GAME THEORETIC APPROACHES TO PARALLEL MACHINE SCHEDULING

DIANA GINETH RAMÍREZ RIOS

CLAUDIA MARCELA RODRÍGUEZ PINTO

Undergraduate Project as a requisite for graduation

DIRECTOR: Ing. Carlos Paternina Ph.D

UNIVERSIDAD DEL NORTE
School of Engineering
Industrial Engineering Department
Barranquilla, Colombia,
February, 2007

TITLE OF THE INVESTIGATION: GAME THEORETIC APPROACHES TO PARALLEL MACHINE SCHEDULING

ABSTRACT: A problem of scheduling jobs on two identical parallel machines is considered, that pursues minimizing two criteria in particular, makespan and total flow time. 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. The scenario implied each job behaving selfishly and attempting to move to a previous position in the machine, which generated a cost for the job agent, who is attempting to minimize the total flow time. At the same time, a controlling agent allows movements of jobs between any two machines, expecting to balance the load on the machines and minimizing maximum completion time. As a result of the dynamic trade-offs between the agents in repeated games, a Pareto Front set of points was obtained.

Key Words: identical parallel machines, makespan, partial makespan, total flow time, partial flowtime, non-cooperative game, dynamic trade-offs, repeated games, Pareto front.

TÍTULO DEL TRABAJO DE GRADO: LA TEORÍA DE JUEGOS APLICADO A LA PROGRAMACIÓN DE MÁQUINAS EN PARALELO

RESUMEN: En un problema de programación de máquinas idénticas en paralelo que persigue minimizar dos criterios en particular, lapso y tiempo de terminación total, un mecanismo basado en la teoría de juegos es propuesto para solucionarlo. Se considera un juego bipersonal no-cooperativo de 2x2 en el que cada jugador busca minimizar alguno de estos criterios que propone el problema de producción. Cada escenario implica que los jugadores jueguen de manera simultanea y busquen minimizar los costos que están relacionados con los criterios a optimizar. El jugador que representa al trabajo tiene la opción de dejar al trabajo en su posición actual o moverlo a una posición previa, buscando minimizar su tiempo de terminación; mientras que el otro jugador, un agente controlador, toma la decisión de dejar al trabajo en la máquina actual o moverlo a otra, esperando balancear la carga de la máquina y minimizar el lapso. Como resultado de una serie de juegos repetidos entre estos agentes, el Frente de Pareto es construido, mostrando un conjunto de soluciones eficientes al problema.

Palabras Clave: máquinas idénticas en paralelo, lapso, tiempo de terminación total, juego bipersonal no-cooperativo, agentes, juegos repetidos, Frente de Pareto.

DIRECTOR: Ing. Carlos Paternina Arboleda, Ph.D

AUTORES:

Ramírez Ríos, Diana Gineth (1982)

Rodríguez Pinto, Claudia Marcela (1982)

A nuestros padres y hermanos(as), con mucho cariño.
Una especial dedicación a nuestros abuelos,
porque el ejemplo que seguimos de ustedes
nos hace ser mejores personas.

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Carlos Paternina Arboleda, Ph.D, Ingeniero Industrial y Director del Proyecto de Grado por sus valiosas orientaciones.

Jair de La Cruz, MSc. Ingeniero Industrial, por su constante apoyo en la programación para obtención de los resultados.

Agradecimientos adicionales a aquellas personas que nos brindaron asesoría durante parte de este proceso: Luis E. Ramírez (Ingeniero Industrial) e Ivan Saavedra (Ingeniero de Sistemas).

TABLE OF CONTENTS

pg

INTRODUCTION…… 1
1. PROBLEM DESCRIPTION……...5
1.1. BACKGROUND INFORMATION…….....5
1.2. TERMINOLOGY AND NOTATION……10
1.2.1. Scheduling in production planning……..10
1.2.2. Game theory…… …….19
1.3. IDENTIFICATION OF THE PROBLEM…….....22
1.4. JUSTIFICATION…….....24
2. LITERATURE REVIEW……...26
2.1. PARALLEL MACHINE SCHEDULING……..26
2.1.1. Parallel machine scheduling problems……..26
2.1.2. Heuristic techniques found in literature……..34
2.2. MULTICRITERIA OPTIMIZATION THEORY……35
2.2.1. Multicriteria decision making (MCDM)……35
2.2.2. Multicriteria optimization problems…….....39
2.2.3. Definition of optimality……..40
2.2.4. Determining Pareto Optimality…….....42
2.2.5. Multicriteria linear programming…….....53
2.2.6. Multicriteria mixed integer programming.....54
2.3 MULTICRITERIA SCHEDULING PROBLEMS…….....54
2.4. GAME THEORY…….....56
2.4.1Two person zero-sum game and Nash Equilibrium……..……56
2.4.2 Resolution methods for non-dominated strategies…….....60
2.4.3 Mechanism Design…….....62
2.4.4. Game Theory and Computer Science…….....66
2.5. APPROACHES OF GAME THEORY IN SCHEDULING…….....69
3. OBJECTIVES……77

3.1. GENERAL OBJECTIVE……...77
3.2 SPECIFIC OBJECTIVES…….....77
4 SCOPE AND LIMITATIONS……78
4.1. SCOPE……78
4.2. LIMITATIONS…….79
5. HYPOTHESIS……...82
5.1. SET OF HYPOTHESES……...82
5.2. CONCEPT VARIABLES DEFINITION……...83
5.3. OPERATIONAL VARIABLES DEFINITION…….....84
6. METHODOLOGY…….....87
6.1 METHODOLOGY APPROACH……87
6.2 SOLUTION METHODOLOGY……..91
6.2.1. Elements and Assumptions……..92
6.2.2. Definition of the Game……..95
6.2.3. Numerical Example…….111
6.2.4. Results Obtained…….....128
6.2.5. Comparing Results to other Heuristics…….....132
6.2.6. An extension to the results…….....135
7. RESEARCH ASSOCIATED COSTS …….....149
8. ACTIVITIES SCHEDULE……..150
9. CONCLUSIONS AND FURTHER RESEARCH……151
9.1. CONCLUSIONS…….....151
9. FURTHER RESEARCH……154
BIBLIOGRAPHY…….....156
APPENDIX A
APPENDIX B

LIST OF TABLES

Table 1. Resources and Tasks (PINEDO and CHAO. p 15)

Table 2. Typology of Scheduling Problems (T'KINDT and BILLAUT, p17)

Table 3. Initial Allocation of jobs in 2 machines by Load Balancing.

Table 4. Job agent selected for the game (A6) and the possible movements it can make.

Table 5. Payoff Matrix for job Agent 5 in the first iteration.

Table 6. Proposed schedule for first iteration.

Table 7. Payoff Matrix for job Agent 9 for iteration 2.

Table 8. Proposed schedule for iteration 2.

Table 9. Payoff Matrix for job Agent 9 for iteration 3.

Table 10. Proposed schedule for iteration 3.

Table 11. Payoff Matrix for job Agent 10 for iteration 4.

Table 12. Proposed schedule for iteration 4.

Table 13. Payoff Matrix for job Agent 2 for iteration 5

Table 14. Proposed schedule for iteration 5.

Table 15. Payoff Matrix for job Agent 7 for iteration 6.

Table 16. Proposed schedule for iteration 6.

Table 17. Payoff Matrix for job Agent 3 for iteration 7.

Table 18. Proposed schedule for iteration 7.

Table 19. Payoff Matrix for job Agent 4 for iteration 8.

Table 20. Proposed schedule for iteration 8.

Table 21. Payoff Matrix for job Agent 2 for iteration 9.

Table 22. Proposed schedule for iteration 9.

Table 23. Payoff Matrix for job Agent 6 for iteration 10.

Table 24. Proposed schedule for iteration 10.

Table 25. Payoff Matrix for job Agent 3 for iteration 11.

Table 26. Proposed schedule for iteration 11.

Table 27. Payoff Matrix for job Agent 8 for iteration 12.

Table 28. Proposed schedule for iteration 12.

Table 29. Payoff Matrix for job Agent 7 for iteration 13

Table 30. Proposed schedule for iteration 13.

Table 31. Payoff Matrix for job Agent 2 for iteration 14

Table 32. Proposed schedule for iteration 14.

Table 33. Payoff Matrix for job Agent 2 for iteration 15.

Table 34. Proposed schedule for iteration 15.

Table 35. Payoff Matrix for job Agent 7 for iteration 16.

Table 36. Proposed schedule for iteration 16.

Table 37. Payoff Matrix for job Agent 10 for iteration 17.

Table 38. Proposed schedule for iteration 17.

Table 39. Payoff Matrix for job Agent 8 for iteration 18.

Table 40. Proposed schedule for iteration 18.

Table 41. Payoff Matrix for job Agent 10 for iteration 19.

Table 42. Proposed schedule for iteration 19

Table 43. Payoff Matrix for job Agent 4 for iteration 20.

Table 44. Proposed schedule for iteration 20.

Table 45. Payoff Matrix for job Agent 2 for iteration 21.

Table 46. Proposed schedule for iteration 21.

Table 47. Payoff Matrix for job Agent 2 for iteration 22.

Table 48. Proposed schedule for iteration 22.

Table 49. Payoff Matrix for job Agent 10 for iteration 23.

Table 50. Proposed schedule for iteration 23.

Table 51. Payoff Matrix for job Agent 8 for iteration 24.

Table 52. Proposed schedule for iteration 24.

Table 53. Values of Pareto Solutions of the Scheduling Game for this example.

Table 54. Schedules generated that belong to the weak Pareto Front solution set.

Table 55. Results from other algorithms/heuristics

Table 56. Comparison of the results for each replication in the instance m=2, n=10

Table 57. Comparison of the results for each replication in the instance m=2, n=20

Table 58. Comparison of the results for each replication in the instance m=2, n=30

Table 59. Comparison of the results for each replication in the instance m=3, n=10

Table 60. Comparison of the results for each replication in the instance m=3, n=20

Table 61. Comparison of the results for each replication in the instance m=3, n=30

Table 62. Comparison of the results for each replication in the instance m=4, n=10

Table 63. Comparison of the results for each replication in the instance m=4, n=20

Table 64. Comparison of the results for each replication in the instance m=4, n=30

Table 65. Improvement in the results and robustness of the solution presented by the proposed model.

LIST OF FIGURES

Figure 1. Conflicting functions on $\Re$ (CARLSSON and FÚLLER, p.4)

Figure 2. Supportive functions on $\Re$ (CARLSSON and FÚLLER, p.4)

Figure 3. Weak and Strict Pareto Optima where Z defines a polyhedron (T'KINDT and BILLAUT. Figure 3.3. p.48)

Figure 4. Geometric Interpretation of a problem (P_a) (T'KINDT and BILLAUT. Figure 3.9. p.47)

Figure 5. Geometric Interpretation of a problem ($P_{(g,b)}$) (T'KINDT and BILLAUT. Figure 3.10. p.61)

Figure 6. Geometric Interpretation of a problem (P_ek) (T'KINDT and BILLAUT. Figure 3.12. p.66)

Figure 7. Geometric Interpretation of a problem (P_q) (T'KINDT and BILLAUT. Figure 3.13. p.69)

Figure 8. Geometric Interpretation of a problem ($P_{(z}$ref, $_w)$) (T'KINDT and BILLAUT. Figure 3.16. p.79)

Figure 9. Supported and non supported Pareto optima (T'KINDT and BILLAUT. Figure 3.18. p.85)

Figure 10. Main variables definition.

Figure 11. Job agent's decision

Figure12. Analyzing β for the agent as an opportunity cost of affecting overall flow time

Figure 13. Agent 0 decides to switch the job to the other machine affecting C_j of second machine.

Figure 14. Agent 0 decides not to switch job, from the machine concerned, it will stay that way until another condition is reached.

Figure 15. Decision Tree for Job Agent

Figure 16. Decision Tree for Agent 0

Figure 17. Pareto Front for initial results of the example.

Figure 18. Pareto Front obtained from the Scheduling Game for this example.

Figure 19. Pareto Front contrasted with results from other MCDM* tools.

Figure 20. Comparison of the Pareto Fronts generated in the instance m=2, n=10.

Figure 21. Comparison of the Pareto Fronts generated in the instance m=2, n=20.

Figure 22. Comparison of the Pareto Fronts generated in the instance m=2, n=30.

Figure 23. Comparison of the Pareto Fronts generated in the instance m=3, n=10.

Figure 24. Comparison of the Pareto Fronts generated in the instance m=3, n=20.

Figure 25. Comparison of the Pareto Fronts generated in the instance m=3, n=30.

Figure 26. Comparison of the Pareto Fronts generated in the instance m=4, n=10.

Figure 27. Comparison of the Pareto Fronts generated in the instance m=4, n=20.

Figure 28. Comparison of the Pareto Fronts generated in the instance m=4, n=30.

INTRODUCTION

Without a doubt, the world today is the result of a huge course in evolution, and certainly mankind has not only witnessed, but also sculpted all surrounding aspects in society. For instance, from the beginnings of the industrial revolution until today, industry has completely changed in terms of labor and resource usage. In early stages of the revolution, factories focused entirely on mass production and efficiency. Under those given conditions they were considered “competitive” and those conditions alone enabled organizations to be productive, guaranteeing a long lasting environment and profitable outcomes.

Nowadays, during the knowledge and information era, things have changed quite a lot; now success does not only depend on increasing production, since this can merely make a difference. Markets are every day more demanding and dynamic, so production must be forecast-based, according to the expected demand. Products and services may also guarantee great quality, efficient resource usage, processing on time, maintaining the right level of inventory, or getting the product to the customer on time. Now, running and controlling so many variables at the same time can be very risky, and certainly complex, for any company; thus, taking under consideration several priorities at the same time is needed; being dynamic

while making decisions of this kind. As Bernard Roy said *"taking account of several criteria enable us to propose to the decision maker a more realistic solution"*[1].

It is not easy to make decisions these days, even when all businesses seek for competitiveness to achieve stability under very dynamic conditions, they reach for the best machinery and the latest technology, but that is not enough. They encounter problems almost daily, for example, the machine was not ready because the materials needed did not arrive on time, the operators did not prepare the machine properly, a machine has just broke down, so it did not start as expected, and so many other interferences that may take place any time within a productive system. Strategic decision making is crucial for these circumstances because it requires the most convenient decisions out of all the possible choices.

Scheduling theory came along in a time where production planning has become rather necessary for organizations. It has turned so important these days that its effectiveness can determine the permanence and fidelity of the clients in a business. In order to reach this, it is necessary that productive systems evolve over time. Recently, it has been quite obvious how technology has taken over and how the need for quick answers has become vital for enterprises permanence in all sorts of industries. Intelligent Business (IB) and industrial engineering have been

[1] T'KINDT, Vincent and BILLAUT, Jean-Charles. "Multicriteria Scheduling: Theory, Models and Algorithms." Germany: Springer-Verlag, 2002. p.1

introduced to management and along with them, a whole new concept of decision making, in which decisions have to come from optimal solutions.

The decision making process takes place every day, especially in businesses. There is no single area in a firm that does not require a person to make decisions. More importantly, decisions have become so crucial, that there is no time to think about the “best” one, it is a matter of taking risks all the time for the sake of the organization. Yet, decision makers have to make the “best” decisions among the branch of so many possibilities. Strategic thinking has done an important role in decision making; hence, it has been applied to such diverse scopes within mankind grasping problems nowadays. Game theory has introduced this new way of thinking and its applications have become widely known these days. In addition to this, many information systems have the capability to decide strategically and so the process is aided for the decision maker pointing and setting out a better illustrated map for the decision maker.

Although game theory has been applied to a variety of fields, this paper will focus on production programming, specifically in dealing with scheduling problems on identical parallel machines. For this type of environment it can be assumed that intelligent agents control it. Through this perspective game theory can be introduced as a mean to solve different criteria, each proposed by two intelligent agents, where each agent defends its criteria by assuming certain strategies that

take into account each other's decisions. That is, one agent will choose the best strategy knowing that the other agent is also choosing its best strategy.

1. PROBLEM DESCRIPTION

1.1. BACKGROUND INFORMATION

All sort of problems can become critical in a productive system and, for these situations, production planning is present to be ahead of them. Moreover, at a scheduling phase, some of the criteria that need to be taken under consideration for decision making are:

- Obtain a high utilization of machines and personnel.
- Minimize the number of extra hours of labor.
- Minimize inventory maintenance costs.
- Delays in the production that can be convenient for the customer.
- Minimize work-in-process costs.
- Minimize the manufacturing costs due to time spent setting up machines or idle time of the machines.

Even though the serial production system introduced by Henry Ford at the beginnings of the industrial age constituted a breakthrough for the economy in its time, it is no longer effective; since too much costing was brought upon in wasted material and extreme resource usage. As the production system became very

important in the industry there was more than a need of producing in large quantities. But not just producing and waiting for everything to be sold, markets became more demanding, so factories faced a new challenge, client satisfaction. Therefore, quality started to evolve and it was not only a matter of detecting a defect in the final product, and controlling quality through inspections, but a whole new concept of quality had to emerge, that was; quality management and prevention of defects, quality is not controlled, quality is thus created. For that reason, a constant supervision during the process of transformation of the product was indeed needed. Nevertheless, quality grew much more and now businesses practice more, commonly, the so called “total quality” process, which involves the long awaited and hope for customer satisfaction. Responding on time to the customer, dispatching products on time and responding effectively to their demands, are just some of the actions that manufacturing businesses need to take as part of their “total quality” process.

Production scheduling arises as a foundation for operational success in manufacturing processes. The tools used to measure it and the methods implemented have revolutionized today’s productive systems.

For instance, Pinedo and Chao (1999) state their perspective unto the scheduling approach by the following quote, “the scheduling function in a company uses mathematical techniques or heuristic methods to allocate resources to the

processing of tasks." [2] They classify the most important elements in scheduling as shown in this chart:

RESOURCES	TASKS
Machines at a workshop	Operations at a workshop
Runways at an airport	airport
Crews at a construction site	Stages in a construction project
computing environment	executed

Table 1. Resources and Tasks (PINEDO and CHAO. p 15)

The scheduling theory first appeared in the mid 1950's as a result of the need for organizing the production. For Carlier and Chrétienne (1988)[3],

> *"scheduling is to forecast the processing of a work by assigning resources to tasks and fixing their start times. (…) The different components of a scheduling problem are the tasks, the potential constraints, the resources, and the objective function (…) The tasks must be programmed to optimize a specific objective (…) of course, often it can be more realistic in the practice to consider several criteria."*

The last phrase shows the importance to the author quoted below, from considering *several criteria*, but that is because the real situation will not give rise to problems one at a time, and those kinds of problems cannot be said to be completely deterministic and known (neither does one criteria problem). As shown above, the criteria that are usually taken in consideration in a productive system are related to time or to costs. In practice, it is more likely to find more than just one factor to consider and to establish corresponding results because systems cannot

[2] PINEDO, Michael and CHAO, Xiuli. Op. Cit. p. 17
[3] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p. 5

be taken as isolated, since a lot of factors may deviate the solution forecasted by one single criteria model. Such models are too idealistic and will never correspond to reality.

Multiple Criteria Decision Making and Multiple Criteria Optimization started with Pareto at the end of the 19th century. Since then, this discipline has grown and developed, especially these last thirty years. To this day, many decision support systems have implemented methods to manage conflicting criteria, by using mathematical theory of optimization under multiple objectives.

On the other hand, game theory formally starts with Zermelo, whose studies show that games such as chess are in fact resolvable. Borel (1921) and Von Neuman(1951) are doubtlessly the best known to be the pioneers in minimax equilibrium, specifically in sum-zero games*. Nevertheless the important breakthrough came not until the early forties, when the book "Theory of Games and Economic Behavior", written by John Von Neumann and Oscar Morgenstern, is finally published. This book really came along to formalize the writings in an extended way and introduced the concept of strategy in extensive games and proposed some applications. Yet in the 50's, there was a great development of this theory, various publications were made in Princeton like, an introductory book by Luce and Raiffa (1950); Kuhn (1953), who defined the concept of information in

* *Sub-zero games are games in which, while one player gains some profit, the other loses the same amount.*

games; Shapley (1953), who established a way to attack cooperative games[**]; and finally, John Nash (1950), who defined the Nash equilibrium in zero-sum games. These last investigations were financed by the United States' Department of Defense, since sum-zero games could be applied to military strategies. Moreover, Harsany (1967) extended the theory of games to games with incomplete information[***] and then Selten (1975) defined the concept of perfect equilibrium in a sub game for games with incomplete information and a generalization to the case of games with imperfect information.

In 1994, the Real Academy of Sciences in Sweden awarded with a Nobel Price in Economy to the mathematician, John Nash and the economists, John Harsanyi and Reinhard Selten, for their "pioneer analysis of the equilibrium in non-cooperative games", which proved to be very useful for modern economic applications. Today game theory has proved to be an important tool and Nash's contribution was fundamental.

[**] *Cooperative games are games in which players can agree with each other on the decisions they take.*
[***] *Games with incomplete information are due to the uncertainty that the players have with all the characteristics of the game.*

1.2. TERMINOLOGY AND NOTATION

1.2.1. Scheduling in production planning

Manufacturing systems are described by various factors, like for example, the number of resources, the configuration of resources and its automatization. All of these different characteristics can be represented in many different scheduling models.

To understand scheduling models, there are important terms and notations that must be considered in order to understand algorithms and heuristics used in multiple criteria scheduling theory.

> "It is important to clearly distinguish between the variables that define the problem, (…) and those variables that describe the solution produced by the scheduling process. To emphasize this distinction we have adopted the convention that lower-case letters denote the given variables and capital letters denote those that are determined by scheduling. The symbols h, x, y, z and Q, X, Y, Z, will be used for those which apply to an individual section."[4]

The number of jobs is denoted by n and the number of machines by m. To name a specific job, J_i represents job number i. To name a specific machine, M_k represents machine number k.

[4] CONWAY, Richard, et.al. "Theory of Scheduling". USA: Dover Publications, 1967. p.9

Vincent T'Kindt and Jean-Charles Billaut, in their book of *Multicriteria Scheduling*, described the different classification of different scheduling problems and the configurations of the resources used[5].

Types of scheduling problems without assignment:

- **Single machine:** Any job is processed in one machine. This is one of the most important types of problems because in practice, solutions to more complex problems are often found by analyzing it as a single machine.

