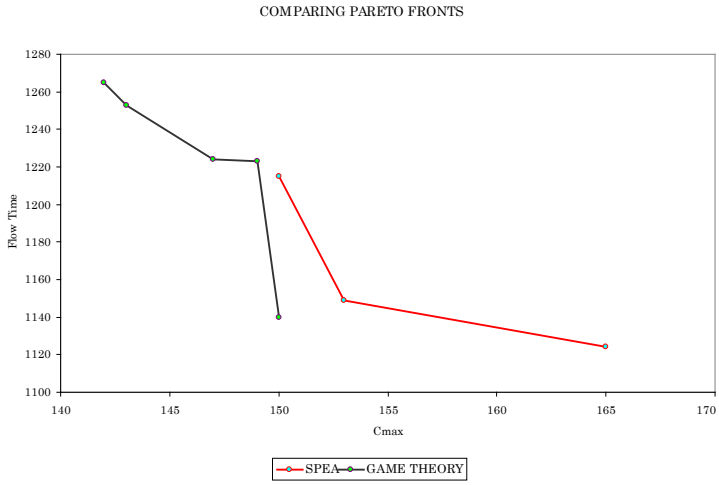

Fig. 8. Comparison of strict Pareto fronts for 20 jobs and 3 machines.

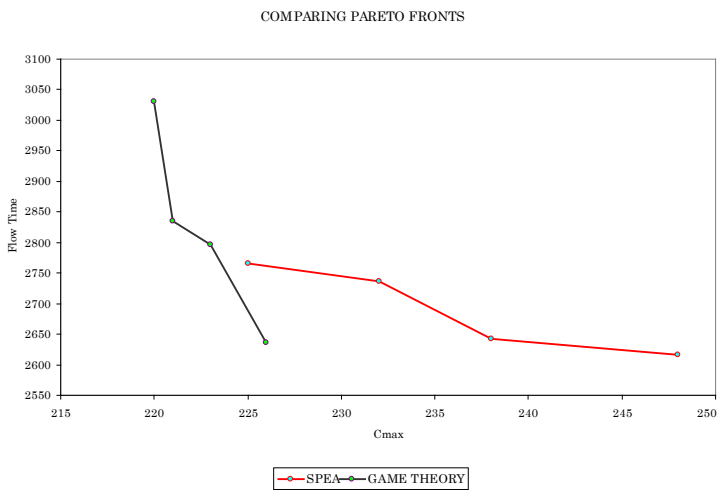

Fig. 9. Comparison of strict Pareto fronts for 30 jobs and 3 machines.

The results for this simulation shows an interesting range of solutions derived from both approaches. Results from one approach do not dominate the other, in fact, what was observed is that one approach complements the other. Thus suggesting these could work together to develop robust Pareto Frontiers in machine scheduling problems, presenting different alternatives to decision makers.

VII. Conclusions and Future Research Directions

Regarding the outcomes obtained and the solutions that were gathered, it can be stated that the sequencing of jobs found throughout the game theoretic model can generate alternate sources of solutions for these types of configurations($Pm||C_{max}$, $\sum C_i$), however it can be rather conservative with respect to the best results found so far in the MCDM approach. Apart from this, randomness may affect results.

The dynamic tradeoffs between the two agents and the fact that each one had a specific objective (conflictive), has demonstrated that it is possible to consider more than one objective for these configurations on productive systems. Additionally, the usefulness of the incentives used on the equations was demonstrated by the participating agents who are willing to move from positions and open the solution space by creating alternate schedules based on rational confrontations between agents.

On the other hand, meta-heuristics, such as evolutionary algorithms, have shown to be more efficient in obtaining solutions to production programming problems with a lower computational complexity than the game theoretic approach. It was also observed that both approaches can complement each other because each one seems to be concentrated on one side of the Pareto front. This is important because it can be considered for further

studies, where more robust solutions can be obtained through a combination of both procedures.