  "Single-machine models are also important in decomposition approaches, where scheduling problems in complicated environments are broken down into smaller, single-machine scheduling problems."[6]

- **Flow Shop (F):** Jobs have the same route and are processed in a series of machines in the same order. Whenever a job completes its processing on one machine it joins the queue on the next. A subset from this group is the Flexible Flow Shop which contains a number of stages in series with a number of machines in parallel at each stage.
- **Job Shop (J):** Each job has a route of its own but the machines are in the same order. The simplest ones assume that a job may be processed at most once. In others a job may visit a machine several times on its route

[5] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.8-9
[6] PINEDO, Michael and CHAO, Xiuli. "Operations Scheduling: With Applications in Manufacturing and Services." USA: McGraw-Hill, 1999. p.15

system. These configurations arise in many industries as the aluminum foils industry or the semiconductors industry.

- **Open Shop (O):** Jobs do not have a definite route to follow and they can be processed in the machines with any order.
- **Mixed Shop (X):** Some jobs have a certain route, others do not.

Configurations of Machines:

- **Identical machines (P**): machines that have the same processing time.
- **Independent machines (R):** Processing time of operation $O_{i,j}$ on machine M_k is $P_{i,j,k}$.

Traditional scheduling and assignment problems:

- **Parallel Machines (P/Q/R):** Problems that have only one stage and jobs have only one operation
- **Hybrid Flow shop (HF):** Problems where jobs have the same route and various stages in the same order.
- **General Job Shop (GJ):** Problems where each job has its own route.
- **General Open Shop (GO):** Problems where jobs do not have a fixed routing.

When dealing with scheduling problems, there are always one or various constraints that need to be measured or taken in consideration for the solution. Some of the most explicit constraints are the processing times *(p_i),* due dates *(d_i),*

release dates *(r_i)* and weights *(w_i).* When solving scheduling problems, some optimality criteria is needed in order to evaluate schedules according to the priorities in the production. It is important to notice that the *difference between a criteria and a constraint only depends on the decision maker*:

> "For example, stating that no job should be late regarding its due date leaves no margin in the schedule calculation. We may even find a situation where no feasible schedule exists. On the other hand, minimising the number of late jobs allows us to guarantee that there will always be a solution even though to achieve this certain operations might be late. (…) the difference between a criterion and a constraint is only apparent to the decision maker (…)." [7]

Explicit constraints in a problem are of various types; some of the ones more used in theory and practice are shown:

- **Release dates (r_j):** Sequence dependent setup s_{jk} means setup time between job *j* and *k.* This is used when set up time depend on the job that is placed on each machine.
- **Preemptions (prmp):** a β field that has this constraint implies that it is not necessary to keep a job on a machine, once started until completion. It is allowed to interrupt the processing of a job and put a different one. It is assumed that the amount of processing a preempted job already has received is not lost. When (prmp) is omitted then preemptions are not allowed.

[7] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.12

- **Breakdowns:** Imply that machines are not continuously available. The time is assumed to be fixed, and for parallel machines it is available at any point in time, that is breakdowns can be put as functions of time.
- **Machine eligibility restrictions (M_j):** When this field is present in parallel machines environment, this (M_j) denotes the set of machines that can process the job *j.*

Criteria are classified in *minimax* criteria and *minisum* criteria; the first one refers to minimize the maximum value of a set of functions and the second one refers to minimize the sum of the functions. Some of the most common criteria used in literature[8] are:

- $C_{\max}$ (*makespan*) $= \max(C_i)$ With C_i, being the completion time of the job J_i, $i = 1,2,...,n$
- $F_{\max}$ (*flowtime*) $= \max(F_i)$ $F_i = C_i - r_i$, for r_i being the release date and $F_{\max}$ the maximum time spent on a job.
- $I_{\max} = \max(I_{ki})$ $I_{\max}$, which I_k is the sum of idle times of resource M_k
- $L_{\max} = (\max, lateness) = \max(L_i)$ with $L_i = C_i - d_i$
- $T_{\max} = (\max : tardiness) = \max(T_i)$; $T_i = \max(0; C_i - d_i)$
- $E_{\max} = (\max : earliness) = \max(E_i)$; $E_i = \max(0; d_i - C_i)$

[8] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.13

- $\overline{C} = \frac{1}{n}\sum C_i$ (*average completion time*) or $\sum C_i$ *(total completion time)*
- $\overline{C}^w = \frac{1}{n}\sum w_i C_i$ *(average weighted completion time)* or $\sum w_i C_i$ *(total completion time)*
- $\overline{F} = \frac{1}{n}\sum F_i$ (*average flow time) or* $\sum F_i$ *(total flow time)*
- $\overline{F}^w = \frac{1}{n}\sum w_i F_i$ *(average weighted flow time)*
- $\overline{U} = \sum U_i$; which is the number of late jobs with U_i = 1 if job J_i is late and 0 if its not.
- $\overline{U}^w$ (*weighted number of late jobs)*
- $\overline{E}$ (*average number of early jobs)*
- $\overline{E}^w$ *=(weighted number of early jobs)*

There are two basic approaches when scheduling problems are analyzed:

- **Backward Scheduling:** by this approach scheduling problems are analyzed taking the due date as a set point and determining the date on which each operation must start by using "inter-operational" acceptance time laps"(1989)[9]. The risk taken by using this approach can be that the starting date can be past date from present.

[9] COMPANY, Ramon, "Planeación y Programación de la producción" España: Barcelona, 1989.p

- **Forward Scheduling:** Problems are analyzed from the release dates and on until due dates. The risks taken can include not attaining to finish on the proclaimed due dates.

Independently from the approach implied, Gantt charts are used in both; programming setting results and data transmissions.

From the types of problems stated above there is a special typology that identifies one problem from the other. Table 1.1 describes this typology in a graphical way. Nevertheless, there are other typologies that need to be considered in problems:

- **Deterministic or stochastic:** problems might have all its characteristics well known, while other problems can have its characteristics described by random variables.
- **Unitary or repetitive:** Operations in a problem can correspond to a unique product or can appear to be cyclical.
- **Static or dynamic:** all data of the problem can be known at the same time or can be calculated and processed during the arrival of new operations.

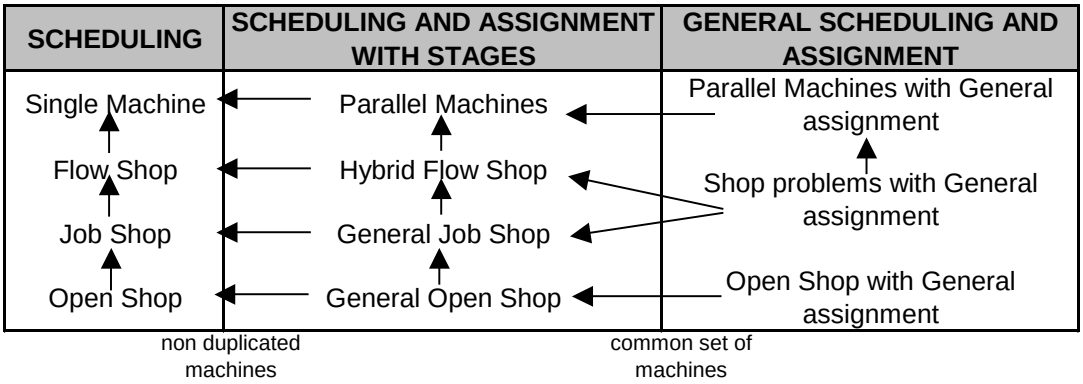

Table 2. Typology of Scheduling Problems (T'KINDT and BILLAUT. Figure 1.1. p.15)

One of the first to propose a notation for scheduling problems was Conway (1967)[10], yet the most frequently used in literature was introduced by Graham (1979)[11], which is divided into: α | β | γ[12].

- α contains the typology shown in the Table 1.1., describes the structure of the problem. $\alpha = \alpha_1\alpha_2$. The sub-fields α_1 and α_2 refer to the type of scheduling problem and the number of machines available, respectively.
- β contains the explicit constraints of the problem.
- γ contains the criterion or criteria to be optimized in the problem.

These are some of the basic rules for solving certain types of problems[13]:

[10] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op.Cit. p.16
[11] Ibid. p.16
[12] Ibid. p.16
[13] Ibid. p.21

- SPT (Shortest Processing Time): Orders the jobs taking first the one that has the shortest processing time. LPT* (Longest Processing Time) is the converse rule.
- SRPT (Shortest Remaining Processing Time) is the preemptive version of the SPT rule and LRPT (Longest Remaining Processing Time) is the converse rule.
- WSPT (Weighted Shortest Processing Time first): Sequences the jobs in increasing order of their ratio p_i / w_i.
- EDD (Earliest Due Date): Orders the jobs by taking first the earliest due date.
- EST (Earliest Starting Time): Sometimes jobs require of a release time, so if it is the case, jobs can be ordered by taking the one that has the earliest starting time.
- FAM (First Available Machine): In parallel machines, it can be necessary to use this rule by placing the job in the next available machine.
- SPT-FAM (Shortest Processing Time - First Available Machine): When assigning jobs to the machines, sometimes it's important to consider the shortest processing time first.
- EDD-FAM (Earliest Due Date - First Available Machine): When assigning jobs to the machines, it can also be important to consider the earliest due date first. $p|d_i|L_{\max}$

* *LPT is commonly used when the objective is minimizing the makespan (C_{max})*

- FM (Fastest Machine First): When machines are not identical, it might be needed to process in the fastest machine first. SPT and EDD can also complement this rule.
- WSPT-FAM (Weighted Shortest Processing Time - First Available Machine):

$$p\left|\left|\overline{C}_i{}^w\right|\right.$$

In order to solve all types of scheduling problems, special procedures are used where the rules mentioned above are taken in consideration. It is important to note that when a scheduling problem belongs to a class P, there is an exact polynomial algorithm to solve it. Otherwise, if the problem belongs to the class NP-hard, there are two possible solutions, either a *heuristic*[*] is proposed to calculate a problem in polynomial time, or an algorithm is used to calculate the optimal solution, but its maximum *complexity*[**] is exponential.

1.2.2. Game Theory

Game theory is the study of the strategic interaction between two or more individuals (also known as players) who take decisions that will affect in some way, depending on what one expects from the other or others. There are two ways to describe games, in the strategic form and in the extensive form. To understand the

[*] *A heuristic is an approximated algorithm.*
[**] *The complexity of an algorithm is measured both in time and in memory space. Yet, in this investigation the complexity is given by the time, calculating the number of iterations the algorithm takes to be processed.*

concepts used in game theory, Fudenberg and Tirole, in their book *Game Theory*, refer to the notation used for games in strategic form[14].

1.2.2.1. Games In Strategic Form

It is composed of three basic elements:

- A set of players $i \in I$, assumed to be in a finite set $\{1, 2, \ldots, I\}$
- The *pure-strategy space* S_i for each player *i*
- The *payoff functions* u_i, which gives each player what von Neumann and Morgenstern called their utility $u_i(s)$ for each set of strategies $s=(s_1, \ldots, s_I)$.

Since players are denoted with i, the *"player i's opponents"* will be referred to as *"-i"*. This terminology is used to emphasize that a player's objective is to minimize his own payoff function and this will affect, positively or negatively, the other player or players. Furthermore, there are important terms that need to be taken in consideration:

- Equalizing strategy: This strategy results in the same average payoff for all players no matter what each player does.
- Value of the game: The average value of the payoff given that both players have played in a proper way.
- Minimax strategy: The optimal strategy that results in the value of the game.

[14]FUDENBERG, Drew and TIROLE, Jean. "Game Theory". Massachusetts: MIT Press, 1991. p. 4

- Pure Strategy (s_i): The optimal strategy that comes from choosing only one of the strategies that a player has.
- Mixed strategy (σ_i): The optimal strategy is the result of taking various proportions of the pure strategies, in which the randomness of the distribution for each player is statistically independent from that of the opponents and the payoffs to each player are the expected values of the payoffs for those pure strategies.
- Utility theory: It states that the payoff must be evaluated by its utility to the player rather than the numerical monetary value.
- Common knowledge: "*Structure of a game that assumes that all players know the structure of the strategic form, and know that their opponents know it, and know that their opponents know that they know, and so on ad infinitum."* [15]
- π_i **:** Mathematical expectation of probability distribution of a player function, given that player i is using strategy, this notation represents the mathematical expectation of the payoff function, given that player i is using strategy $\sigma_{i,} \in \sum_i$

$$\pi(\sigma_{1,}\sigma_{2,}...\sigma_{n,}) = (\pi_1(\sigma_{1,}\sigma_{2,}...\sigma_{n,}), \pi_2(\sigma_{1,}\sigma_{2,}...\sigma_{n,}), ..., \pi_n(\sigma_{1,}\sigma_{2,}...\sigma_{n,}))$$ [16]

1.2.2.2. Games in an Extensive Form

[15] FUDENBERG, Drew and TIROLE, Jean. Op.Cit. p.4
[16] OWEN, Guillermo. "Game Theory". San Diego: Academic Press.1995. p5.

By an n-person game in extensive form is meant. (α) a topological tree Γ with a distinguished vertex A called the starting point of Γ;

(β) a function, called the payoff function, which assigns an n-vector to each Terminal vertex of Γ;

(γ) a partition of the non terminal vertices of Γ into n+1 sets $S_0, S_1, ..., S_n$, called the player sets;

(δ) a probability distribution, defined at each vertex of S_0, among the immediate followers of this vertex.

(ε) for each $i=1,2,...,n,$ a subpartition of S_i into subsets S_0^j, called *information subsets*, such that two vertices in the same information set have the same number of immediate followers and no vertex can follow another vertex in the same information set.

(ζ) for each information set S_i^j, an index set I_i^j together with a 1 -1 mapping of the set I_i^j onto the set of immediate followers of each vertex of S_i^j.

Condition (α) states that there is a starting point, (β) gives a payoff function, (γ) divides the moves into chance moves S_0 and personal moves which correspond to the n players $S_0, S_1, ..., S_n$, ; (δ) defined a randomization scheme at each chance move; (ε) divides a player's moves into "information sets": he knows which information set to be, but not which vertex of the information set. [17]

[17] OWEN, Guillermo. "Game Theory". San Diego: Academic Press.1995 p2

1.3. IDENTIFICATION OF THE PROBLEM

Today's productive systems confront a reality that requires them to be each day more competitive. Industries all around the world are using their best technology to make better decisions when planning production. Although technology is advancing day after day, the main importance relies on the ways decisions are made. Yet, modeling productive systems in order to plan the production can turn out to be a fairly complex job. These complex problems cannot be solved with basic scheduling algorithms and that is why today's researchers have studied these problems in many different configurations. Tanaev (1994), Pinedo (1995), Blazewicz(1996) and Brucker(1998)[18] are just some of the authors that have proposed algorithms and heuristics to solve scheduling problems following different objectives in each one of them. Yet, the most important breakthrough in scheduling has been with problems involving multiple objectives. These are the kind of problems that model real life situations in industries nowadays and it is necessary to understand the different approaches that have been done to solve them.

More specifically, this investigation will analyze a particular type of problem configuration, parallel machine scheduling problems. Usually, the schedule for this type of configuration results from the arrangement of each one of the n jobs,

[18] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op.Cit. p.21

assigned to *m* number of machines and 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). 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 order of decreasing processing time is constructed (LPT).

Assuming that for this type of problem, a schedule that minimizes makespan is found. Does this mean that inventory maintenance costs are also minimized? Or would it mean that work in process is reduced? The highest chance is that none of this may happen. Considering the alternatives in a specific scheduling environment, the decision maker must give priorities according to these, and construct an appropriate sequence to meet the specific requirements. Even so, some of the objectives considered might be *conflicting*[*] and thus, focusing in reducing one of the priorities may lead to an excessive increase in the other criteria values, resulting in a net loss. The decision maker must take into account these considerations, appropriately choosing the combination that could result *minimizing the maximum loss*.[**]

[*] *Criteria are considered to be conflicting if the reduction of one leads to increase the other.*
[**] *"Minimax or Maximin Strategy": a term commonly used in game theory stated in a two-person zero sum game.*

1.4. JUSTIFICATION

In today's industries, a production planner faces not only the problem of attaining a good schedule sequence to provide results to meet client's needs, but must also consider minimizing total costs, or inventory costs; for instance. This is why; focusing on single criterion may just hinder finding a more integral solution. Finding the proper combination of criteria not only provides a more robust solution, but may also approach to real systems environment in a more direct way.

Amplifying the existing approaches to multicriteria scheduling problems using mathematical models such as the ones in game theory, broadens this field for further research in these topics, as well as having more alternate routes to find Pareto Fronts. However, this is still a new topic and there are a lot of gaps and unknowns yet to discover, so the results gathered may differ slightly from the ones found through Pareto analysis. The need to open new ways to tackle problems just opens a gate that in the future may turn to more knowledge.

This way, scheduling theory will eventually evolve and therefore strengthen actual possible applications. The interaction of both *scheduling and game theory* is therefore, positive for both research areas. On one side, game theoretic applications can involve more topics than the usual ones treated in economic decisions and biology. On the other side, scheduling, being so important

throughout the 20th century, especially for the last 50 years, has obtained good solutions for short term decisions and will continue bringing results to meet today's client's changing and demanding needs.

2. LITERATURE REVIEW

2.1. PARALLEL MACHINE SCHEDULING

2.1.1. Parallel machine scheduling problems

While analyzing the configuration of these type of problems, the following assumptions will be taken in consideration:

- Jobs are independent
- They arrive simultaneously
- The setup is sequence independent
- There is one or more machines to perform the processing
- All machines are identical within the system

When analyzing an environment of parallel machines the next matrix is therefore useful to analyze what is going on in the shop:

		Machines				
		1	**2**	**3**	**…**	**m**
Jobs	**1**	p_{11}	p_{12}			p_{1m}
	2	p_{21}				
	.					
	.					
	.					
	n	p_{n1}				p_{nm}

Here p_{ij} is the time to perform the single operation of job i on the machine j. This of course assumes that the machine j performs the whole operation. And the simplest case involves identical machines, which implies that all elements on a given row are equal. If there are different numbers within the same row, then the jobs have different processing time that depends on the machines, this is common where the resource is people and therefore there might be specialties for each working performances. If the machines alone have different performance rates or speeds then the whole column for each machine has a number that corresponds to that acquainted speed. Otherwise the subscripts can be omitted and p alone can be used. It is assumed for this research that all machines are identical.

A key question in the situation concerning parallel machines arises: Is it better to divide a single job so that the process time of this job is minimized? As stated by CONWAY (1967)[19] *:"If some division of the job is allowed, then better schedules are possible, but the determination of the schedule is more difficult."*

This practice is reasonably common in some types of industries, especially the ones that may have operations that are repetitions of some smaller elements in work on these pieces.

[19] CONWAY, et. al., Op. Cit. p.75

For instance, let m be the number of identical machines and n jobs to be performed, each with a processing time p. There is a total of mp work to be done, and if this is divided equally among the machines, they will finish simultaneously after p time units, Regardless of how the jobs are assigned to individual machines. If the job is divided as said, then the first job will finish at p/m, the second time at $2p/m$, the third at $3p/m$, and so on. Therefore, the average flow time can be obtained by adding al these terms and dividing them by m. The following sequence will be obtained.

$$\overline{F} = \frac{p}{m}\left(\frac{1}{m} + \frac{2}{m} + \frac{3}{m} + \ldots + \frac{m}{m}\right) = \frac{p}{m^2}\sum_{i=1}^{m} i = \frac{n+1}{2m}p$$

Taking these to the limits it is easy to see that when there are too many machines, that is, when the limit tends to ∞ this procedure can improve flow time up to 50%, and in the extreme when there are only two machines the increasing percentage is 25.

One could see m machines working simultaneously on a single job, as a single machine with m times the power of the basic machine. So, from a scheduling point of view, it is better to provide required capacity to a single machine than to an equivalent number of separate machines. However considerations of reliability work operate in opposite directions.

If machines are identical a pseudo-processing time is defined for each job as $p' = \frac{p_i}{m}$ If machines are uniform then this p' is given by $p_i' = \frac{1}{\sum_{j=1}^{m} 1/p_{ij}}$

However, there are many cases in which this is not possible, in such cases; the scheduling procedure consists on assigning a job to both a particular machine and to a position in sequence on that machine. Let $j[k]$ be the job which is in the *k*th position in sequence on the j th machine and n_j be the number of jobs processed on the j th machine:

The mean flow time is given by: $\overline{F} = \frac{\sum_{j=1}^{m} \sum_{k=1}^{nj} (n_j - k + 1) p_{j[k]}}{n}$

One can eventually interchange jobs in equivalent positions in sequence without affecting in mean flow-time, yet SPT rule does not guarantee to minimize maximum flow time. An example taken from CONWAY shows how sometimes this cannot be achieved:

Let there be four jobs with processing time 1,2,3,10 to be processed, the two shortest processing time schedules A and B, have maximum flow-times of 12 and 11, respectively. Schedule C is not a shortest processing time schedule; it has a maximum flow time of 10 but a greater mean flow-time.

	Schedule						
	A		B		C		
Machine 1	1	3	2	3	1	2	3
Machine 2	2	4	1	4	4		

For the *m* identical machine case, in which each job must be assigned to an individual machine, no optimal procedure has been offered, but still there are bounds such as the following, called, weighted lower bound for weighted mean flow time:

$\overline{F}_u \geq \frac{m+n}{m(n+1)} \overline{F}_u$ *"It is known that no greater bound is possible by exhibiting a set of jobs that actually attain this bound".*

Even when it is not possible to find solutions to given problems such as $\frac{C_{\max}(LPT)}{C_{\max}(OPT)}$ because most problems for parallel machines are NP-hard; other important priorities in parallel machine environments arise, such as makespan. By minimizing makespan the sequence obtained is going to be the shortest one, and many times, depending on the constraints these can be achieved. As stated before, in parallel machines preemptions play a more important role than with single machines. For these models there are optimal schedules.

In order to show the advantages that preemptions allow for parallel machine environments it is useful to see how efficient both models can be.

THE MAKESPAN WITHOUT PREEMPTIONS:

First, it has been demonstrated that the problem $Pm|\ |C_{\max}$ is NP-hard. During the last couple of decades, many heuristics have been proposed; a very

common one is the (LPT) rule*, in which the largest jobs are assigned to the *m* machines. After that, whenever a machine is freed, the longest job among those not yet processed is put on the machine. So this heuristic tries to place the shorter jobs toward the end of the schedule, this way it balances the load.

In order to give an indication of the efficiency of this algorithm Pinedo(2002) [20] determines a lower bound given by the following expression:

$$\frac{C_{\max}(LPT)}{C_{\max}(OPT)}$$

For these types of problems the given inequality is always true:

$$\frac{C_{\max}(LPT)}{C_{\max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m}$$

C_{max}(LPT) denotes the makespan of the LPT schedule and C_{max}(OPT) denotes the makespan of the (possible unknown) schedule.

COMPLETION TIME WITHOUT PREEMPTIONS:

When the objective is completion time, that is $Pm||\sum C_j$ then the SPT rule gives an optimal solution, and thus minimizes this given objective, but for instance the problem $Pm||\sum w_j C_j$ is NP hard. So this result cannot be generalized to parallel machines. It has been shown that the WSPT heuristic is a good heuristic, the worst case on this heuristic leads to the lower bound:

---

* *See Terminology and Notations, algorithms and heuristics.*

[20] PINEDO, Michael. "Scheduling: Theory, Algorithms, and systems". New Jersey. Upper Saddle River.2002.p 94.

$$\frac{\sum w_j C_j(WSPT)}{\sum w_j C_j(OPT)} \leq \frac{1}{2}(1+\sqrt{2})$$

THE MAKESPAN WITH PREEMPTIONS:

Sometimes allowing preemptions simplify the analysis, for example the problem $Pm|prmp|C_{\max}$, sometimes even linear programming LP formulation can be used to obtain information about the optimal solution, take PINEDO (2002) [21] linear programming formulation of the problem (LP):

$Minimize\ C_{\max}$

$subject\ to$

$$\sum_{i=1}^{m} x_{ij} = P_j \qquad j = 1,2,...,n$$

$$\sum_{i=1}^{m} x_{ij} \leq C_{\max} \qquad j = 1,2,...,n$$

$$\sum_{i=1}^{n} x_{ij} \leq C_{\max} \qquad i = 1,2,...,m$$

$$x_{ij} \geq 0 \qquad i = 1,2,...,m \qquad j = 1,2,...,n$$

Where x_{ij} represents the total time job *j* spends on machine *i.* The first set of constraints makes sure the jobs receive the required amount of processing. The second enforces that the total amount of processing each job receives is less or equal to the makespan. The third set makes sure that the total amount of processing on each machine is less than the makespan. The solution of course does not prescribe an actual schedule, it just specifies the amount of time job *j*

[21] Ibíd. p.105

should spend on machine *i*, and from this point a schedule can be constructed.[22] :

Taking into account the fact that $C_{\max} \geq \max\left(p_1, \sum_{j=1}^{n} p_j / m \right) = C^*_{\max}$

This bound allows constructing a simple algorithm that finds an optimal solution:

Algorithm 5.2.3 (minimizing Makespan with Preemptions)

Step1. *Take the n jobs and process them one after another on a single machine in any sequence. The makespan is then equal to the sum of the n processing times and is less than or equal to* $mC^*_{max.}$

Step2. *Take a single machine schedule and cut it into m parts. The first part constitutes the interval* $[0, C^*_{\max}]$, *the second part the interval* $[C^*_{\max}, 2C^*_{\max}]$, *the third part of the interval* $[2C^*_{\max}, 3C^*_{\max}]$ *and so on.*

Step3. *Take as the schedule for machine 1 in the bank of parallel machines the processing sequence of the first interval; take as the schedule for the machine 2 the processing sequence of the first interval; and so on.*

Another way to obtain an optimal solution is through one of the most used strategies, is the LRPT rule*. This schedule is structurally appealing, in the theoretical point of view, but in the practical point of view, it has drawbacks

[22] PINEDO, Michael. Op. Cit. p.106

* *See Terminology and Notations, Section 1.2.1.*

because most of the times the number of preemptions in the deterministic approach is infinite.

2.1.2. Heuristic Techniques found in Literature [23]

Kurz and Askin (2001) presented three: Slicing (SL), Multi-Fit (MMF) and Multiple Insertion (MI). SL solved the problem first as if it was a single machine and then the sequence was divided in the m machines. MMF starts by assigning the jobs to the machines and then solves the problem of assigning to each machine using the techniques of solution given by the Traveling Salesman Problem (TSP). MI orders the jobs first by SPT and then assigns the jobs in the machines by placing the job first on the machine that has the least partial makespan.

Franca, [24] et.al. (1994) developed a heuristic technique that minimizes the makespan and it consists on three steps. Franca, et.al. (1994) developed a heuristic that does not consider set-up times and its objective is to minimize makespan in three steps: jobs are classified in each machine in order to maintain the machines with approximately the same load, then they are balanced *by*

[23] GUTIERREZ, Eliécer and MEJÍA, Gonzalo. "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.". Universidad de los Andes: Enero 25, 2006. (Pdf document form the World Wide Web) p.6
http://triton.uniandes.edu.co:5050/dspace/bitstream/1992/812/1/Evaluacion+de+algoritmos+geneticos+para++el+problema+de+maquinas+en+paralelo.pdf

[24] Franca, et.al.

passing jobs from the machine with more load to the one with less and, finally, the machines are balanced once more by *switching jobs between them.*

2.2. MULTICRITERIA OPTIMIZATION THEORY

2.2.1. Multicriteria Decision Making (MCDM)

In the process of decision making there are a set of tools that permit a correct approach to an optimal solution of a problem. Many authors have presented significant contributions and, in general, the MCDM approach is more of a description where possible solutions are defined, including the attributes and evaluation of the criteria, but most importantly, there is a utility function where the criteria is incorporated. This utility function has to be maximized during this process and that is how optimal solutions are reached.

> There has been a growing interest and activity in the area of multiple criteria decision making (MCDM), especially in the last 20 years. Modeling and optimization methods have been developed in both crisp and fuzzy environments.[25]

There are several axioms presented by Boysseu (1984) and Roy (1985)[26] that are fundamental to MCDM:

---

[25] CARLSSON, Christer and FÚLLER, Robert. "Multiple Criteria Decision Making:The Case of Interdependence". p.1 http://citeseer.ist.psu.edu/cache/papers/cs/607/ftp:zSzzSzftp.abo.fizSzpubzSziamsrzSzco95p.pdf/carlsson95multiple.pdf

[26] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op.Cit. p.43

- The decision maker always maximizes, implicitly or explicitly, a utility function.
- An optimal solution exists for every situation.
- No comparable solution exists, it will always need to have to choose or sort between a pair of decisions.
- Decision maker's preferences can depend upon two binary relations: preference (P) and indifference (I).

Apart from the fundamentals, T'Kindt describes in MCDM two different approaches[27]:

- Multiple Attribute Utility Theory (MAUT): Proposed by Von Newman and Morgenstein in 1954 is more of a stochastic approach that is done when decisions are subject to uncertainty at a criteria level.
- Analytical Hierarchy Process (AHP): Proposed by Saaty in 1986, in which criteria is classified in groups using a hierarchical analysis in form of a tree and each criterion has been weighted in the utility function.

Yet there are also some limitations to MCDM because problems are said to be unrealistic and this makes the theory less useful than what it should be. According to Zeleny (1992)[28], MCDM is not useful when there is time pressure, when the problem is more completely defined, when using a strict hierarchical decision

[27] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op.Cit. p.43
[28] CARLSSON, Christer and FÚLLER, Robert. Op.Cit. p. 2

system, when there is changing environment, when there is limited or partial knowledge of the problem and when there is collective decision making in businesses; all this because it reduces the number of criteria being considered, leaving behind other possible alternatives.

Some authors, like Carlsson and Fuller, agree that the traditional assumption used in MCDM, in which the criteria are taken as independent, is very limited and ideal to be applied to today's business decision making. Reeves and Franz introduced a multicriteria linear programming problem, where they presume the decision maker has to determine his preferences in terms of the objectives but he must have more than an intuitive understanding of the trade-offs he is probably doing with the objectives. For this reason, an assumption is made and that is, that a decision maker is taken to be a rational thinker and with a complete understanding of the whole situation in which his preferences have some basis with the use of a utility function.

It has been universally recognized that there is no such thing as an optimal solution valid for any multiobjective problem. In literature, much has been found in terms of different approaches to solving MCDM problems. Delgado, et. al. (1990) used, for example, fuzzy sets and possibility theory not only to involve MCDM but also, multiobjective programming. Also, Felix (1992) worked with fuzzy relations between criteria by presenting a novel theory for multiple attribute decision making.

Carlsson, on the other hand, "used fuzzy Pareto optimal set of non-dominated alternatives to find the *best compromise solution* to MCDM problems with interdependent criteria".[29]

In order to understand more about the interdependencies between criteria, it is important to notice the problem defined by Carlsson in terms of multiple objectives:

$$\max_{x \in X} \{ f_1(x), ..., f_k(x) \}$$
$$where\ f_1 : \Re^n \to \Re\ are\ objrective\ functions$$
$$x \in \Re^n\ is\ a\ decision\ variable\ and\ x\ is\ a\ subset\ of\ \Re^n.$$

Definition[30]:

Let f_i and f_j be the two objective functions of the problem defined above.

i. f_i *supports* f_j *on* X (denoted $f_i \uparrow f_j$) *if* $f_i(x') \geq f_j(x)\ entails\ f_j(x') \geq f_j(x),\ for\ all\ x', x \in X;$

ii. f_i *is in conflict with* f_j *on* X (denoted $f_i \downarrow f_j$) *if* $f_i(x') \geq f_j(x)\ entails\ f_j(x') \leq f_j(x),\ for\ all\ x', x \in X;$

iii. *Otherwise,* f_i *and* f_i *are independent on X.*

[29] CARLSSON, Christer and FÚLLER, Robert. Op. Cit. p.4
[30] Ibíd. Def. 2.1. p.4

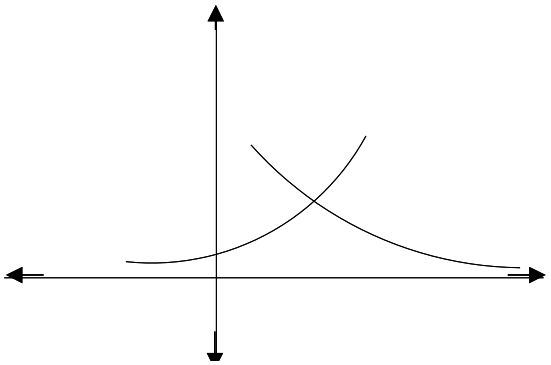

Figure 1. Conflicting functions on $\Re$
(CARLSSON and FULLER. Figure 1. p.4)

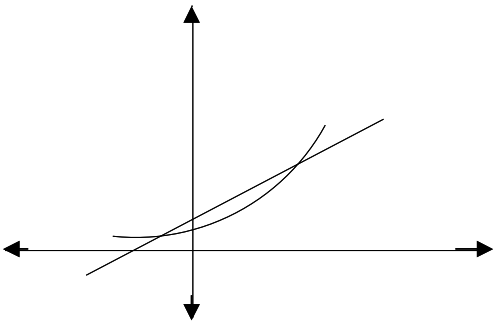

Figure 2. Supportive functions on $\Re$
(CARLSSON and FULLER. Figure 2. p.5)

In traditional MCDM it has been found that the criteria should be independent, yet there are some methods that deal with conflictive objectives but do not recognize other interdependencies that can be present, which makes the problem more unrealistic. Zeleny (1992)[31] recognized that there are *objectives that might support each other* when he shows the *fallacy with using weights independent from criterion performance*.

2.2.2. Multicriteria Optimization Problems

When scheduling problems have more than one objective, they are said to be multicriteria-based. It is important to understand the theory that they have considered to solve these types of problems. The multicriteria optimization theory takes basically a set of priorities established by the decision maker and provides the best solution under their preferences. T'Kindt shows a mathematical definition of the multicriteria optimization problems expressing them as a *special case of*

[31] CARLSSON, Christer and FÚLLER, Robert. Op. Cit. p.12

vector optimization problems where the solution space is S and the criteria space, Z(S), are vectorial euclidian spaces of finite dimension, Q and K respectively[32]*:*

$$Min\ Z(x)\ with\ Z(x) = \left[Z_1(x);...; Z_K(x)\right]^T$$
$$Subject\ to$$
$$x \in S$$
$$S = \left\{x / \left[g_1(x);...; g_M(x)\right]^T \leq 0\right\}$$
$$i.e.\ S \subset \Re^Q \ and\ Z(S) \subset \Re^K \ with\ 1 \leq Q, K < \infty.$$

2.2.3. Definition of Optimality [33]

Let $S \subset \Re^Q$ be a set of solutions and $Z(S) \subset \Re^K$ the image in the criteria space of S by K criteria Z_i. $\forall x, y \in \Re^K$:

$$x \leq y \Leftrightarrow x_i \leq y_i, \forall i = 1,..., K$$
$$x = y \Leftrightarrow x_i = y_i, \forall i = 1,..., K$$

This is valid for $K \geq 2$, because for single criterion problems *(K=1),* there is no way to compare between two solutions, for which the optimal solution is given right away. In the case of multiple objectives, this is no longer the case because there will be various solutions that minimize several criteria and they need to be compared. To approach it, Pareto Optima, a general definition of optimality, is used. There are three types of Pareto optima that have been defined by several authors: weak, strict and proper Pareto optima.

---

[32] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op.Cit. p.46
[33] Ibid. p. 47

Definition of weak Pareto optima:

$x \in S$ is a weak Pareto optimum if and only if $\nexists y \in S$ such that $\forall i = 1,...,K, Z_i(y) < Z_i(x)$. This set of weak Pareto optima, WE, defines the trade-off curve in the criteria space, which is called the efficiency curve. This is the more general class of Pareto optima, the other two types are subsets of this one. See Figure 3 for an example of the set of points that represent the weak and the strict Pareto optima.

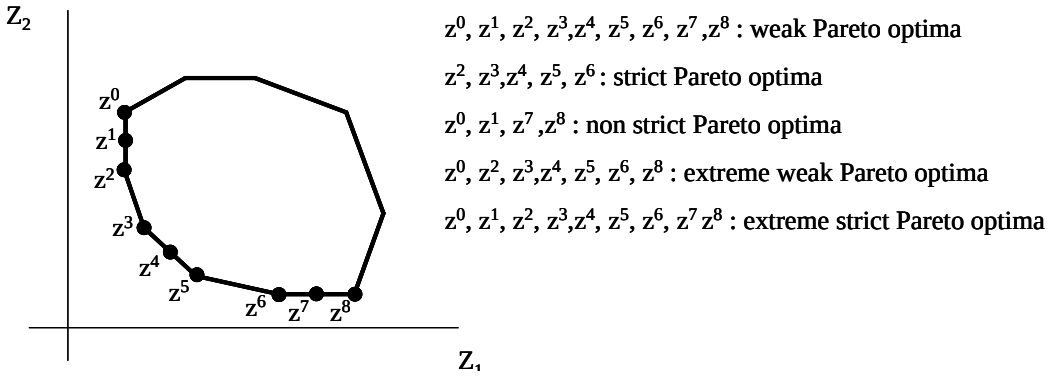

Figure 3. Weak and Strict Pareto Optima where Z defines a polyhedron (T'KINDT and BILLAUT. Figure 3.3. p.48)

Definition of strict Pareto optima:

$x \in S$ is a strict Pareto optimum if and only if $\nexists y \in S$ such that $\forall i = 1,...,K, Z_i(y) \leq Z_i(x)$ with at least one strict inequality. E is the set of strict Pareto optima of S and $E \subseteq WE$.

Definition of proper Pareto optima [Geoffrion, 1968]:

Let $x, y \in S, y \neq x$ *and* $I_y = \{i \in [1;K] / Z_i(y) < Z_i(x)\}$. $x \in S$ is a proper Pareto optimum if and only if x is a strict Pareto optimum and $\exists M > 0$ such that

$\forall\, y \in S, y \neq x, I_y = \varnothing \Rightarrow$

$\forall\, i \in I_y$, ($\exists j, 1 \leq j \leq K$ *with* $Z_j(x) < Z_j(x)$) such that $\dfrac{Z_i(x) - Z_i(y)}{Z_j(y) - Z_j(x)} \leq M$

PRE is the set of proper Pareto optima of S and $PRE \subseteq E$. Notice that this definition is only valid if each criterion Z_i can reach is minimum value.

2.2.4. Determining Pareto Optimality

When reaching for Pareto optima, the decision maker has to look for the "best trade-off" solutions between conflicting criteria, and it is assumed to be done by optimizing a utility function. When searching for the solution, the decision maker must choose for an algorithm or heuristic that can determine the whole Pareto optima set. The decision maker provides weights to the different criteria being analyzed in order to determine the priorities. In literature many ways have been used to determine Pareto optima, it is just a matter of choosing the correct one depending on the *quality of the calculable solutions and the ease of the application*[34].

[34] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.54

Determination by Convex Combination of Criteria [Geoffrion, 1968][35]

Let S be the convex set of solutions and K criteria Z_i convex on S. x^0 is a proper Pareto optimum if and only if $\exists \alpha \in \Re^K$, *with* $\alpha_i \in]0;1[$ *and* $\sum_{i=1}^{K} \alpha_i = 1$ *such that* x^0 *is an optimal solution of the problem (P_α):*

$$Min\ g(Z(x))\ with\ g(Z(x)) = \sum_{i=1}^{K} \alpha_i Z_i(x)$$

$$Subject\ to$$

$$x \in S$$

The above theorem, Geoffrion's Theorem, the parameters α_i cannot be equal to zero because, otherwise, not all the results found will correspond to proper Pareto optima. So another condition is needed to determine a weak Pareto optima:

Let S be the convex set of solutions and K criteria Z_i convex on S. x^0 is a set of weak Pareto optimum if and only if $\exists \alpha \in \Re^K$, *with* $\alpha_i \in [0;1]$ *and* $\sum_{i=1}^{K} \alpha_i = 1$ *such that* x^0 *is an optimal solution of the problem (P_α).*[36]

T'Kindt[37] introduces how graphical representations of the different optimization problems can be done by using *level curves*. For minimizing the convex

[35]T'KINDT, Vincent and BILLAUT, Jean-Charles.. p.54-56
[36] Ibid. Op.Cit *Lemma 2. p.58*
[37]Ibid. p.58

combination of criteria, problem (P_α) can be represented by defining first the set of level curves in the decision space, using the conditions for this specific approach:

$$Let\ X\_(a) = \left\{ x \in S / \sum_{i=1}^{K} \alpha_i Z_i(x) = a\ with\ \alpha_i \geq 0; 1 \text{ and } \sum_{i=1}^{K} \alpha_i = 1 \right\}$$

By writing $L_=(a) = Z(X\_(a))$ in order to construct the curves in the graphs, the curve of minimal value g* is found, where the line $L_=(g^*)$ is tangent to Z in the criteria space. See figure 4 for a geometric representation of the problem described above.

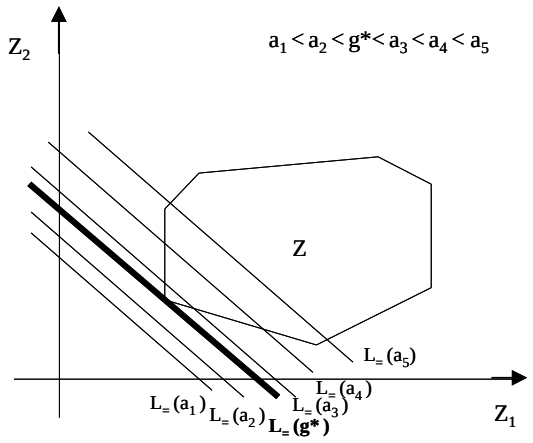

Figure 4. Geometric Interpretation of a problem (P_α)
(T'KINDT and BILLAUT. Figure 3.9. p.59)

Determination by Parametric Analysis [Soland, 1979]

Before stating the theorem that conditions this new method, it is necessary to define what a strictly increasing function is.

A function $f; \Re^K \to \Re$ is strictly increasing if and only if

$$\forall x, y \in \Re^K, x \neq y, x \leq y \Rightarrow f(x) < f(y).$$

Theorem[38]

Let G_Y be the set of strictly increasing functions from $\Re^K$ to $\Re$ which are lower bounded on Z, and $g \in G_Y$. $x^0 \in S$ is a strict Pareto optimum if and only if such that x^0 is an optimal solution of problem $P_{(g,b)}$:

$$\begin{aligned} &Min\ g(Z(x)) \\ &Subject\ to \\ &\quad x \in S \\ &\quad Z(x) \le b \end{aligned}$$

According to T'Kindt, the problem ($P_{(g,b)}$) can be interpreted geometrically by the use of level curves.

$$Let\ S' = \{x \in S / Z(x) \le b\},\ X_=(a) = \{x \in S' / g(Z(x)) = a\}\ and\ \ L_=(a) = Z(X_=(a)).$$

This time, the optimal solution is found by searching for the level curve that has its minimal value g^* such that $L_=(g^*)$ is tangential to Z' in criteria space. So the interception between both spaces $L_=(g^*)$ and Z' defines the decision space of the strict Pareto optima. See figure 5.

[38] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. Theorem 4 p.60

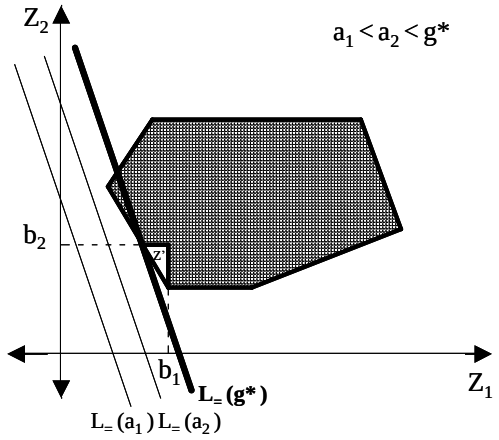

Figure 5. Geometric Interpretation of a problem ($P_{(g,b)}$)
(T'KINDT and BILLAUT. Figure 3.10. p.61)

Determination by Means of the ∈-constraint Approach[39]

This method is used to minimize a criterion assuming that the others (K-1) are upper bounded and it enables the decision maker to find a strict Pareto Optima. The following theorem was proposed in [Yu,1947] (Theorem 5. p.62) and is today frequently used by many authors:

$x^0 \in S$ is a strict Pareto optimum if and only if $\forall k \in \{1;...;K\}\ \exists \varepsilon^k = (\varepsilon_1^k;...;\varepsilon_{K-1}^k;\varepsilon_{k+1}^k;...;\varepsilon_K^k) \in \Re^{K-1}$ such that $Z(x^0)$ is a unique criteria vector corresponding to the optimal solution of the following problem (P_{ε^k}):

[39] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.62

$$
\begin{aligned}
&Min\ Z_k(x)\\
&Subject\ to\\
&\quad x \in S\\
&\quad Z_i(x) \le \varepsilon_i^k, \forall i \in \{1,\ldots,K\}, i \neq k
\end{aligned}
$$

The last theorem is harder to apply because of the *constraint of uniqueness* considered. However, there is another theorem developed by [Miettinen, 1994] (Theorem 6. p.62-63) that does not take this into account, instead, it develops weak Pareto optima rather than a strict one:

Let $x^0 \in S$. If $\exists k \in \{1,\ldots,K\}$ *and if* $\exists \varepsilon^k = (\varepsilon_1^k;\ldots;\varepsilon_{K-1}^k;\varepsilon_{k+1}^k;\ldots;\varepsilon_K^k) \in \Re^{K-1}$ such that x^0 is an optimal solution of the following problem (P_{ε^k}):

$$
\begin{aligned}
&Min\ Z_k(x)\\
&Subject\ to\\
&\quad x \in S\\
&\quad Z_i(x) \le \varepsilon_i^k, \forall i \in \{1,\ldots,K\}, i \neq k
\end{aligned}
$$

According to T'Kindt, this problem can also be interpreted by means of level curves.