Gates towards further research are still broad and the results that have turned out from this study can be sharpened in order to make solutions more robust and, to get to equilibrium faster by considering more accurate equations concerning payoffs within the dynamic matrix. As a further analysis to the game theoretic interpretation of the mechanism proposed during this research, it is important to understand that it relies on a game that considers giving incentives to the players, whose payoffs have an additional portion related with a ß calculated for the game.

Branches that may originate from Game theoretic principles (GTP) and meta-heuristic approaches to productive systems can be said to be endless. For instance, this same problem may have a big research potential if set up times are considered. It can also be suited to analyze other objectives; working with due dates, deadlines, etc. GTP is suited to any environment where an intelligent agent has decisions to take that must satisfy someone else's, who is also acting rationally.

References

[1] V. Suresh and D. Chaudhuri, "Bicriteria scheduling problem for unrelated parallel machines," Comput. Ind. Eng., vol. 30, no. 1, pp. 77–82, Jan. 1996. DOI:10.1016/0360-8352(95)00028-3

[2] L. Yu, H. M. Shih, M. Pfund, W. M. Carlyle, and J. W. Fowler, "Scheduling of unrelated parallel machines: an application to PWB manufacturing," IIE Trans., vol. 34,no. 11, pp. 921–931. DOI: 10.1023/A:1016185412209

[3] M. Pfund, L. Yu, J. W. Fowler, And M. Carlyle, "TheImpacts Of Variability On Scheduling Approaches For A Printed Wiring Board Assembly Operation," J. Electron. Manuf., vol. 11, no. 01, pp. 19–31, Mar. 2002. DOI:10.1142/S0960313102000242

[4] Y. Yang and L. Tang, "Local Search Heuristic for Multiple Objective Coil Scheduling Problem on Unrelated Parallel Machines," in 2009 International Joint Conference on Computational Sciences and Optimization, 2009, vol. 2, pp. 777–780. DOI: 10.1109/CSO.2009.402

[5] Y.-K. Lin, J. W. Fowler, and M. E. Pfund, "Multiple-objective heuristics for scheduling unrelated parallel machines," Eur. J. Oper. Res., vol. 227, no. 2, pp. 239–253, Jun. 2013. DOI: 10.1016/j.ejor.2012.10.008

[6] C. Low, Y. Yip, and T.-H. Wu, "Modelling and heuristics of FMS scheduling with multiple objectives," Comput. Oper. Res., vol. 33, no. 3, pp. 674–694, Mar. 2006. DOI:10.1016/j.cor.2004.07.013

[7] M. J. Geiger, "Solving multi-objective scheduling problems—An integrated systems approach," in Artificial Intelligence in Theory and Practice, vol. 217, M. Bramer, Ed. Springer US, 2006, pp. 493–502. DOI: 10.1007/978-0-387-34747-9

[8] G. Kulcsár And M. K. Forrai, "Solving Multi-Objective Production Scheduling Problems Using A New Approach," Prod. Syst. Inf. Eng., vol. 5, pp. 81–94, 2009.

[9] S. Bandyopadhyay and R. Bhattacharya, "Solving multi-objective parallel machine scheduling problem by a modified NSGA-II," Appl. Math. Model., vol. 37, no. 10–11, pp. 6718–6729, Jun. 2013. DOI: 10.1016/j.apm.2013.01.050

[10] S. A. Torabi, N. Sahebjamnia, S. A. Mansouri, and M. A. Bajestani, "A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem," Appl. Soft Comput., vol. 13, no. 12, pp. 4750– 4762, Dec. 2013. DOI: 10.1016/j.asoc.2013.07.029

[11] E. Kutanoglu and S. D. Wu, "An Incentive Compatible Mechanism for Distributed Resource Planning," pp. 28, 2001.