Let $k \in \{1,\ldots,K\}$ *and* $\varepsilon^k = (\varepsilon_1^k;\ldots;\varepsilon_{K-1}^k;\varepsilon_{k+1}^k;\ldots;\varepsilon_K^k) \in \Re^{K-1}$.

Let us define $S^k = \{x \in S / Z_i(x) \le \varepsilon_i^k \ \forall i \in \{1,\ldots,K\}, i \neq k\}$, $X_=(a)^k = \{x \in S^k / Z_k(x) = a$

and $L_=(a^*)^k = Z(X_=(a)^k)$.

To solve it, the minimum value a^* must be determined such that $Z(S^k)$ is tangential to $L_{=}(a^*)^k$ in the criteria space. Yet, a value x^* is a strict Pareto optimum if $\forall k, \in S^k \; such\, that \; L_{=}(a^*)^k \cap$ [illegible]. See figure 6.

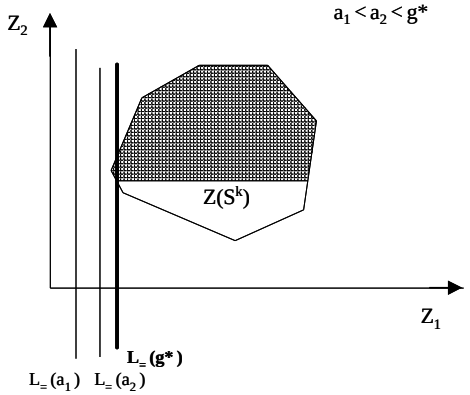

Figure 6. Geometric Interpretation of a problem (P_{ε^k})
(T'KINDT and BILLAUT. Figure 3.12. p.66)

Use of the Tchebycheff Metric[40]

This metric was proposed by [Bowman, 1976] and is practically used to look for the closest solution to a reference criteria vector or reference point*. Before describing this metric, it is necessary to consider some definitions:

i. $z^{id} = [z_1^{id};...; z_K^{id}]^T$ is the **ideal point** if and only if $z^{id} = \min_{x \in S}(Z_i(x)), \forall i = 1,..., K$ [41]

ii. $Let\ K\ vectors\ z^i = [z_1^i;...; z_K^i]^T verifying\ z_i^i = z_i^{id}, \forall i = 1,..., K.$ So a **gains matrix** is defined by $G_{j,i} = z_j^i, \forall i = 1,..., K, \forall j = 1,..., K.$ [42]

[40] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.66
[41] Ibíd. Def. 25 p.66
[42] Ibíd. Def. 26, p.67

iii. Let G be the gains matrix. The **nadir** is defined as a criteria vector, noted z^{na} $defined\,by$ $z^{na} = \max_{i=1,...,K}(G_{j,i}), \forall j = 1,..., K.$[43]

iv. z^{ut} is a **utopian point**, if and only if z^{ut} dominates z^{id} with at least one strict inequality.[44]

1. A **reference point** is known as every vector, noted z^{ref}, which is considered to be an objective to reach. The objetive is to find the closest possible solution to this point in order to optimize the function. Points mentioned above in the last definitions are also considered reference points.

Bowman's definition of the Tchebycheff metric[45].

Let $z^1 \; and \; z^2 \in \Re^K$. The Tchebycheff metric is a measure of the distance in the $\Re^K$ between $z^1 \; and \; z^2$ defined by:

$$\left\| z^1 - z^2 \right\|_T = \max_{i=1,...,K}(\left| z_i^1 - z_i^2 \right|).$$

The Tchebycheff point, noted $(z^* - \theta)$, is a special reference point, such that, $z_i^* = \min_{x \in S^{i-1}}(Z_i(x))$ $with$ $S^i = \{x \in S^{i-1} / Z_i(x) = \min_{x' \in S^{i-1}}(Z_i(x'))\}$ and $S^0 = S. \forall \theta = (\theta_1, \theta_2, ..., \theta_K) \in \Re^K$

Bowman's Theorem[46].

---

[43] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. Def. 27 p.67
[44] Ibíd. Def. 28, p.67
[45] Ibíd. Def. 67. p.67

If $x^0 \in S$ is a strict Pareto optimum then $\exists \theta^* = (\theta^*_2, ..., \theta^*_K) \in \Re^K$ such that x^0 is an optimal solution to the problem (P_θ):

$$Min \left\| Z(x) - (z^* - \theta^*) \right\|_T$$
$$subject\ to$$
$$x \in S$$

The geometric interpretation of the problem P_θ can be done by the use of level curves. Let x^* and θ be fixed, $X_=(a) = \{x \in S / \left\| Z_i(x) - (z^* - \theta^*) \right\|_T = a\}$ and $L_=(a) = Z(L_=(a))$. So by determining the minimal value a^*, such that $L_=(a^*) \neq \emptyset$, thus, the solutions for $X_=(a^*)$ are found. See figure 7.

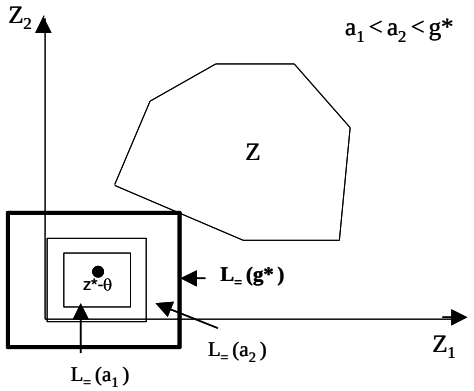

Figure 7. Geometric Interpretation of a problem (P_θ)
(T'KINDT and BILLAUT. Figure 3.13. p.69)

[46] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. Theorem 7. p.67

Determination by Goal-Attainment Approach[47]

This approach is similar to the last one mentioned, but the difference lies on how the solution is searched. This requires for the decision maker to define a goal for the criteria and so it looks for the solution that gets closer to this goal. [Gembicki, 1979] and [Wierzbicki,1990] proposes the following theorem[48]:

$x^0 \in S$ is a weak Pareto optimum if and only if $\exists\, z^{ref} \in \Re^K$ a reference point and $w \in \Re_+^K$, a weights vector such that x^0 is an optimal solution of the problem ($P_{(z^{ref},w)}$) stated below.

$$Max\ g(Z(x))\ with\ g(Z(x)) = \min_{i=1,\dots K}(\frac{1}{w_i}(z^{ref} - Z_i(x)))$$

$$subject\ to$$

$$x \in S$$

A geometric interpretation of this problem is done by T'Kindt by *projecting the point onto the trade-off curve in a direction specified by the weights value* w_i.[49] See figure 8 that describes the case where a solution is found and where no feasible solution is found.

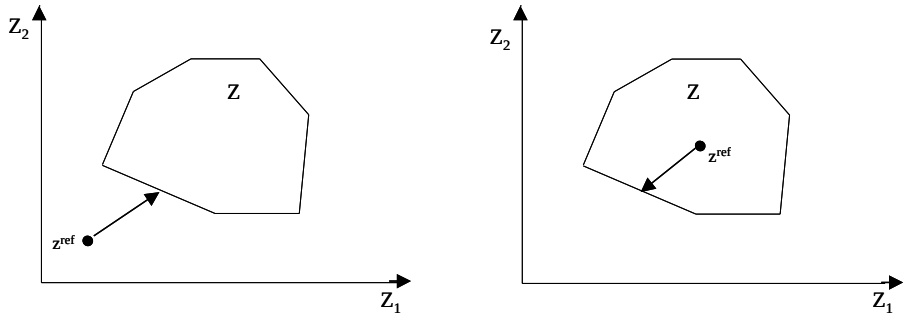

Figure 8. Geometric Interpretation of a problem ($P_{(z^{ref},w)}$) (T'KINDT and BILLAUT. Figure 3.16. p.79)

[49] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit.. p.79

Determination by the Use of Lexicographical Order[50]

This method is used when no trade-off is allowed in the problem, so it is defined according to a *lexicographical order,* $Z_1 \to Z_2 \to ... \to Z_K$*, and noted* $\min_{Lex}(Z)$. In order to obtain a solution to this problem, two conditions must be satisfied: Z_i is lower bounded on each subset S^{i-1} and that $S \neq \emptyset$. So, to determine the optimal solution, x^0, the solution for $x^0 \in S^K$ must be found with:

$$S^1 = \{x^0 \in S / Z_1(x^0) = \min_{x \in S}(Z_1(x))\},$$
$$S^2 = \{x^0 \in S^1 / Z_2(x^0) = \min_{x \in S^1}(Z_2(x))\},...,$$
$$S^K = \{x^0 \in S^K / Z_K(x^0) = \min_{x \in S^K}(Z_K(x))\}.$$

2.2.5. Multicriteria Linear Programming

Even though, the approaches shown in the previous section are used for many different hypothesis made, there is a simpler way to solve them through Multicriteria Linear Programming. The model is presented by T'Kindt[51]:

[50] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.81
[51] Ibid. p.83

$$Min\ Z_1\ with\ Z_1 = \sum_{j=1}^{Q} c_j^1 x_j = c^1 x$$

$$\vdots$$

$$Min\ Z_K\ with\ Z_K = \sum_{j=1}^{Q} c_j^K x_j = c^K x$$

$$subject\ to$$

$$ax = b$$

$$x \in \Re^Q$$

with A representing the coefficients matrix ($M\ x\ Q$) and b representing the constants vector of dimension M. The criterion Z_i is a linear function and so Z is a convex polyhedron defined by the set of solutions S.

Some of the applications of Multicriteria Linear Programming include the determination of strict Pareto optima by convex combination of criteria and by ε-constraint approach.

2.2.6. Multicriteria Mixed Integer Programming

Some of the approaches studied above have a lack of *convexity hypotheses on* Z, which determines that some *non supported* solutions appear. Given this cases, Multicriteria Mixed Integer Programming models the *supported* and the *non*

supported Pareto optima. T'Kindt shows an example to explain the difference between these two terms:

Set Z is the set of points represented and $co(Z)$ is the convex hull defined by Z. We have $co(Z) = \left\{ z \in \Re^K / \forall \alpha_i \in [0;1], \sum_{i=1}^{K} \alpha_i = 1 \; and \; \forall z^i \in Z, \sum_{i=1}^{K} \alpha_i z^i \right\}$. Since x^0 does not belong to the border of $co(Z)$, it is considered to be non supported strict Pareto optima. Also, point x^4 represents a weak Pareto optima. See figure 9 to observe the graphical interpretation of the problem.

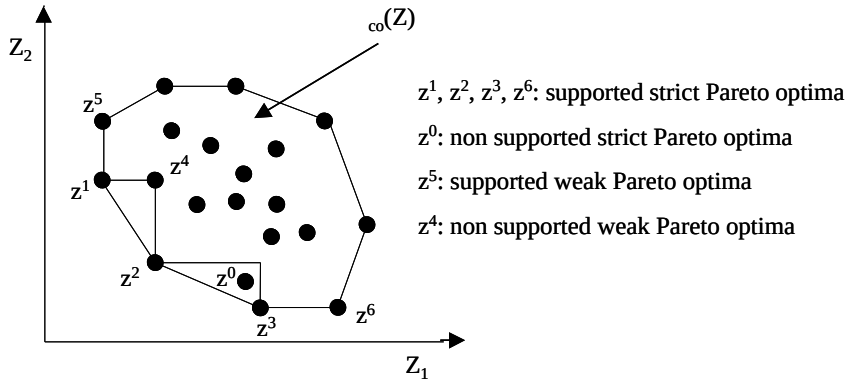

Figure 8. Supported and non supported Pareto optima (T'KINDT and BILLAUT. Figure 3.18. p.85)

The resolution methods like the parametric analysis, the Tchebycheff metrics and the goal-attainment approach, does not present any problem in determining the non supported and supported Pareto optima.

2.3 MULTICRITERIA SCHEDULING PROBLEMS

The principal objective of scheduling is to optimize the objective function started by the problem by defining a schedule that best fits it. The resulting solution corresponds to the Pareto optimum for the multicriteria scheduling problem.

According to the notation presented before[**], the scheduling problems are referred to in a general way by using the three-field notation. The last field,γ, denotes a list of criteria that need to be considered to solve the problems. When there is more than one criterion, this corresponds to a multiple objective problem: $\alpha|\beta|Z_1, Z_2,..., Z_K$, where Z_i is the criterion to be minimized. It is just Z if it is a single criterion problem. For better understanding of this type of problems, a new field is introduced and it corresponds to the resolution methods, studied previously and used to solve these types of problems. Since these resolution methods will be taken in consideration later in the investigation, it is necessary to get acquainted with the notation used[52]:

- $F_l(Z_1,..., Z_K)$, if the objective is to minimize a linear convex combination of criteria.
- $\varepsilon(Z_1,..., Z_{u-1}, Z_{u+1},..., Z_K)$, if the objective is to minimize only the criterion Z_u, by using the ε–constraint approach.

[**] *See section 1.2. Terminology and Notation.*
[52] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.110

- $P(Z_1,..., Z_K)$ if the objective is to minimize by using parametric analysis.
- $F_T(Z_1,..., Z_K)$ if the objective function is a distance known as an ideal solution and calculated by Tchebycheff metric.
- $F_s(Z_1,..., Z_K)$ if the objective is to minimize by using the goal-attainment approach.
- $Lex(Z_1,..., Z_K)$ indicates that no trade-off is authorized so they must order the criteria beginning with the most important one.

Kumar, Marathe, Parthasarathy and Srinivasan have implemented approximation algorithms for scheduling on multiple machines in order to solve a bi-criteria problem based on minimizing makespan and weighted completion time. They proposed a single randomized rounding algorithm that combines the power of LST and randomization in order to obtain a simultaneous optimization of multiple objectives. With this, they obtain a (2, 3/2) bi-criteria approximate algorithm for makespan and weighted completion time.

2.4. GAME THEORY

In game theoretic literature much has been said with regards to many types of games and each one of them has approached an application to the different areas of study. All the theory that surrounds it makes it a very dynamic and extensive

field. Yet, it is necessary to know the fundamental aspects of game theory and for this there are various authors that had contributed to the understanding of it.

The main objective in game theory is to develop rational criteria in order to decide over two or more strategies. There are two basic assumptions given for this: players are rational thinkers and players choose their strategy to maximize their own benefit.

2.4.1. Two person zero-sum game and Nash Equilibrium

One of the simplest forms of a game is the one that involves two players and whose sum of the utilities is equal to zero, sometimes referred to as *strictly competitive games*[53], where $\sum_{i=1,2} u_i(s) = 0$ for all s. The non-zero sum games may be more practical in many applications; yet, for purposes of the analysis shown, it is important to understand this type of a game before going any further. The benefits for each player are shown in a matrix, in the form of payoffs. Usually the payoffs are positive to show earnings and negative to show losses.

It is necessary to note that the games considered are finite games, which means that each player's set of strategies is finite. There are several ways to approach these types of problems depending on the situation of each player in terms of

[53] OWEN. Op.Cit. p.11

strategies and respective payoff functions. To illustrate this in a better way, an example is described in Fudenberg and Tirole's book, for which players 1 and 2 have three pure strategies each. Player 1 has strategy U,M,D (upper, middle and

	L	M	R
U	4,3	5,1	6,2
M	2,1	8,4	3,6
D	3,0	9,6	2,8

down) and player 2 has L, M, R (left, middle and right). The chart above shows the resulting matrix.

Each player has one strategy to choose, yet, sometimes the player can choose more than one strategy. When this possibility is contemplated, then the payoff for the players can be estimated rather than fixed. The payoff of player I to a mixed strategy profile is given by the following expression:

$$\sum_{s\in S}\left(\bigcap_{j=1}^{I}\sigma_j(s_j)\right)u_i(s)$$

In the example above, the vector representing the mixed strategy of player 1 is $(\sigma_1(U),\sigma_1(M),\sigma_1(D))$. The profiles for each player are: $\sigma_j=\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$ and $\sigma_j=\left(0,\frac{1}{2},\frac{1}{2}\right)$. The payoff for this given profiles, can be calculated as shown below:

$u_i(\sigma_1\sigma_1)=\frac{1}{3}(0\cdot 4+\frac{1}{2}\cdot 5+\frac{1}{2}\cdot 6)+\frac{1}{3}(0\cdot 2+\frac{1}{2}\cdot 8+\frac{1}{2}\cdot 3)+\frac{1}{3}(0\cdot 3+\frac{1}{2}\cdot 9+\frac{1}{2}\cdot 2)=\frac{11}{2}$ and

$u_i(\sigma_1\sigma_2)=\frac{27}{6}$.

But, before going any further on mixed strategies, there are various ways of obtaining optimal strategies. Starting with the simplest form, this is by detecting those dominated strategies in the matrix for each player. Given the last example, note that for player 2, R gives a higher payoff than M does, no matter what player 1 chooses:

	L	R
U	4,3	6,2
M	2,1	3,6
L	3,0	2,8

Likewise, for player 1, U will give a higher payoff for both M and L:

	L	R
U	4,3	6,2

At this point of the game, player two is able to choose the strategy that best satisfies his needs, the strategy that gives him/her the greatest utility; in this case, it is L. So the pair of strategies chosen by both players are: U for player 1 and L for player 2, representing for them a payoff of 4 and 3 respectively. In this case, they have pure and strictly dominated strategies, where solutions are independent and in equilibrium, which means that the solution will be always (4,3) no matter what each player does independently. It is important to note the definition for Nash Equilibrium, which is introduced by the famous Noble price winner, John Nash. Fudenberg and Tirole[54] present it as definition 1.2

A mixed-strategy profile σ^* is a Nash equilibrium if, for all players i,

[54] FUDENBERG, Drew and TIROLE, Jean. Op.Cit p.11

$$u_i(\sigma^*, \sigma_{-i}^*) \geq u_i(s_i, \sigma_{-i}^*) \ \textit{for all}\ s_i \in S_i$$

For pure strategies, it satisfies the same conditions as the mixed strategies, only that the probabilities can only take values of 0 or 1. Harsany (1973b)[55] introduces a strict Nash Equilibrium, where each player has a *unique best response to his rivals' strategies.* So the pure strategy s^* is strict for all i and all $s_i \neq s_i^*, u_i(\sigma^*, \sigma_{-i}^*) > u_i(s_i^*, \sigma_{-i}^*)$.

Notice that this strict equilibrium happens only with pure strategies. It seems that the strict equilibria are more compelling than the equilibria where players are indifferent to their equilibrium strategy and even to a non-equilibrium response. It is also said that due to various small changes in the nature of the game, strict equilibria are robust.

In general, Nash Equilibria give reasonable predictions to how a game is played and it is the only one that has the property of common knowledge between players. There is no incentive to play differently when a game has Nash Equilibria because players can detect it. On the other hand, in a non-Nash profile, players can make "decision mistakes" during the optimization of their own payoff function or in the prediction of the other's possible moves. This type of mistakes is the reason why most economic applications of game theory restrict Nash equilibria.

2.4.2. Resolution methods for non-dominated strategies

[55] FUDENBERG, Drew and TIROLE, Jean. Op.Cit. p.11-12

Not many applications have dominated strategies, and there are still other solutions that can still be Nash Equilibria. There are two ways to solve a 2 by 2 game when there is no *iterated dominance*. The first one is to search for saddle points, which are what we already know as Nash Equilibria. The second way is to find the mixed strategies, which have to be found if there is no saddle point found. To find the saddle points, the Minimax Theorem is introduced. The Minimax Theorem is proven by Von Neumman and Morgenstern and is the most important in game theory. Owen presents the definition of this theory, but before defining the theory it is necessary to know some additional terms needed to understand the definition.

Given a matrix game A,

$A = \begin{pmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} \end{pmatrix}$, where a_{ij} represents is the payoff to each player for choosing strategy s_i while the opponent chooses the strategy s_j, there is v_1 that represents the "gain-floor" of player 1 and v_2 represents the lost-ceiling" of player 2. These values are defined by Owen[56] as:

$v_1 = \max_i \min_j a_{ij}$ and $v_2 = \min_i \max_j a_{ij}$

[56] OWEN. Op.Cit. p.14

So to find the saddle points player 1 should not win less than v_1 and player 2 should not lose more than v_2, satisfying this condition: $v_1 < v_2$..

Minimax Theorem

For any function $F(x,y)$ defined on any Cartesian product X X Y,

$\max_{x \in X} \min_{y \in Y} F(x, y) \leq \min_{y \in Y} \max_{x \in X} F(x, y)$. Hence we have, $v_1 \leq v_2$.

This theorem is used to find other possible solutions that cannot be treated with *iterated dominance*. The following example of a game matrix is given to show this:

	L	R	min(1)
U	1,6	6,5	1
L	5,2	**2,4**	2
min(2)	2	4	

In this example, by getting the maximum value of the minimum of each column and of each row, it is possible to reach for a saddle point. In this example, one saddle point is found, yet there are other possibilities.

	L	R	min(1)
U	**1,1**	6,2	1
L	5,3	**1,1**	1
min(2)	1	1	

	L	R	min(1)
U	3,2	6,2	3
L	4,3	5,1	4
min(2)	2	1	

The first game matrix is shown below, describes a situation where 2 saddle points are found. The second game matrix describes a game with no saddle point; in this case, it is necessary to find the mixed strategies.

2.4.3. Mechanism design

Mechanism design is a subfield of microeconomics and game theory, which is used to obtain an optimal system-wide solution to a decentralized optimization problem with multiple self interested agents, each with private information about their preferences. In a mechanism design problem, an agent is asked to "input" their confidential information to the system and this one, in response, provides an action and an outcome, accompanied by an incentive to promote truth-revelation in their participation, in order to reach for an optimal solution.

In order to understand how the mechanism works, it is important to recall some notations used. The way an agent recognizes its preferences, a type must be declared. Let $\theta_i \in \Theta_i$ denote the type of an agent i, from a set of possible types θ_i. Let $u_i(o,\theta_i)$ denote the utility of agent i for the outcome $o \in \mathrm{O}$, given type θ_i. For the agent to choose for a course of action, it must have a set of strategies to choose from. Let $s_i(\theta_i) \in \sum_i$ denote the strategies of agent i given type θ_i, where $\sum_i$ is the set of all possible strategies available to the agent. Let $u_i(s_1,...,s_I,\theta_i)$ denote the utility of agent i at the outcome of the game, given preferences θ_i and strategies profile $s_i(s_1,...,s_I)$ selected by each agent.

There are three solution concepts used for solving these particular type of problems; two of them have already been introduced earlier in this chapter, Nash equilibrium and dominated strategy equilibrium, a third one, Bayesian-Nash equilibrium is also being used. According to Nash equilibrium, every agent maximizes its utility with strategy s_i, given its preferences and the strategy of the other agents. *To play Nash equilibrium in a one-slot game, every agent must have perfect information about the preferences of the other agent, agent rationality must also be common knowledge*[57]. A robust solution concept is the dominated strategy equilibrium, where each agent has the same utility-maximizing strategy, for all strategies of the rest of the agents. It does not make any assumptions about the information handled by the agents and does not require an agent to believe the other agents behave rationally to choose its own strategy. In mechanism design, dominant strategy implementations of social choice functions are much more desirable than Nash implementations. The last concept, the Bayesian-Nash equilibrium, in comparison to Nash equilibrium, agent i's strategy $s_i(\theta_i)$ must be a best response to the distribution over strategies of other agents, given the information of their preferences in a distributed function. This type of solution makes more reasonable assumptions about agent information than Nash equilibrium, but a weaker solution concept compared to the dominant strategy equilibrium. These three solution concepts are applicable both for static and dynamic games; in static games, every agent chooses its strategy simultaneously,

[57] PARKES, David. "Chapter 2. Classic Mechanism Design". Harvard University. p. 23-61 (Pdf document via www) <http://www.eecs.harvard.edu/~parkes/pubs/ch2.pdf>

and in dynamic games, actions are based on observation and learning from other agents preferences throughout the course of the game.

The system-wide goal then is defined with a social choice function, $f : \theta_1 x ... x \theta_I$, which selects the optimal solution, $f(\theta)$, given the agent types, $\theta(\theta_1,...,\theta_I)$. At first, the mechanism, $M = (\sum_1,...,\sum_I, g(\cdot))$, defines a set of strategies available $\sum_i$ and the method used to select the final outcome based on agent's strategies.

According to game theory, the mechanism implements social choice function $f(\theta)$ if the outcome computed with equilibrium agent strategies is a solution to the social choice function for all possible agent preferences. This equilibrium concept may be Nash, Bayesian-Nash, dominant or any other. The social function has many properties, for example, it is *Pareto optimal* if no agent can ever be happier without making at least one other agent less happy, it is efficient if it maximizes the total value over all agents, it can also be budget balanced so no net transfers out or into the system. Both allocative efficiency and budget-balance imply Pareto optimality.

The type of mechanism may vary depending on its properties, for example, the agent's preferences may be described by a quasilinear function if their utility is decomposed into a valuation function that depends on a choice rule and a payment function which is assigned based on the strategy profile. The mechanism is not just subject of the agent's preferences, also on the equilibrium concepts and

participation conditions (individual-rationality is applied those players outside the mechanism). Particularly, the direct revelation mechanism is characterized by an incentive compatibility property; this means that agents report the truthful information about their preferences in equilibrium, out of its own self interest. Their strategy is to report a type $\hat{\theta}_i = s_i(\theta_i)$, based on its actual preferences θ_i. The outcome specification is given by a positive real valued objective function $g(o,\theta)$. The required output is the outcome $o \in F$ that minimizes g. The direct revelation mechanism characterizes in the implementation of dominant strategies, which means that it is *strategy proof*. Yet, the case can be such that the solution is obtained from Bayesian-Nash equilibrium, but this only happen if $s_i * \theta_i = \theta_i$, where every agent's expected utility maximizing strategy in equilibrium with every other agent is to report its true profit.

VCG (Vickrey-Clark-Groves) mechanism is applied to mechanism design optimization problems where the objective function is simply the sum of all agents' valuations and implements dominated strategy equilibrium solutions. When introducing transfers, it depends on the characteristics of the mechanism. If quasilinear preferences are assumed, then the transfer function, t_i, takes part of the utility function, $u_i(o,\theta_i) = v_i(k,\theta_i) - t_i$. In order to implement an efficient outcome, $k^* = \max_k \sum_j v_i(k,\hat{\theta}_i)$, and compute transfers,

$$t_i(\hat{\theta}) = \sum_{j=i} v_j(k^{-i},\hat{\theta}_j) - \sum_{j=i} v_j(k^*,\hat{\theta}_j)$$

Where $k^{-i} = \max_k \sum_{j=i} v_j(k, \hat{\theta}_j)$.

This transfer function must guarantee both an optimal strategy and a balanced budget. But it has been shown that for this mechanism it is impossible to implement a solution in dominant strategies and satisfy balanced budget constraint for every possible message profile. A simple way to solve this budget balancing problem in dominant strategies is to introduce an extra agent to the mechanism, "agent 0", whose preferences are known and has no preferences over the solutions, and whose only interest relies on the transfers, $u_0 = t_0(\theta)$. Agent 0 will collect all the payments of the agents so if, $t_0 *(\theta) = -\sum_{i \in N} t_i(\hat{\theta}_j)$, then this mechanism guarantees both a balanced budget and a selection of an optimal solution.

2.4.4. Game Theory and Computer Science

Game theory has been continuously used in the branch of computer science that can be observed as simple interpretations of zero-sum games for analyzing problems in online computation to more complex aspects of game theory in artificial intelligence. Agent-based simulation is been advancing in the area of computation and best describes how game theoretic principles are beneficial to their models.

As observed in mechanism design, the figure of "agents" is used to model objects with some degree of decision, whose actions depend on these decisions. Luis

Mateus Rocha in his research project[58] as part of the Complex Systems and Applications Group in New Mexico, United States, referred to agents as an *entity* that *must be able to step out of the dynamics of an environment, and make a decision about what action to take next:*

> Since choice is a term loaded with many connotations from theology, philosophy, cognitive science, and so forth, I prefer to discuss instead the ability of some agents to step out of the dynamics of its interaction with an environment and explore different behavior alternatives. In physics we refer to such a process as *dynamical incoherence* [Pattee,1993]. In computer science, Von Neumann, based on the work of Turing on universal computing devices refer to these systems as *memory-based systems*. That is, systems capable of engaging with their environments beyond concurrent state-determined interaction by using memory to store descriptions and representations of their environments. Such agents are dynamically incoherent in the sense that their next state or action is not solely dependent on the previous state, but also on some (random-access) stable memory that keeps the same value until it is accessed and does not change with the dynamics of the environment-agent interaction.

This is how agents have been defined as part of computational models, yet aspects can become interesting when analyzing how these models are based on game theoretic strategies, where the model aims to study only the decision strategies and evolution of the strategies over time. They also follow a synchronous behavior, which means that all agents are updated simultaneously and there is an outcome as part of this behavior. The iterated Prisoner's Dilemma is an example of an idealized model for many real-world phenomena, like the *arm-races* and *evolutionary biology*. It consists of 2 individuals which are arrested together but placed in separated rooms. As they are questioned, no communication is allowed between them, but they are offered to testify against each other. If one betrays the other, he gets a suspended sentence while the other gets the whole sentence. If both testify against each other, the testimony is

[58] MATEUS ROCHA, Luis. "From Artificial Life to Semiotic Agent Models." Los Alamos, NM: Los Alamos National Laboratory. October 1, 1999. (Pdf document).

discredited and they both get a high sentence. Yet, if they decide not to testify, they both get a smaller sentence. This model is defined as a non-cooperative game of 2 players, each one with 2 strategies each: to betray or to not betray. The iterated Prisoner's Dilemma (IPD) game has been widely used by economists and other researchers to discover *the potential emergence of mutually cooperative behavior among non-altruistic agents*[59]. This game typically assumes that individual players have no control over who they play with; instead, they are modeled by a mechanism, where randomness is implemented as part of the simulation. One of the most important conclusions reached by these studies has been that the mutual cooperative behavior can be reached on the long-run, a pretty large or infinite number of iterations. This is given by the sufficiently large frequency of mutual cooperative matches and the perceived high probability of future interactions. The researcher Tesfatsion remarked in his paper about the IPD game:

> In actuality, socio-economic interactions are often characterized by the preferential choice and refusal of partners. The question arises whether the emerging and long-run viability of cooperative behavior in the IPD game would be enhanced if players were more realistically allowed to choose and refuse their potential game partners. (…) The traditional IPD game is extended to an IPD/CR game in which players choose and refuse partners on the basis of continually updated expected payoffs.[60]

A more simply type of game, the zero-sum game is used in computer science to model what is known as "demonic" nondeterminism, which is based on choosing the worst possible outcome when there is no sufficient information about future events. Randomization algorithms are used with this model in order to analyze

[59] TESFATSION, Leigh. "How Economists Can Get Alife: Abbreviated Version." Iowa. September 6, 2006. (Pdf document)
[60] Ibid.

problems of *online computation*, which describes a situation were individuals have to input data at the same time and with this information, decisions are made.

2.5. APPROACHES OF GAME THEORY IN SCHEDULING

In literature, there is very little written about the possible game theoretic interactions made in game theory. Yet some fairly recent papers have introduced on the topic. Authors like T.C. Lai and Y.N. Sots (1999) propose a way to search for a "*minimal set of certain schedules*"[61] through the use of game theory. For this, they propose a number of scheduling problems that need of the best expected processing times, which are under the control of a decision maker. At each decision point of the scheduling problem, a two-person zero sum game with the decision maker being player 1 and nature being player 2. Other authors like Serafini[62] mention game theory as a way to reach for *optimal and non-dominated solutions*, and consider specifically the objective of minimizing the maximum tardiness of the jobs whose completion times can be known in advance. He also mentions that in mathematical programming, this type of approach is named *Unordered Lexico Optima*.

[61] LAI, T.C. and SOTS, Y.N. "Sequencing with uncertain numerical Data for Makespan". USA: The Journal of Operational Research Society, 1999. Vol. 50, No.3. pdf document. p.1
[62] SERAFINI, Paolo. "Scheduling Jobs on several machines with the job splitting property". USA: Operations Research, 1996. Vol.44, No.4. pdf document. p. 620

Yet, one of the most interesting contributions was presented by Kutanoglu and Wu, in their papers, *An Incentive Compatible Mechanism for Distributed Resource Planning* and *On Combinatorial Auction and Lagrangean Relaxation for Distributed Resource Scheduling.* In the first paper they implement a mechanism design problem, where job agents' are considered to represent the jobs and their preferences may be motivated by any constraint considered "local", like delivery requirements. They define the game as an n-person non-cooperative game with incomplete information, *where the n players are considered to be the job agents*, where each one has a strategy to choose and for each decision, a utility function is assigned. In the previous section, the mechanism design procedure was explained, yet the goal of the mechanism in these type of scheduling problems is to choose a particular function using the outcome function $h()$ for a particular realization of agents' utility functions in order to choose an optimal or socially efficient schedule y^*. This procedure is also known as "schedule selection game". Kutanoglu and Wu used a resource allocation problem to illustrate this approach. The local constraint used was the job due date and two simplifications were made of the problem, it was decomposed in a series of single machine problems and set up times could be added. First the mechanism created some candidate schedules using Lagrangean-based auction theoretic algorithm. The utility was considered as the negative value of weighted tardiness and the agents' performance depends only on its job allocation in a schedule and its transfer. The second paper describes how local decision makers base their idea on local utility which means that their problem is to

maximize their expected reward subject to local constraints. The type of coordination is known as a *"bid"* where the auctioneer is a bid processor that makes resource allocation in form of an auction processing using *"bidding information"*.

These approaches of game theory to scheduling have been closely related to the area of computer science and in this area, two authors have contributed in an extraordinary way, Ronen and his professor Nisan, following the techniques of mechanism design and applying it to task allocation problems, especially contribute in the computational possibilities of these mechanisms. A formal model is introduced by them for studying optimization problems, in order to observe how mechanism design can be applied to several of these problems.

> *The model is concerned with computing functions that depend on inputs that are distributed among n different agents. A problem in this model has, in addition to the specification of the function to be computed, a specification of the goals of each of the agents. The solution, termed a mechanism, includes, in addition to an algorithm computing the function, payments to be handed out to the agents. These payments are intended to motivate the agents to behave "correctly".*[63]

They defined a task allocation problem with k tasks that need to be allocated on n agents. Each agent type i is, for each task j, the minimum amount of time t_j^i the agent is capable of performing this task in. The goal is to minimize the makespan. The valuation of each agent is the negation of the sum of the times it has spent on

[63] NISAN, Noam. "Algorithms for Selfish Agents: Mechanism Design for Distributed Computation". Jerusalem: Institute of Computer Science. (pdf document)

the tasks allocated to it. They denoted the direct revelation mechanism $m(x, p)$, where $x=x(t)$ is the allocation algorithm and $p=p(t)$ is the payment. They studied this task scheduling problem and designed an n-approximated mechanism, where n is the number of the agents; he proved that a lower bound of 2 to the approximate ratio that can be achieved by any mechanism (for the case of two agents); and finally designed a *randomized mechanism*[*] that beats the determined lower bound. Nisan and Ronen have shown that worst case behavior can be improved using randomness without weakening the "game theoretic" requirements of the mechanism. Finally they came up with a *Second Chance Mechanism*, where the agents are allowed, besides declaring their types, to declare an appeal function where the mechanism is able to compute a better possibility for each agent. In order to work out this mechanism, an algorithm k must be defined by the mechanism for the corresponding optimization problem, in order to produce the best result for each agent. After an iteration is done, the agent can modify this appeal function or it can be automatically done. Either way, the importance of this method lies on the fact that each agent is able to get the best from two solutions, depending on the situation.

Many authors have contributed to the mechanism proposed by Nissan, according to specific applications and situations. Koutsoupias and Papadimitriou initiated investigations on the coordination ratio, which is the relation between the cost of

[*] *A Randomized mechanism is a probability distribution over a family of mechanisms, all sharing the same set of strategies and posible outputs.*

the worst possible Nash equilibrium and that of the social optimum. Specifically, they showed that for two identical machines, the worst-case coordination ratio is exactly 3/2. The task allocation model they proposed is *considered a problem of allocating scheduling tasks to machines according to game theoretic assumptions*. Each task is considered a single entity and cannot be split to be assigned in a part to different machines. In this type of problem, the parameter to be investigated is that of the maximum cost associated with any machine. It is similar to a routing flow model where tasks are to be routed efficiently using a classical makespan minimization scheduling problem, each task can be assigned to a single machine but the decision to which machine it is assigned to, is determined by the user's strategy. If the task is scheduled deterministically to a machine in [m], then it is a pure strategy, but if it is allocated by some probability distribution it follows a mixed strategy. For identical tasks and machines, a balance deterministic allocation is considered where each task i is allocated to a machine $(i \bmod m) + 1$ in order to reach Nash Equilibrium. It has been shown in literature that it is NP-hard to find the best and the worst pure Nash equilibria, but there exists a polynomial time algorithm that computed, for any given task allocation problem: a Nash equilibrium with no higher cost. Moreover, the existence of a PTAS (polynomial-time approximation scheme) for the problem of computing Nash equilibrium with minimum social cost is demonstrated. Yet, all algorithms cited above have an undesirable property: they are centralized and off-line. Recently, there has been some research on non-centralized algorithms for finding Nash equilibria.

The problem is described as a *load balancing process*, where each task is reallocated according to a selfish rule, by defining some strategies. It is assumed that the task is reallocated in a single step. For the identical machine case, Even-Dar[64], et al., found that if one moves the first the maximum weight task in a machine with a minimum load, then Nash equilibrium can be reached. Their approach considers a game of many players (jobs) and actions (machines) and studies their asymptotic behavior. During this game, jobs are allowed to select a machine to minimize their own cost. The cost that a job observes is determined by the load of the machine, which is the sum of the weights of the jobs running on it. During this process, at least one job is willing to change to another machine, until Nash Equilibrium is reached. Only one job is allowed to move in each step and it is the centralized controller that selects which job will move in the current time step. The strategy used by the controller is the algorithm used to select which of the computing jobs will move. Since all jobs behave selfishly, it is assumed that when a job migrates, the observed load on the machine is strictly reduced, which we refer to as a *best-reply* policy, otherwise it is an *improvement* policy. In the case of identical machines, they proved that if one moves the minimum weighted task, the convergence may take place in exponential number of steps, otherwise, if one moves the maximum weighted task and this one follows the best reply policy, Nash

[64] EVEN-DAR, Eyal, KESSELMAN, Alex and MANSOUR, Yishay. "Covergence Time to Nash Equilibrium". Tel-Aviv University. (PDF document) p.18

equilibrium is reached in at most n steps. This is one of the reasons why it is important to choose the “right” scheduling strategy.

Recent works had been using the mechanism of load balancing but without using a centralized control system. Berenbrink[65], et al., in their paper, *Distributed Selfish Load Balancing*, they discussed a natural protocol for the agents which was implemented in a strongly distributed setting, without any centralized controller and with good convergence properties. In each round, the load of each task from the current machine was being compared to that of a randomly chosen machine and if the observed load of the other machine was less than that of the current machine, then the job automatically moved. The following procedure shows the steps already described:

For each task b do in parallel

Let i_b be the current resource of task b

Choose resource j_b uniformly at random

Let $X_{ib}(t)$ be the current load of resource i

Let $X_{jb}(t)$ be the current load of resource j

If $X_{ib}(t) > X_{jb}(t) + 1$ then

Move task b from resource i_b to j_b with probability $1-X_{jb}(t)/ X_{ib}(t)$

[65] BERENBRINK, Petra, et. al. “Distributed Selfish Load Balancing”. Canada, Natural Sciences and Engineering Research Council of Canada. May 2, 2006. (PDF document) p.17

The advantage of this protocol is that it is very simple and there is no need of global information, tasks did not even need to know the total number of tasks being allocated, they only needed to know their observed load on each of the resources.

3. OBJECTIVES

3.1. GENERAL OBJECTIVE

To use the principles established in Game Theory in order to obtain solutions for parallel machine scheduling problems under multiple criteria and test its effectiveness in production decision making by comparing them to preexisting heuristics and algorithms used under multicriteria scheduling.

3.2 SPECIFIC OBJECTIVES

- To understand the importance of Game Theory in the context of productive systems programming by determining the sequence of jobs that can give a quick and robust solution to scheduling problems involving parallel machines, in order to generate alternate solution sources for these types of configurations.
- To prove whether Game Theory can approach scheduling problems under multiple objectives, by establishing schedules through the switching of jobs according to trade-offs among intelligent agents within the system.
- To establish comparisons with other heuristics that do not use game theoretic principles, in order to test the robustness of Game Theory in this field, through the contrast of solutions generated by those other heuristics and the ones attained by Game Theory.

4. SCOPE AND LIMITATIONS

4.1. SCOPE

Allocating resources in dynamic environments is a very wide topic and can be approached from many different perspectives. This investigation is focused on parallel machines related problems. There are many problems that arise from different environments up to this, including flexible flow shops. Literature on flexible flow shops is based mostly on TOC approaches, thus these are almost always treated focusing on the bottlenecks. Within these bottlenecks further analysis can be made when realizing that a working center may be considered as a group of parallel machines. Hence, this research will not focus on the whole flexible flow shop frame but on same based procedure stations parallel arranged.

There are also several problems that arise from parallel machines. Those include identical machines, related machines, and unrelated machines. This investigation will not consider related and unrelated machines focused problems. Its main scope will be within the identical machines consideration.

Another typical classification for these kinds of problems will be the consideration of preemption and consideration of dividing a single job into parts so they can be thoroughly processed once at a time by splitting the job among all available

machines. Literature includes algorithms for preemption consideration and also when preemption is not allowed. But as our approach is mainly on tradeoffs, and job switching among machines, we will not be able to consider preemption. This is clearly a limitation since multicriteria scheduling theory is able to improve the main variables within a parallel machine environment by allowing preemption on jobs, as seen in section 2.11 (Literature Review on Parallel machine scheduling problems concerning completion time and makespan for schedules when preemption is allowed).

4.2. LIMITATIONS

Assumptions of independent setup times, release times for all jobs considered equal, not considering due dates nor deadlines, not taking into account the unrelated machines problems, limit our research, however the highest involved constrain lies in the complexity that scheduling problems may have, especially when they are NP hard, and literature is not well illustrated by effective allocation algorithms. So our comparisons once we generate new schedules have to be limited to a series of data gathered from results found in the papers such as the papers by Gupta and Ho[66], and Archer[67], Amir Ronen[68], Noam NISAN[69].

[66] GUTPA and Ho . "Bicriteria optimization of the makespan and mean flow time on two identical parallel machines"
[67] ARCHER, Aaron, TARDOS Eva "Truthful Mechanisms for One-Parameter Agents

Another limitation is the isolation we might give to the perspective of the problems, given that the considered or main focused approaches will be made just for parallel machine configurations. It is known that most systems have more than one station, and could be considered a flexible jobshop, however these might bring specific approaches, and thus these considerations might make feasible other investigations that consider more general flexible jobshops.

The convergence of game theoretic approaches to consider the problem of allocating resources in a scheduling environment is almost a new approach, direct containing literature might not be entirely available; nevertheless, further papers and articles have been published in the internet, containing these considerations. As mentioned in the literature review, problems have been focused using job agents, machine agents, online and offline mechanisms with a centralized or distributed system. In many of these papers convergence to Nash equilibrium and comparisons were made with other type of game theoretic solutions such as dominant equilibrium and Bayesian Nash equilibrium. Solutions in this investigation will be limited to Nash and dominated solutions.

---

[68] RONEN, Amir. "Solving Optimization Problems Among Selfish Agents". Jerusalem: Hebrew University, 2000. (PDF document) 92 p.

[69] NISAN, Noam. "Algorithms for Selfish Agents: Mechanism Design for Distributed Computation". Jerusalem: Institute of Computer Science. (PDF document) 17p.

A specific limitation that may be present when allocating jobs lies in the fact that our model is a conservative one, first of all it is based on Nash Equilibrium, and secondly we allow jobs to move to the previous position, this means it only considers trades of one timeslot per job, so changes may not be as drastic.

5. HYPOTHESIS

The following hypotheses were identified according to the objectives and problem formulation; of course all of these, within the literature review context.