[12] S. D. W. Erhan Kutanoglu, "An Auction-Theoretic Modeling of Production Scheduling to Achieve Distributed Decision Making," Ph.D. dissertation, Dept. Ind. Man. Sys. Eng., Lehigh Univ, Bethlehem, PA, 1997.

[13] A. Archer and E. Tardos, "Truthful mechanisms for one-parameter agents," in Proceedings 2001 IEEE International Conference on Cluster Computing, 2001, pp.482–491. DOI:10.1109/SFCS.2001.959924

[14] E. Even-dar, A. Kesselman, and Y. Mansour, "Convergence time to nash equilibria," School of Computer Science, Tel Aviv Univ., p. 17, 2003.

[15] N. Nisan, "Algorithms for Selfish Agents," in STACS 99 Lecture Notes in Computer Science, vol. 1563, C. Meinel and S. Tison, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–15.

[16] G.-Q. Zhang, J.-J. Wang, and Y.-J. Liu, "Scheduling Jobs with Variable Job Processing Times on Unrelated Parallel Machines," Sci. World J., no. 242107, pp. 1–7, 2014. DOI: 10.1155/2014/242107

[17] B. Heydenreich, R. Müller, and M. Uetz, "Games and Mechanism Design in Machine Scheduling-An Introduction," Prod. Oper. Manag., vol. 16, no. 4, pp. 437–454, Jan. 2009. DOI: 10.1111/j.1937-5956.2007.tb00271.x

[18] D. R. Rios, C. R. Pinto, and C. Paternina-Arboleda, Game Theoretic Approaches to Parallel Machine Scheduling: A Bi-objective Optimization Problem Viewed as a Non-cooperative Game of Two Players. LAP Lambert Academic Publishing, pp.176, 2012.

[19] E. Zitzler and L. Thiele, An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Zurich: TIK, 1998

[20] A. Cama Pinto, E. De la Hoz Franco, and D. Cama Pinto, "Las redes de sensores inalámbricos y el internet de las cosas," INGE CUC, vol. 8, no. 1, pp. 163–172, 2012.

[21] Y. Donoso and R. Fabregat, Multi-Objective Optimization in Computer Networks Using Metaheuristics. Boston, MA: Auerbach Publications, pp. 472, 2007

[22] C. E. Gómez Montoya, C. A. Candela Uribe, and L. E. Sepúlveda Rodríguez, "Seguridad en la configuración del servidor web Apache," INGE CUC, vol. 9, no. 2, pp. 31–38, 2013.

[23] E. Gutierrez and G. Mejía, "Evaluación de los algoritmos Genéticos para el problema de Máquinas en Paralelo con Tiempos de Alistamiento Dependientes de la Secuencia y Restricciones en las Fechas de Entrega." p. 6, 2006

[24] P. M. França, A. S. Mendes, and P. Moscato, "Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times," in Proceedings of the 5th International Conference of the Decision Sciences Institute, 1999, pp. 1708–1710

[25] A. Gómez Cabrera and A. R. Orozco Ovalle, "Simulación digital como herramienta para la gestión del conocimiento en la construcción de edificaciones en concreto," INGE CUC, vol. 10, no. 1, pp. 75–82, 2014.

[26] K. Deb and D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms. New York, NY: John Wiley & Sons, Inc., pp. 249-258, 2001

[27] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in Proceedings of the third international conference on Genetic algorithms, 1989, pp. 51–60

[28] A. P. Cortés Vásquez, "Sistema de Aprendizaje de Patrones de Navegación Web Mediante Gramáticas Probabilísticas de Hipertexto," INGE CUC, vol. 11, no. 1, pp. 72–78, 2015. Doi: 10.17981/ingecuc.11.1.2015.07

[29] E. Altman, T. Başar, T. Jiménez, and N. Shimkin, "Routing into Two Parallel Links: Game-Theoretic Distributed Algorithms," J. Parallel Distrib. Comput., vol. 61, no. 9, pp. 1367–1381, Sep. 2001. DOI: 10.1006/jpdc.2001.1754