5.1. SET OF HYPOTHESES

INVESTIGATION HYPOTHESES

- The scheduling function interaction with game theory approaches can lead to generate alternate solutions that may simplify the decision making process by reducing the complex sample space of schedules in parallel machine related configurations and provide a range of solutions that belong to the Pareto Front, thus, represent an effective combination of criteria for the sequencing of jobs.
- The Pareto Front set of points obtained from the game theory approach complement those obtained through the classical multicriteria techniques.

NULL HYPOTHESES

- The scheduling function interactions with game theoretic approaches do not lead to Pareto Front points that represent an effective combinatorial criteria tradeoff that may simplify the decision making process, and thus do not generate alternate solutions.

- The Pareto Front set of points obtained from the game theory approach do not complement those obtained through the classical multicriteria techniques.

ALTERNATE HYPOTHESES

- The principles of game theory will lead to a Pareto Front set of points, yet they are not always the best ones due to the complex sample space of possible solutions, but can be considered for future research.
- The Pareto Front set of points obtained from the game theory approach represent a set that approximates that obtained through classical multicriteria approaches but may differ due to randomness involved in the process.

5.2. CONCEPT VARIABLES DEFINITION

The main variables of the problem are obtained from the set of hypothesis, those include:

- T^{SG} (Scheduling and game theory technique rules)→ scheduling and game theory interaction techniques allocation rules for approaching multicriteria parallel machine problems.

- T^{MC} (Other multicriteria technique rules) → allocation rules for other different approaching techniques concerning multicriteria parallel machine problems.
- g →Game theory tools.
- s →Scheduling for parallel machine elements.
- P (Set of points in a Pareto Front) →Set of points that belong to a Pareto Front which represent robust solutions.
- P^{MC} → Set of Pareto points that result from multicriteria approaching techniques.
- P^{SG} → Set of Pareto points that result from game theory approaching techniques.

1.3. OPERATIONAL VARIABLES DEFINITION

In order to determine how the hypotheses above are affected, it is necessary to establish the dependence and interaction among the variables described, that is in terms of correlation.

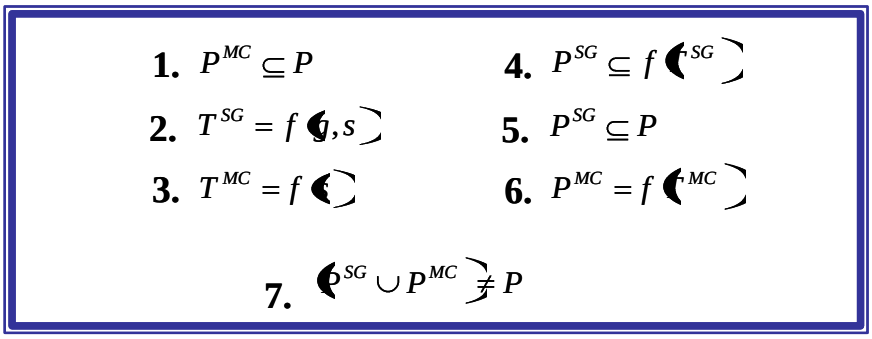

Figure 10. Main variables definition.

Reasons for these results:

1. The set of Pareto points that result from multicriteria approaching techniques are subset of all points that represent all possible Pareto Front.
2. The scheduling and game theory techniques depend highly on what can be obtained from game theory tools and scheduling parallel machine elements, its interaction produces the according convergence technique that will eventually determine other variables as the set of Pareto points that result from multicriteria approaching techniques.
3. The other multicriteria techniques do not depend on game theory implication, so it is just function of the scheduling interaction of elements.
4. The set of Pareto points that result from game theory approaching techniques are a function of scheduling and game theory interaction techniques.
5. The set of Pareto points that result from game theory and scheduling approaching techniques are subset of all points that represent all possible Pareto Front.

6. The set of Pareto points that result from multicriteria approaching techniques are a function of other multicriteria techniques approaches.
7. The unified set composed of these two subsets does not represent the whole set of points possible in a Pareto Front, that is, infinitive points can not be found by neither of these approaches.

6. METHODOLOGY

6.1. METHODOLOGY APPROACH

Parallel identical machine configuration, when preemption is not allowed, sketches the research outlook and gives an integral scene of the systems this research will focus into. This research is not considering due date based jobs, nor deadlines, and is assuming equal release dates for all jobs. It is also supposed that the system is not setup dependent, although this project can be adapted to setup dependent environments in further research. Diverse authors have focused on giving solutions towards these kinds of environments; currently, considerations on this basis have made multicriteria a main focus since such models respond more effectively to the way problems arise in real productive systems. However, all these problems are very complex, and require very specific algorithms and heuristics to evaluate trade-off among criteria, and how the integral performance of the sequence is determined. Game theory has turned to a focus towards these environments' applications. Several papers have been published but still these approaches are mostly new. Literature has focused on other means to tackle these problems.

Through out this research a *descriptive study* will be made, heading for the attempted hypotheses, in order to test them and thus corroborate the fact that different perspectives can be used in order to undertake such problems, giving rise to other means of attaining solutions to provide backup for decision making. Also, discovering the correlation among two fields; set of points resulted from pure multicriteria Scheduling techniques and game theory-scheduling interaction techniques may lead to the fact that the last one is not independent of the classical approach.

By classifying and identifying all elements of this problem, and generating a logical model where solutions can be attained according to the parameters previously stated, a set of points can be obtained as the outcome values of the variables that are the criteria that need to be improved within the resulting schedules, an analysis and synthesis method can be used in order to establish whether the points obtained are truly efficient by comparing the values of these variables with the outcome of variables obtained through other multicriteria models such as the heuristics found in the paper **"**Analytical Evaluation of Multi-Criteria Heuristics**"**[70]**.** In this paper the author states that an algorithm such as the General SB Routine / sumC, where this routine consists of adjusting the bottleneck; and SB stands for Shifting Bottleneck. Another paper key for our multicriteria comparison is "Minimizing Flow Time subject to Optimal Makespan on Two Identical Parallel

[70] DANIELS Richard L, **"**Analytical Evaluation of Multi-Criteria Heuristics**".**

Machines"[71] where they use Lexicographical Search in order to obtain values for the makespan and flow time on identical parallel machines configuration. The values found through this routine will also be compared with the outcomes from our model.

In order to corroborate the robustness of our model we will also design an experiment in which we will run the model under two different setups, and varying two different factors within it: the number of jobs to be processed and the number of machines available. The two different setups will be: Agents using incentives to pursue them to make changes in the actual schedule, and Agents not using incentives on the equations. By this analytical experiment we will try to confirm how solutions under incentives may bring more interesting solutions and achieve the first of our specific objective stated which was to prove that our model indeed generates alternative schedules and as stated on the set of hypotheses, *interesting solutions* for scheduling problems.

Secondary sources taken into account include, books in scheduling theory, game theory, multicriteria scheduling theory and also published papers concerning the applications and interaction between game theory. These papers were basically the driver for our proposed model, including the mechanism design[72], Auction

[71] GUTPA, Jatinder N.D; and HO, Johnny C. "Minimizing Flow Time subject to Optimal Makespan on Two Identical Parallel Machines"
[72] KUTANOGLU, Erhan and WU, David. "An Incentive Compatible Mechanism for Distributed Resource Planning".

Theoretic Modeling[73], Worst case Equilibra[74] and KUTANOGLU's incentive design scheduling that will mainly be obtained from the internet as well as from specialized journals in the data base resources. Another document that was specially important for prescribing our model was Even-Dar's[75] paper where he proposed n jobs with an associated agent, over m machines, and 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 is stated that at least one job is willing to change to another machine, until Nash Equilibrium is reached. In this paper it is also assumed that only one job is allowed to be moved in each step, which is slightly different from which we want to propose within our model. It is also stated that there is a general controller of the entire system who is in charge of allowing or not a movement of a job from one machine to other.

The platform under which this model is based is Visual Basic macros for Excel, where a model is presented according to the assumptions stated, and the steps that needed to be followed were tagged within the algorithm, in order to find an allocation schedule that would allow the testing of the hypotheses. Once a set of schedules is found, the corresponding results need to be filtered to disregard

[73] KUTANOGLU, Erhan and WU, S. David. "An Auction-Theoretic Modeling of Production Scheduling to Achieve Distributed Decision Making".2003.36p
[74] KOUTSOUPIAS Elias, and PAPADIMITRIOU "Worst-case Equilibra". Document in PDF. Berkley University and UCLA. 10p.
[75] EYAL Even-Dar, KEESELMAN Alex, and YISHAY Mansour "Convergence Time to Nash Equilibra". School of Computer Science. Tel Aviv University. March 2006. PDF Document.p17.

dominated points and finally obtain a point or a set of points to conform part of the Pareto Efficient Front obtained from our approach.

6.2. SOLUTION METHODOLOGY

In order to test the project's hypothesis, that is, to find solutions for parallel machine scheduling through the means of Game Theory, a simulation of a scheduling game will be made. From this simulation, several schedules will be attained. Those schedules will have the output for our research, since different systems variables can be analyzed. Since two of these variables are conflictive, makespan and flow time, a scheduling problem considering both of these criteria can be solved by using the mechanism design approach, see ARCHER, Aaron [76], where jobs are considered agents, also see[77], and [1] KUTANOGLU, Erhan and WU, David [78]; on the other hand Nisan also states the usefulness of a mechanism design for selfish agents [79]. Our model contains a different type of interaction and a modification in the payoffs implemented. Each job agent will be playing with a central agent in a sequential order. Each type of agent has an objective that can interact rationally and lead overall system efficiency, in order to meet both criteria through tradeoffs within a payoff matrix.

---

[76] ARCHER, Aaron, TARDOS Eva "Truthful Mechanisms for One-Parameter Agents"
[77] KUTANOGLU, Erhan and WU, David. "An Incentive Compatible Mechanism for Distributed Resource Planning".
[78] RONEN, Amir. "Solving Optimization Problems Among Selfish Agents". Jerusalem: Hebrew University, 2000.
[79] NISAN, Noam. "Algorithms for Selfish Agents: Mechanism Design for Distributed Computation". Jerusalem: Institute of Computer Science. (PDF document) 17p

6.2.1. Elements and Assumptions

The elements considered in the game are as follow:

- **Agent 0 (A_0):** This agent is in charge of the overall system, it is the mechanism controller, its interest is to finish all jobs as fast as possible on the set of machines, that is to minimize the maximum makespan.
- **Job Agents (A_i):** There will be an agent that will look after each job; its main interest will be to seize the corresponding job as soon as possible; there are as many job agents as jobs in the system. Job agents are not willing to wait too long for its job to be processed; they want their job out of the system so they do not want to wait in queue. This is why most job agents will have conflictive objectives with Agent 0, since this last one would rather have jobs with larger processing times allocated first on each machine so that the whole system will finish up in the least amount of time, resulting in the minimum completion time of the last job. These job agents also want their jobs to move ahead in the schedule, so they can be processed first and so, be taken out of the system. *Partial flow time*[*] will give an indicator of how long does each specific job on a particular machine on each time slot will have to wait, so each agent will want the *partial flow time* to be as short as possible.
- **Set of Machines (M_j):** The system will be considered to have n identical parallel machines, j=1, 2…, n. Jobs will be allocated on the machines according

[*] *Flow time on the machine until a corresponding job, that is flow time until a specific time slot on a particular machine.*

to what Agent 0 believes is best for the overall system taking into account what the job agent will do, since both are intelligent and both know that they are acting rationally and will choose their strategy according to what the other thinks the other will do.

- **Strategies for job agents (S_i):** Each job agent i will have a corresponding strategy vector S_i= {A, B}, according to its selfish nature and thus, will have an interest to act upon it, since going for this strategy is what benefits him the most.
- **Strategies for Agent 0 (S_0):** The controlling agent, A_0, will have a corresponding vector S_0= {C, D}, according to the overall system efficiency.
- **Payoff Matrix (P_{2x2}):** Space matrix where decisions on the participating agents are made, according to trade-offs among criteria.
- **Payoffs (a_{ikl}):** Associated cost that an agent i will face if he chooses a specified strategy k, given that the opponent agent chooses another strategy l. Thus, each agent will want to minimize the associated cost in its payoff matrix. A payoff matrix will generate changes in the allocation of jobs in the machines, according to the preferences of the agents and associated costs referring thus, will allow changes and iterations among jobs in the machines. Throughout these iterations, new payoff matrices will be generated and new swaps in jobs will be made until the model can not accomplish better solutions, that is, the termination condition for the iterations within the model.

Assumptions:

In order to simulate a scheduling game on two parallel machines with a central agent or controller, A_0, and a set of job agents, A_i (one agent per job *i*), it is necessary to take a set of assumptions under consideration:

- According to the premises of game theory, agents are assumed to act rationally and, thus, they will choose their strategy according to what the other thinks the other will do.
- Jobs can only move from one machine to another because of the difference in load of the machines, in order to try to balance the system, allowing it to move only from the machine with the largest load to some other machine randomly picked with a lower load.
- Jobs are allowed to move to the same position in the same machine only once.
- If A0 is trying to move a job allocated on a timeslot where the other randomly chosen machine does not have a corresponding job assigned on the same timeslot; that is, the machine with highest load has more jobs assigned than the other, then the swapping will be done with the job selected, along with the empty time slot available on the other machine.
- Job agents only have knowledge of their partial flow time after each possible move and the overall total flow time of the system, they do not know the processing time of other jobs.

- A job that is positioned in the first timeslot is not motivated to participate at all in the game, since it has its lowest possible Partial Flow time, but it can be moved, if other jobs are swapped to its position.

6.2.2. Definition of the Game

Consider a non-cooperative repeated game of two players, where player one represents a job agent and player 2, the controlling agent; agent 0. Each one has two strategies that correspond to their own interests, which are, for the job agent, to seize his job first, which leads to an improvement of his associated partial flow time, and for agent 0, to reach a better makespan. Regarding the system mechanism to reduce not only maximum makespan but also total flow time, it is well known 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; e.g., the flow time of the system is improved when a job with a lower processing time is allocated in a previous position where the prior job before had a longer processing time. In this means, the model must penalize movements according to what represents a system improvement rather than a selfish improvement (what the job agent wants).

Strategies for job agent, S_1= { A,B }:

A= Stay on 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 on the players will be represented as costs, so the involved players will want to minimize the associated costs, in time units. These costs are functions of the makespan and flow time, which are the two conflictive criteria. The assumptions and the objectives for both players enable the design for the payoffs in the corresponding way, each pair of chosen strategies represent an outcome cost (OC). There are four costs for each agent, creating a non-zero sum game matrix.

6.2.2.1 Costs

Costs For each Job agent in the system:

- **ai_{11}** → OC associated when A_i decides not to move to the previous available position and A_0 decides for the job to stay in the current machine, no change is done over the schedule.
- **ai_{12}** → OC associated when A_i decides not to move to the previous available position but A_0 decides to move the job to another machine.
- **ai_{21}** → OC associated when A_i decides to move to the previous available position and A_0 decides for the job to stay in the current machine.

- **ai_{22}** → OC associated when A_i decides to move to the previous available position and A_0 decides for the job to move to another machine.

Costs For Agent 0:

- **$a0_{11}$**→ OC associated when A_0 decides for the job to stay in the current machine and A_i decides not to move to the previous available position.
- **$a0_{12}$** → OC associated when A_0 decides to move the job to another machine associated but A_i decides not to move to the previous available position.
- **$a0_{21}$** → OC associated when A_0 decides for the job to stay in the current machine associated and A_i decides to move the job a previous available position.
- **$a0_{22}$** → OC associated when A_0 decides for the job to move to another machine and A_i decides to move to the previous available position.

These OC's[*] represent different costs for each agent making the game non-zero sum. The costs associated are functions of the maximum makespan, the partial completion time[**], partial flow time[***] and total flow time. So the job agent will be better off, if his associated partial flow time were smaller, since he will have to wait less to be seized. On the other hand, for agent 0, the cost is represented by the associated makespan under given conditions that have to do with his decision of swapping one job from one machine to the other.

---

[*] OC: Outcome Costs

[**] Partial Completion time represents the amount of time until a set of jobs are processed, until the last job that is being taking under account.

[***] A partial flow time itself represents the amount of time the job i has to wait for it to be seized, that is the flow time that represents all jobs before it.

If the job agent has decided for the job to stay on its place, then his outcome depends on what agent 0 decides for it to do, to swap it or not to swap it.

Equations Definitions for the costs within the game matrix

- **Job Agent**

  The associated costs functions for the job agent depend mainly on flow time variables, since this is what major concerns the agent. So there is an incentive for him to move forward in the schedule, but as previously stated there is the need within the model to place a payment over the opportunity loss the system will face if the given job is moved or if it is not.

  In order to achieve all those stated conditions the proposed equation is as follows:

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

As the job agent is trying to minimize this cost, he would want the second term in the equation to be as high as possible in order to decrease the value of the job cost.

Equation Term by Term:

- $Tot\ FT$ represents the total flow time associated from the system, it indicates the maximum value if the rest of the equation tends to zero, that is, if the movement is the least convenient, then this term will represent almost all the cost; the cost associated to a non convenient strategy, which in terms of job agents and flow

time is, to allocate a job that has a greater processing time compared to the one that was formerly on that timeslot.

- F_{job} represents the associated flow time for that machine, that is the machine where was finally allocated.
- β_i in terms of agents it represents the fraction of the processing time of the job on the time slot where agent0 decided to move or leave, with respect to the partial flow time for that job on the analyzed machine and position. Together with F_{job}, they represent the opportunity cost, that is, what the system and the job agent give up in order for the movement associated from the swap among machines, or movement ahead in the schedule into a previous timeslot to take place. Assuming that the agent decides to move ahead to a previous timeslot, then not making the movement represents the opportunity lost, and this is taken into account by considering the processing time of the job that was formerly in the previous slot. If J_{i+1} is picked by the system to play the game of switching allocations then the associated job agent would want to switch to the time slot where J_i is. It is convenient for the system if and only if $P_{i+1} < P_i$.

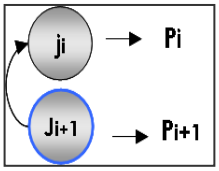

Figure 11. Job agent's decisión

This way it will be an improvement for the overall flow time. If this condition is not true, then the system must penalize the job agent in order to encourage him not to

move from his first position. As these parameters depend on P_{i+1} *and* P_i, then the ratio of the incentive can be defined as the ratio between P_i (processing time of job that was chosen, and the partial flow time associated with a specific machine, the current one where the job is, in case agent 0[*] decides for the job to stay on the same machine, or the second machine, in case agent 0 decides for the job to move to an alternate machine. So this associated flow time on this machine is taken into account in order to obtain β as:

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

This ratio shows a relationship for how much of the flow time is absorbed by the job chosen, up to where it is located, since it is partial. The smaller the processing time, the less it will cost to agent 0 which is the controller. Notice that (1) implies a negative sign; that is, that agents will receive a bonus for their job depending on the partial flow time on the corresponding machine and the processing time. Whenever they gain, agent 0 loses. That way the system balances itself.[**] This way no better completion time can be achieved, then flow time can be compensated, so overall performance can be achieved.

[*] Recall that this game is based on suppositions as how the other player can react, since both players are assumed.

[**] Mechanism designed is based on incentives which imply losses for one side and gains for others in order to maintain balance.

- **Agent 0:**

The associated costs functions for Agent 0 mainly depend on the completion time, since this is of major concern for him. So there is an incentive for him not to allow low processing times first, since his interest is to finish all jobs as soon as possible. In order to achieve all those stated conditions, and as said before, balance both sides of the game, then the incentives given to the job agents have to be paid by the agent controller, and then the proposed equation follows with a positive sign, in other words this is an opportunity cost agent 0 will face to compensate the job agents:

$$(2)\ Agent0\ Cost = C_{\max} + \beta_0\, C_i$$

As Agent 0 is trying to minimize its cost he would want to choose its cost as low as possible.

Equation Term by Term:

- $C_{\max}$ → is the maximum completion time calculated for the machine with the highest load on the set, which agent zero wants to have as low as possible.
- β_0 → for agent 0* it represents the opportunity cost associated to the movement that is about to be done. As agent 0 is concerned about the machines and not one single job on them, this β will consider ratio of partial completion times. More specifically it compares the partial completion time for the job that has been selected to the partial completion time on the second machine, for the same

* β_0 stands for agent 0, while β_i stands for the job agent i.

associated timeslot. If β <1, then the second machine has a higher partial completion time for that same timeslot. For this reason a swap among jobs can be worth the cost. On the other hand if β>1, then a higher cost for agent 0 is implied, since the partial completion time on the second machine is lower than the partial completion time on the current machine analyzed. Of course, this does not guarantee that a swap between jobs may improve total makespan. There may be times when even with β>1, the swap results in a better Schedule. Still the model needs to penalize these costs in a sound way, such as the one chosen.

$$\beta_0 = \frac{C_i(current\ machine)}{C_i(other\ machine)}$$

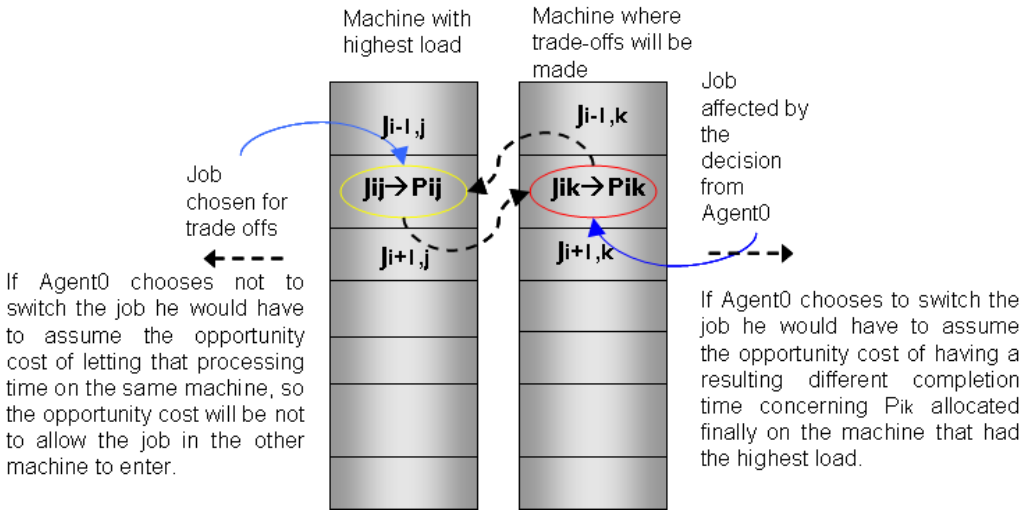

Figure12. Analyzing β for the agent as an opportunity cost of affecting overall flow time

By assuming this cost and adding it to the equation of the associated costs for the job agent, a coherent function will be declared, regarding how much will be taken into account about the other parameter C_i since this β would now mean the percentage that the model will take from this other parameter.

- C_i → this parameter concerns the completion time for the chosen job within the machine that contains the chosen job, depending on the given conditions of whether the job stays on the same position or if it changes to another machine. So if it changes, the completion time accounted will be where the job finally gets to. The two scenerarios can be represented as follows:

1.

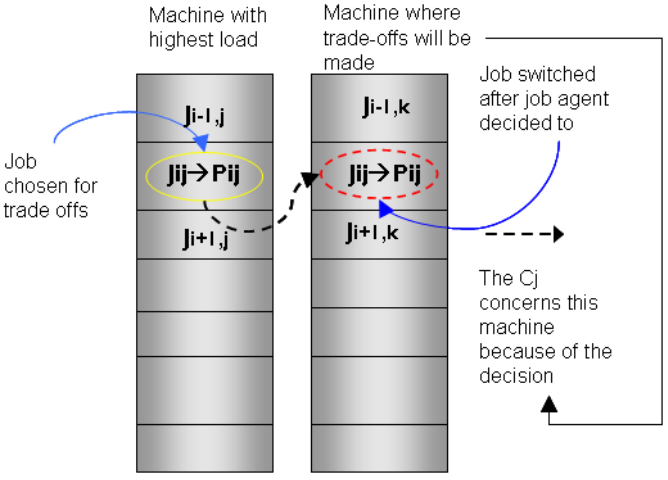

Figure 13. Agent 0 decides to switch the job to the other machine affecting Cj of second machine

2.

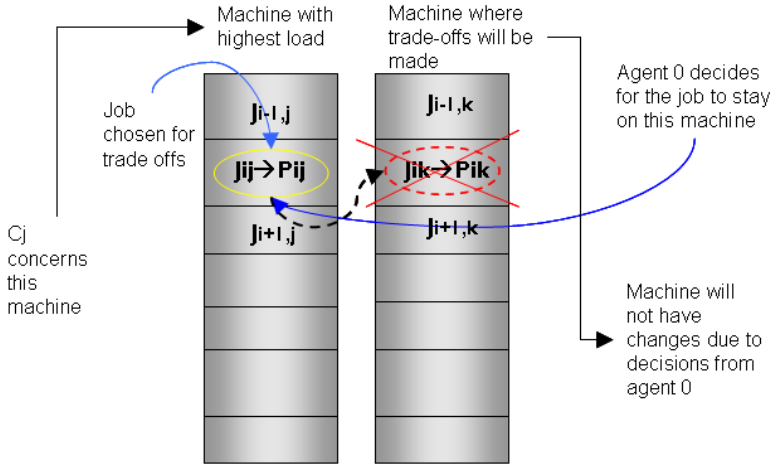

Figure 14. Agent 0 decides not to switch job, from the machine concerned, it will stay that way until another condition is reached.

As the costs concerning the job agent depend on the decisions made by agent 0, and vice versa, an extensive form of the game can be analyzed and this visualizes the whole perspective in order to obtain the stated solutions in terms of the decisions made by the other player (to be thinking rationally about what the other agent is thinking). An extensive perspective can be applied to show how the decisions among agents react for the duration of the game by the following tree:

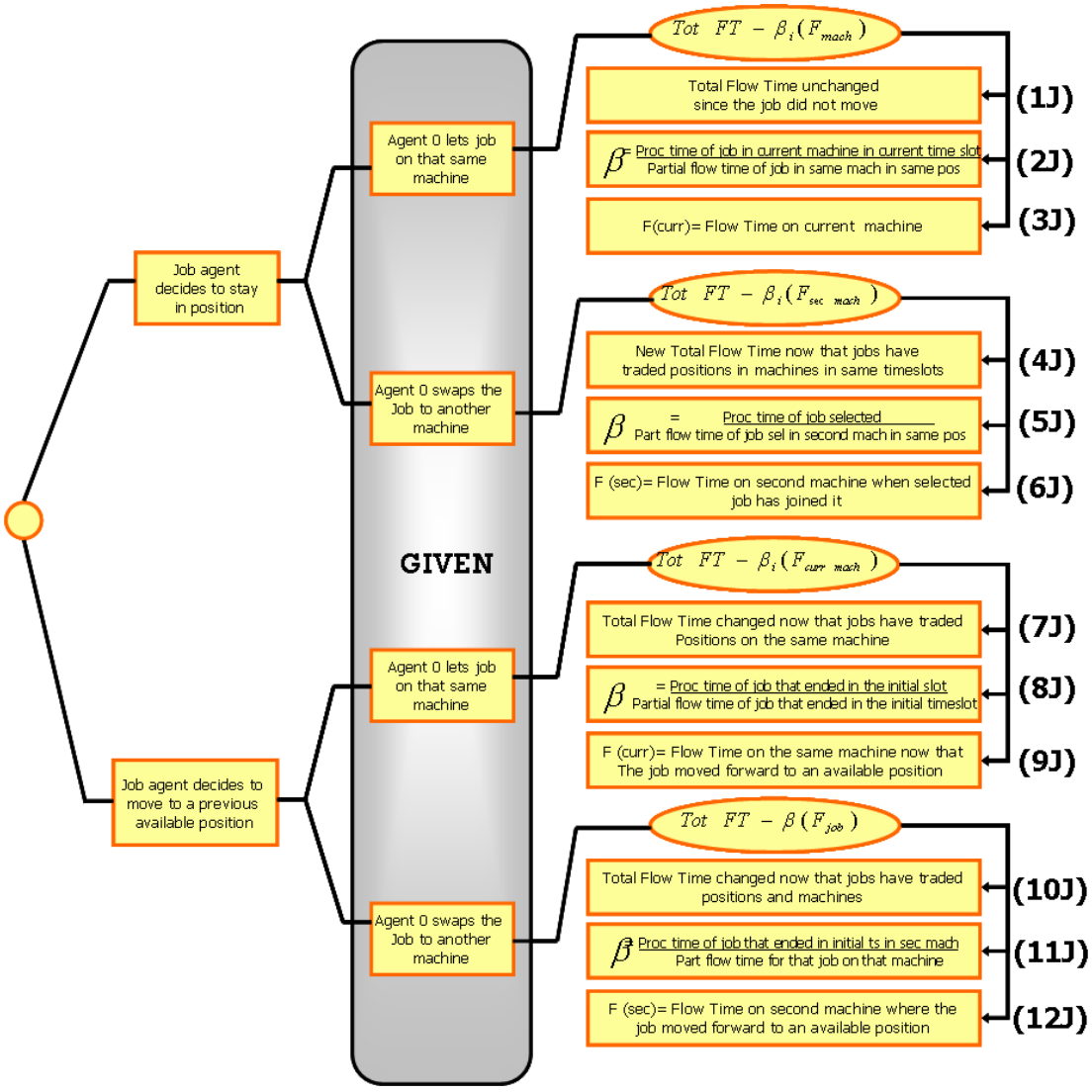

Figure 15. Decision Tree for Job Agent

In Figure 15, the job agent will enter the game, knowing already what agent 0 might do, so the gray part represents given conditions where the job agent has no control.

Split Out Decisions for Job Agent:

1. If Job agent decides, given that Agent 0 has decided not to move that job into another machine, to stay on the current machine and assume a cost of OC= (1)-(2)*(3), according to the numbers given in the tree to identify all elements within the outcome costs, for any decision made.

2. If Job Agent decides to stay in the current position given that Agent 0 has decided to switch jobs the outcome costs would be: OC= (4) - (5)*(6).

3. If Job Agent decides to move to a previous position given that Agent 0 has decided not to move to another machine, then the OC= (7) - (8)*(9).

4. If Job Agent decides to move to a previous position given that Agent 0 has decided to move the job to another machine, then OC = (10) – (11)*(12).

Same analysis can be made for Agent 0 in Figure 16, which shows how agent 0 will enter the game, knowing already what the corresponding job agent might do, so the gray part represents given conditions where agent 0 has no control of.

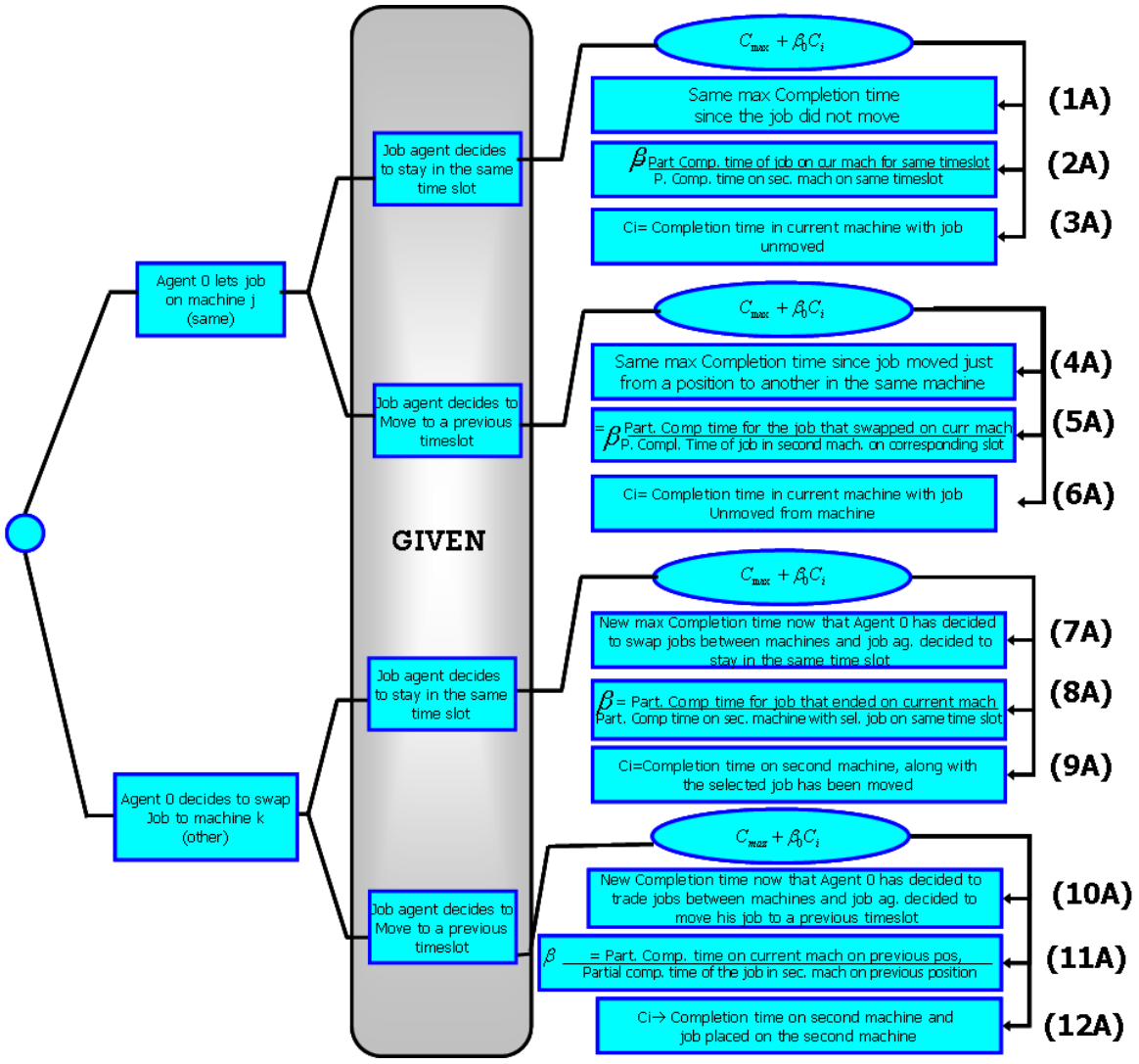

Figure 16. Decision Tree for Agent 0

Split Out Decisions for Agent 0:

1. If agent 0 decides, given that the job agent has decided not to move the job to a previous position, to let the job stay on the current machine, then he would assume

a cost of OC= (1A)-(2A)*(3A), according to the numbers given in the tree to identify all elements within the outcome costs, for any decision made.

2. If Agent 0 decides not to move the job to another machine, given that the job agent has decided to move to a previous position, the outcome costs would be: OC= (4A) - (5A)*(6A).

3. If Agent 0 decides to move the job to another machine given that the Job Agent has decided not to move to a previous position, then the OC= (7A) - (8A)*(9A).

4 If Agent 0 decides to move the job to another machine, given that the Job Agent has decided to move the job to a previous position, then OC = (10A) – (11A)*(12A).

6.2.2.2 Steps the Model Takes

As presented above, from the previous two charts the payoff matrix can be designed and the game can be solved in order to reach an equilibrium; that is, *the decision for which none of the agents will be willing to change, and thus, any change within this decision will lead to worsen conditions for the agents*. This matrix is the core of this project, since this is what leads to the changing mechanisms along the schedules until certain conditions are reached and a set of schedules can be obtained.

Now let us focus on how this game will serve along the scheduling problem, creating iterations among jobs and generating new outcomes. Until this point, it can be said that this mechanism serves as a bi-criteria decision making process, but

has a different approach since it involves rationality among agents. The steps for the game as a whole are organized as follows:

1. As the jobs arrive to the system, they must be assigned to the machines available according to the used load on each machine, which is the sum of the processing times of each job that has arrived to the system, when a job arrives it will be assigned to the machine with the smallest load. This serves as a filter concerning load balancing between the machines in the system.
2. A_0 will want to swap a job in the machine with highest load (this machine is the machine 1 named before on Figure 15 and Figure 16)[*] by randomly picking among all jobs initially allocated in this machine. Let this machine be *current*, the chosen machine from which Agent 0 will pick the jobs he may want to trade with jobs assigned in other machines. At the same time, all jobs within current will prefer to move to a previous position in the time slot to finish its processing earlier.
3. Once a job is picked, agent 0 must select a second machine to switch it with, from the available set of machines (Sec).
4. Once A_0 has determined which job he would rather swap, and which machine swap it into, say job *, he would have to decide between swapping the job or not, and so would the job * agent. They would both do this according to their payoff matrices, the job agent would take into account what Agent 0 would do,

[*] Machine 1 (current) is the machine wih the highest load on the set of machines, and Machine 2 (Sec), is another machine within the group randomly picked by Agent 0 in order to start the swapping among jobs.

and vice versa. So, this is the point where the two agents meet, to solve their differences, and find equilibrium among their strategies. This is where the dynamic of the game relies. Once the equilibrium is reached, each agent will make a decision and the current schedule will change according to both agents' choice.

5. The reallocation will be implemented and the job that was iterated will get to a new position, or not, depending on the outcomes on the payoff matrix. Iterations will later be completed when none of the jobs in the system have incentives to move to a different position. In this sense, it can be said that the procedure takes a social concern since it involves all associated stakeholders, in the decision making process.
6. The final schedule is the outcome of all these confrontations, and since the game provides so many iterations, results from many schedules can be analyzed, and thus, compared. This will be done by analyzing the outcomes: makespan and flow time through a *Pareto filter*, which is an important tool within our model for the outcomes of the game. Through this, we will analyze the Pareto Front which are the non dominated solutions, and thus, the points that will stand for, *our Pareto Generated Points*[80], as stated in the problem's hypothesis. In the long run, it is possible that a solution within the payoff matrix converges, as what happens when repeated games are played and in the long run they reach equilibrium. The importance of these points lies on the fact that

[80] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.54

they allow making comparisons between the obtained schedules, and other schedules generated through pure multicriteria approaches found in literature. These results were tested against some known heuristics well known such as Lexicographical Search Base[81], the results can be tested and can tell whether the mechanism provides robust solutions, and thus compliments the set of points of the total Pareto Front found through other means.

6.2.3. Numerical Example

Consider a bi-criteria scheduling problem that consists of 10 jobs, J_i={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 (ΣC_j). Neither release dates nor setups are considered and preemption is not allowed.

This problem is to be solved by using the game theoretic approach defined in the previous section. The job selected, and its associated agent will be competing against the controlling agent in a non-cooperative game of two players for every iteration; where the players are considered rational and the strategies reflect their own preferences and the payoffs, their own incentives.

[81] T'KINDT, Vincent and BILLAUT, Jean-Charles. Op. Cit. p.79

Table 55 shows the solutions for heuristics in multicriteria scheduling. These solutions were compared to the ones obtained from the Scheduling Game proposed in this investigation. Figure 19 shows the graph with the results from the other solutions to compare them to the Pareto Front.

Solution procedure:

- As the jobs arrived, in a random way, they were assigned to the machine with the lowest load. The initial position of the game is given by the allocation of the jobs as seen in Table 3.

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. Initial Allocation of jobs in 2 machines by Load Balancing

- In order to choose the job that will participate first, it is necessary to select the machine from which to select the jobs. This machine is the one that gives the maximum makespan and so the jobs in this machine will tend to move to the other machine in order to balance the load. In this case, it is found that *machine 2* has the biggest makespan with a difference of 8 compared to the other machine. Once chosen the machine, a job is selected from that machine randomly. The job agent that represents this job will be the first player, and for this case, the first one to play is job *agent 5*.

- The payoffs for both players are related one with the other, even though each one seeks different and conflicting objectives. The job agent that represents job 5, A_5 has two options, either stay in its actual position (timeslot 2), or move to the previous position (timeslot 1). On the other hand, the controlling agent, A_0, has the choice to move the job to machine 1 or leave it in machine 2.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
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 4. Job agent selected for the game (A_5) and the possible movements it can make

If A_5 chooses to stay in the current position and A_0 decides to leave it in the current machine, it will cost A_5 649.04 (see equation 1 in figure 17), while A_0 will respond to a cost of 252.42 (see equation 2). Yet, if A_5 decides to move to the previous position and A_0 prefers to leave the job in the same machine it will cost the job agent 332.52 (see equation 3) and the controlling agent with a cost of 163.50 (see equation 4). On the other hand, if A_5 decides to stay in the same position but is moved by the controlling agent to the alternate machine, the costs associated to this movement will be 629.40 (see equation 5), while A_0 will have to incur with a cost of 610 (see equation 6). Lastly, if the job agent decides to move to a previous position while agent 0 moves the job to the alternate machine, the cost that A_5 has to assume is (see equation 7) while A_0 incurs in a cost of 1228.50 (see equation 8).

Recall that the strategies are, A: to stay in position, B: to move to previous position, C: leave job on current machine and D: move job to the other machine.

(1)	$Tot\ FT = 665 \quad \beta_i = 0.04167 \quad F_{sec\ mach} = 96$ $Tot\,FT - \beta_i(F_{secmach}) = 665 - 0.04167*383 = 649.04$
(2)	$C_{\max} = 109 \quad \beta_0 = 50/38 = 1.315 \quad C_i = 109$ $C_{\max} + \beta_0(C_i) = 109 + 1.315*109 = 25242$
(3)	$Tot\ FT = 623 \quad \beta_i = 46/54 = 0.852 \quad F_{currmach} = 341$ $Tot\,FT - \beta_i(F_{secmach}) = 623 - 0.852*341 = 33262$
(4)	$C_{\max} = 109 \quad \beta_0 = 4/8 = 0.5 \quad C_i = 109$ $C_{\max} + \beta_0(C_i) = 109 + 0.5*109 = 163.5$
(5)	$Tot\ FT = 665 \quad \beta_i = 4/20 = 0.2 \quad F_{currmach} = 178$ $Tot\,FT - \beta_i(F_{secmach}) = 665 - 0.2*178 = 629.4$
(6)	$C_{\max} = 135 \quad \beta_0 = 76/12 = 6.333 \quad C_i = 75$ $C_{\max} + \beta_0(C_i) = 135 + 6.33*75 = 610$
(7)	$Tot\ FT = 661 \quad \beta_i = 30/38 = 0.789 \quad F_{currmach} = 262$ $Tot\,FT - \beta_i(F_{secmach}) = 661 - 0.789*262 = 454.16$
(8)	$C_{\max} = 113 \quad \beta_0 = 46/4 = 11.5 \quad C_i = 97$ $C_{\max} + \beta_0(C_i) = 113 + 11.5*97 = 1228.5$

- Table 5 shows the resulting payoff matrix for job agent 5's movements. When solving the matrix, each player will try to minimize their corresponding cost. By Nash equilibrium, it is observed that job 5 is preferred in the current machine but in a previous position. Given this first move, the allocation of

jobs in the machines changes, proposing the first changed schedule and the makespan, C_{max}, is now 109 and the total flow time, ΣC_i is now 623 units of time. See table 6 with the updated schedule.

		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

Table 5. Payoff Matrix for job Agent 5 in the first iteration.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	2	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

Table 6. Proposed schedule for first iteration.

- In the second iteration, the machine selected is the second one, since it is the one with the makespan, C_{max}. From this machine, job 9 is selected randomly. The resulting matrix has a dominant strategy equilibrium solution where it is left in machine 2 but moved to a previous position. Table 7 shows the payoff matrix for this job agent and table 8 shows the updated schedule for this iteration.

		AGENT 0			
	ROUND 2	C		D	
JOB AGENT 9	A	614,18	237,56	613,89	270,06
	B	537,57	216,09	577,08	294,00

Table 7. Payoff Matrix for job Agent 9 for iteration 2.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	2	46	50	54
4	19	57	103	9	6	56	110
7	21	78	181	6	36	92	202
8	23	101	282	10	17	109	311

Table 8. Proposed schedule for iteration 2.

- The resulting schedule shows a value of 109 for C_{max} and 593 for total flow time.
- The game matrix and updated schedule for the next iterations are shown in the next tables. For the fourth iteration, the resulting matrix has a dominant strategy equilibrium solution and the resulting schedule shows a value of 109 for C_{max} and 553 for Total flow time.

		AGENT 0			
	ROUND 3	C		D	
JOB AGENT 9	A	576,04	216,09	576,80	260,00
	B	374,91	137,68	504,75	408,00

Table 9. Payoff Matrix for job Agent 9 for iteration 3.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	9	6	10	14
4	19	57	103	2	46	56	70
7	21	78	181	6	36	92	162
8	23	101	282	10	17	109	271

Table 10. Proposed schedule for iteration 3.

- In the fourth iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 109 for C_{max} and 534 for total flow time.

		AGENT 0			
	ROUND 4	C		D	
JOB AGENT	A	536,00	226,63	536,00	230,00
10	B	498,00	211,01	526,00	233,59

Table 11. Payoff Matrix for job Agent 10 for iteration 4.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	9	6	10	14
4	19	57	103	2	46	56	70
7	21	78	181	10	17	73	143
8	23	101	282	6	36	109	252

Table 12. Proposed schedule for iteration 4.

- In the fifth iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 128 for C_{max} and 534 for total flow time.

		AGENT 0			
	ROUND 5	C		D	
JOB AGENT	A	368,40	216,09	405,55	172,19
2	B	558,07	252,42	501,30	138,67

Table 13. Payoff Matrix for job Agent 2 for iteration 5.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	9	6	10	14
2	46	84	130	4	19	29	43
7	21	105	235	10	17	46	89
8	23	128	363	6	36	82	171

Table 14. Proposed schedule for iteration 5.

- In the sixth iteration, the resulting matrix follows a Nash equilibrium strategy solution and the resulting schedule shows a value of 124 for C_{max} and 534 for total flow time.

		AGENT 0			
	ROUND 6	C		D	
JOB AGENT	A	501,56	420,17	493,58	297,72
7	B	434,96	388,41	503,65	353,61

Table 15. Payoff Matrix for job Agent 7 for iteration 6.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	9	6	10	14
2	46	84	130	4	19	29	43
10	17	101	231	7	21	50	93
8	23	124	355	6	36	86	179

Table 16. Proposed schedule for iteration 6.

- In the seventh iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 110 for C_{max} and 534 for total flow time.

		AGENT 0			
	ROUND 7	C		D	
JOB AGENT	A	302,48	595,20	316,89	155,29
3	B	511,65	1.054,00	531,91	141,87

Table 17. Payoff Matrix for job Agent 3 for iteration 7.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
9	6	14	22	3	30	34	38
2	46	60	82	4	19	53	91
10	17	77	159	7	21	74	165
8	23	100	259	6	36	110	275

Table 18. Proposed schedule for iteration 7.

- In the eighth iteration, the resulting matrix follows a Nash equilibrium strategy solution and the resulting schedule shows a value of 113 for C_{max} and 547 for total flow time.

		AGENT 0			
	ROUND 8	C		D	
JOB AGENT	A	476,58	207,17	472,51	313,97
4	B	424,00	290,71	414,54	255,30

Table 19. Payoff Matrix for job Agent 4 for iteration 8.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
4	19	27	35	3	30	34	38
2	46	73	108	9	6	40	78
10	17	90	198	7	21	61	139
8	23	113	311	6	36	97	236

Table 20. Proposed schedule for iteration 8.

- In the ninth iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 137 for C_{max} and 547 for total flow time.

		AGENT 0			
	ROUND 9	C		D	
JOB AGENT	A	414,54	319,23	408,22	193,51
2	B	526,43	292,47	546,64	174,02

Table 21. Payoff Matrix for job Agent 2 for iteration 9.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
4	19	27	35	3	30	34	38
9	6	33	68	2	46	80	118
10	17	50	118	7	21	101	219
8	23	73	191	6	36	137	356

Table 22. Proposed schedule for iteration 9.

- In the tenth iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 124 for C_{max} and 547 for total flow time.

		AGENT 0			
	ROUND 10	C		D	
JOB AGENT	A	511,00	394,11	511,00	248,00
6	B	541,00	454,84	543,00	252,67

Table 23. Payoff Matrix for job Agent 6 for iteration 10.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
4	19	27	35	3	30	34	38
9	6	33	68	2	46	80	118
10	17	50	118	7	21	101	219
6	36	86	204	8	23	124	343

Table 24. Proposed schedule for iteration 10.

- In the eleventh iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 113 for C_{max} and 547 for total flow time.

		AGENT 0			
	ROUND 11	C		D	
JOB AGENT	A	276,21	280,15	385,26	171,71
3	B	549,94	589,00	493,48	122,40

Table 25. Payoff Matrix for job Agent 3 for iteration 11.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
9	6	44	90	2	46	69	96
10	17	61	151	7	21	90	186
6	36	97	248	8	23	113	299

Table 26. Proposed schedule for iteration 11.

- In the 12th iteration, the resulting matrix follows a mixed strategy solution and the resulting schedule shows a value of 113 for C_{max} and 547 for total flow time.

		AGENT 0			
	ROUND 12	C		D	
JOB AGENT	A	524,00	244,64	524,00	252,00
8	B	528,00	283,43	517,00	245,36

Table 27. Payoff Matrix for job Agent 8 for iteration 12.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
9	6	44	90	2	46	69	96
10	17	61	151	7	21	90	186
6	36	97	248	8	23	113	299

Table 28. Proposed schedule for iteration 12.

- In the 13th iteration, the resulting matrix follows a mixed strategy solution and the resulting schedule shows a value of 113 for C_{max} and 522 for total flow time.

		AGENT 0			
	ROUND 13	C		D	
JOB AGENT	A	513,24	279,72	512,32	242,63
7	B	443,71	226,00	534,48	242,98

Table 29. Payoff Matrix for job Agent 7 for iteration 13

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
9	6	44	90	7	21	44	71
10	17	61	151	2	46	90	161
6	36	97	248	8	23	113	274

Table 30. Proposed schedule for iteration 13.

- In the 14th iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 126 for C_{max} and 522 for total flow time.

		AGENT 0			
	ROUND 14	C		D	
JOB AGENT 2	A	443,71	279,72	443,80	211,40
	B	513,24	290,20	534,92	208,76

Table 31. Payoff Matrix for job Agent 2 for iteration 14

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
9	6	44	90	7	21	44	71
2	46	90	180	10	17	61	132
6	36	126	306	8	23	84	216

Table 32. Proposed schedule for iteration 14.

- In the 15th iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 113 for C_{max} and 522 for total flow time.

		AGENT 0			
	ROUND 15	C		D	
JOB AGENT 2	A	443,80	311,90	443,71	189,59
	B	552,56	366,55	519,82	178,51

Table 33. Payoff Matrix for job Agent 2 for iteration 15.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
9	6	44	90	7	21	44	71
10	17	61	151	2	46	90	161
6	36	97	248	8	23	113	274

Table 34. Proposed schedule for iteration 15.

- In the 16th iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 112 for C_{max} and 522 for total flow time.

		AGENT 0			
	ROUND 16	C		D	
JOB AGENT	A	440,96	226,00	463,40	167,05
7	B	452,16	187,34	495,33	191,79

Table 35. Payoff Matrix for job Agent 7 for iteration 16.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
7	21	59	105	9	6	29	56
10	17	76	181	2	46	75	131
6	36	112	293	8	23	98	229

Table 36. Proposed schedule for iteration 16.

- In the 17th iteration, the resulting matrix follows a Nash equilibrium strategy solution and the resulting schedule shows a value of 109 for C_{max} and 533 for total flow time.

		AGENT 0			
	ROUND 17	C		D	
JOB AGENT	A	494,48	225,49	493,50	298,50
10	B	483,71	324,41	454,23	269,77

Table 37. Payoff Matrix for job Agent 10 for iteration 17.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
7	21	59	105	10	17	40	67
9	6	65	170	2	46	86	153
6	36	101	271	8	23	109	262

Table 38. Proposed schedule for iteration 17.

- In the 18th iteration, the resulting matrix follows a Nash equilibrium strategy solution and the resulting schedule shows a value of 109 for C_{max} and 510 for total flow time.

		AGENT 0			
	ROUND 18	C		D	
JOB AGENT	A	510,00	226,63	510,00	244,00
8	B	464,00	214,65	514,00	241,76

Table 39. Payoff Matrix for job Agent 8 for iteration 18.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	4	19	23	27
7	21	59	105	10	17	40	67
9	6	65	170	8	23	63	130
6	36	101	271	2	46	109	239

Table 40. Proposed schedule for iteration 18.

- In the 19th iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 109 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 19	C		D	
JOB AGENT	A	449,36	182,90	466,41	190,60
10	B	438,72	169,24	438,78	202,96

Table 41. Payoff Matrix for job Agent 10 for iteration 19.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	10	17	21	25
7	21	59	105	4	19	40	65
9	6	65	170	8	23	63	128
6	36	101	271	2	46	109	237

Table 42. Proposed schedule for iteration 19.

- In the 20th iteration, the resulting matrix follows a Nash equilibrium strategy solution and the resulting schedule shows a value of 109 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 20	C		D	
JOB AGENT	A	438,72	182,90	459,12	183,95
4	B	449,36	174,97	439,57	190,00

Table 43. Payoff Matrix for job Agent 4 for iteration 20.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	10	17	21	25
7	21	59	105	4	19	40	65
9	6	65	170	8	23	63	128
6	36	101	271	2	46	109	237

Table 44. Proposed schedule for iteration 20.

- In the 21st iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 111 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 21	C		D	
JOB AGENT	A	462,00	226,63	462,00	210,00
2	B	508,00	253,22	512,00	225,60

Table 45. Payoff Matrix for job Agent 2 for iteration 21.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	10	17	21	25
7	21	59	105	4	19	40	65
9	6	65	170	8	23	63	128
2	46	111	281	6	36	99	227

Table 46. Proposed schedule for iteration 21.

- In the 22nd iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 109 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 22	C		D	
JOB AGENT	A	462,00	235,45	462,00	210,00
2	B	542,00	296,00	495,00	214,21

Table 47. Payoff Matrix for job Agent 2 for iteration 22.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
3	30	38	46	10	17	21	25
7	21	59	105	4	19	40	65
9	6	65	170	8	23	63	128
6	36	101	271	2	46	109	237

Table 48. Proposed schedule for iteration 22.

- In the 23rd iteration, the resulting matrix follows a mixed strategy solution and the resulting schedule shows a value of 122 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 23	C		D	
JOB AGENT	A	346,84	169,24	395,18	241,68
10	B	494,68	340,63	368,88	135,88

Table 49. Payoff Matrix for job Agent 10 for iteration 23.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
10	17	25	33	3	30	34	38
7	21	46	79	4	19	53	91
9	6	52	131	8	23	76	167
6	36	88	219	2	46	122	289

Table 50. Proposed schedule for iteration 23.

- In the 24th iteration, the resulting matrix follows a dominant strategy solution and the resulting schedule shows a value of 105 for C_{max} and 508 for total flow time.

		AGENT 0			
	ROUND 24	C		D	
JOB AGENT	A	468,20	300,31	468,68	194,78
8	B	479,44	273,17	500,00	219,38

Table 51. Payoff Matrix for job Agent 8 for iteration 24.

MACHINE 1				MACHINE 2			
Ji	Pi	Ci	PartialFi	Ji	Pi	Ci	PartialFi
1	8	8	8	5	4	4	4
10	17	25	33	3	30	34	38
7	21	46	79	4	19	53	91
8	23	69	148	9	6	59	150
6	36	105	253	2	46	105	255

Table 52. Proposed schedule for iteration 24.

After twenty four iterations, the jobs in the current machine were no longer motivated to move to other positions that had not been occupied already, so the system reached equilibrium for this game. In this case, this equilibrium corresponded to a dominated strategy where all the job agents selected from machine 2 preferred to stay on their current positions and both machines were balanced, having each 105 in C_{max}. The results to all these schedules can be observed in a Pareto Chart, where two of the 24 schedules correspond to Pareto Optimal Solutions and form the Pareto Front. From these solutions, only one is considered the “better” solution since one solution is *stricter* than the other. This

corresponds to the schedule for the last iteration, for a value of value of 105 for C_{max} and 508 for total flow time. See figure 18 with the chart that shows the Pareto front for the initial results.

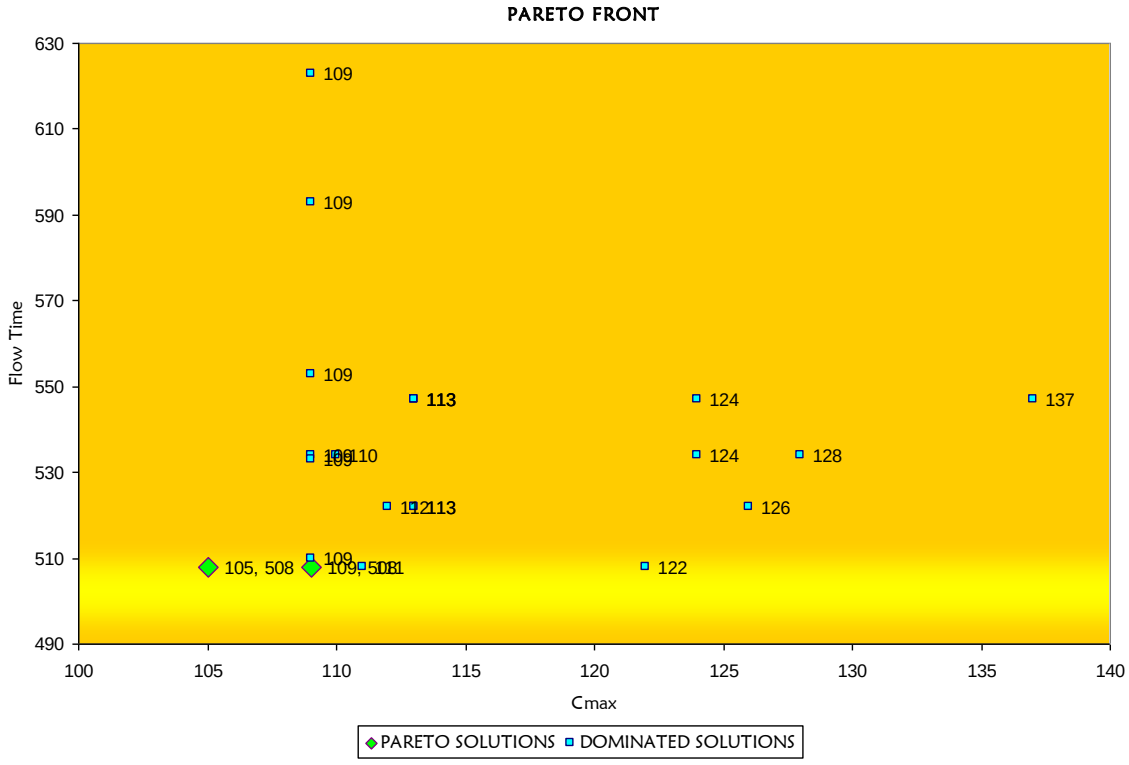

Figure 17. Pareto Front for initial results of the example.

6.2.4. Results Obtained

In order to obtain results from the scheduling problem presented above and for which its procedure has been explained, a computerized application has been designed in Microsoft Excel programmed under Visual Basic, known as “The Scheduling Game”. This application asks the user to enter the number of jobs, their

processing times and the number of parallel machines available. It is a rather dynamic game, for each iteration; a job is moved according to the selected strategies that have been previously defined. The number of iterations can be entered by the user, if given the option, and to avoid unnecessary repetition of movements, every time a solution is found, it is saved so that a specific job agent does not move to the same machine in the same position. In case the number of iterations entered by the user exceeds the number of iterations needed by the problem in specific, a termination condition was designed for the game, given by the tendency for the players to choose a specific strategy that can be dominated in the long run. For each problem, given the value for number of jobs and machines, the number of iterations can not exceed a value of *jobs *jobs* machines*, which also marks a termination for the game.

This same example was tested once again, but this time the program runs until equilibrium is reached. After twenty two iterations, mixed equilibrium was reached where the percentages corresponding to the strategies AC, AD, BC and BD were 0, 76.3, 23.7 and 0.3, respectively. This mixed equilibrium achieved is due to the fact that the agents have not chosen a definite strategy in the long run, but are more likely to choose the strategy AD, which means that the job agents prefer to stay on their current positions while the controlling agent prefers to move them to the other machine. The solutions obtained are graphed in figure 18 where the Pareto Front

has been constructed and the schedules for all the points in the graph are specified in a report generated by the program. See Appendix A.

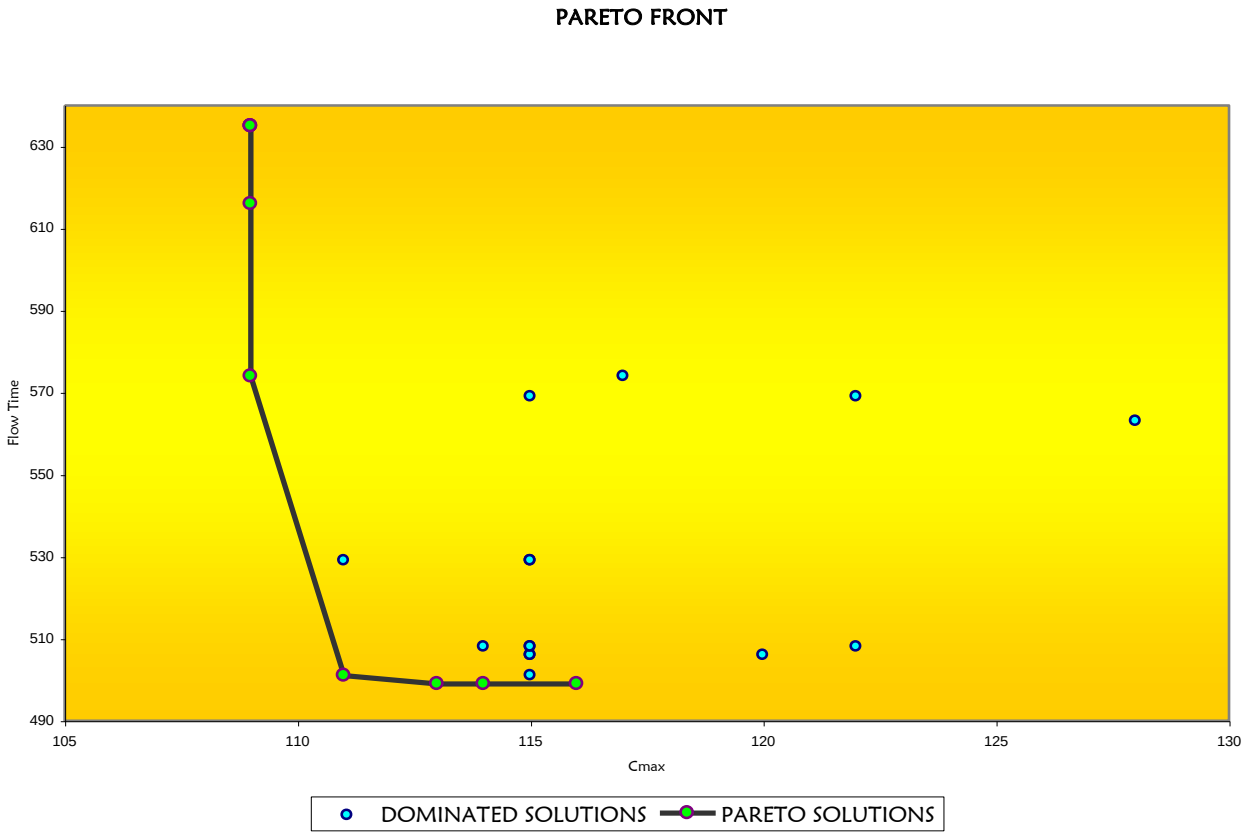

Figure 18. Pareto Front obtained from the Scheduling Game for this example.

For this example, the Pareto Front shown represents the solutions that are considered non-dominated by both types of agents in the Scheduling Game. They have been obtained from *better* allocation of jobs and the interaction of them on both parallel machines. It is important to recall that jobs are to be moved once they have been allocated as soon as they enter the system, giving no importance to the

processing time each job has. During the dynamic interaction of the game, these processing times are reflected in the payoffs but the initial allocation takes no consideration of them. The solutions shown are clearly conservative, due to the nature of game theory, where solutions might avoid getting a better one; that is evidently selfish. Also, these values are a result of some randomness that takes place during the game. See table 53 for the Pareto Optimal Solutions graphed in figure 18.

PARETO SOLUTIONS	
Cmax	Flow Time
109	635
109	616
109	574
111	501
113	499
114	499
116	499

Table 53. Values of Pareto Solutions of the Scheduling Game for this example.

The values that have constructed the Pareto Front are efficient schedules that have been generated throughout the game, for which the reported schedules have been saved in a report from Microsoft Excel to a text file. In the Appendix, the specific schedules for the results reported are shown. In table 54, the schedules are specified with their values, Z.

SCHEDULE	MACHINE 1	MACHINE 2	Z(Cmax,ΣCj)
1	{ J1,J3, J4, J7, J8 }	{ J2, J5, J9, J6, J10 }	(109 , 635)
2	{ J1, J3, J4, J7, J8 }	{ J2, J5, J9, J10, J6 }	(109 , 616)
3	{ J1, J3, J4, J7, J8 }	{ J5, J2, J9, J10, J6 }	(109 , 574)
4	{ J1, J7, J8, J3, J10 }	{ J9, J5, J4, J2, J6 }	(111 , 501)
5	{ J1, J4, J8, J3, J10 }	{ J9, J5, J7, J2, J6 }	(113 , 499)
6	{ J1, J4, J7, J3, J6 }	{ J9, J5, J8, J2, J10 }	(114 , 499)
7	{ J1, J4, J8, J3, J6 }	{ J9, J5, J7, J2, J10 }	(116 , 499)

Table 54. Schedules generated that belong to the weak Pareto Front solution set.

From all of these solution sets, it is important to understand that these are weak Pareto solutions, because there is no strict inequality between them. From this solution set, a strict Pareto solution set could be obtained and for this example only 3 schedules give these solutions, schedules 3, 4 and 5.

6.2.5. Comparing Results to other Heuristics.

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" but with the difference that they applied a hierarchical approach, but in two steps, each step takes an optimal criterion and from this, obtains the minimum value of the second criterion. In this case, it first obtains a minimum flow time, subject to an optimal makespan, α^*, solving the multicriteria problem expressed as P2 || $F_h(\Sigma C_i/C_{max})$ and after this, they obtain a minimum makespan subject to a total flow time β^*, solving the multicriteria problem expressed as P2 || $F_h(C_{max}/\Sigma C_i)$ problem. This procedure used is known as the lexicographic search base algorithm[82].

Also, other results were obtained from the LEKIN SCHEDULING SYSTEM Software, using the rules already known in scheduling like SPT, LPT, FCFS (first

[82] GUPTA, Hatinder N.D. and HO, Johnny. "Minimizing Flow Time subject to Optimal Makespan on Two Identical Parallel Machines."

come first served) and a heuristic used by this software to solve multicriteria scheduling problem, known as the Shifting Bottleneck Heursitic.

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 55. Results from other algorithms/heuristics

Table 55 shows the solutions for heuristics in multicriteria scheduling. These solutions were compared to the ones obtained from the Scheduling Game proposed in this investigation. Figure 20 shows the graph with the results from the other solutions to compare them to the Pareto Front.

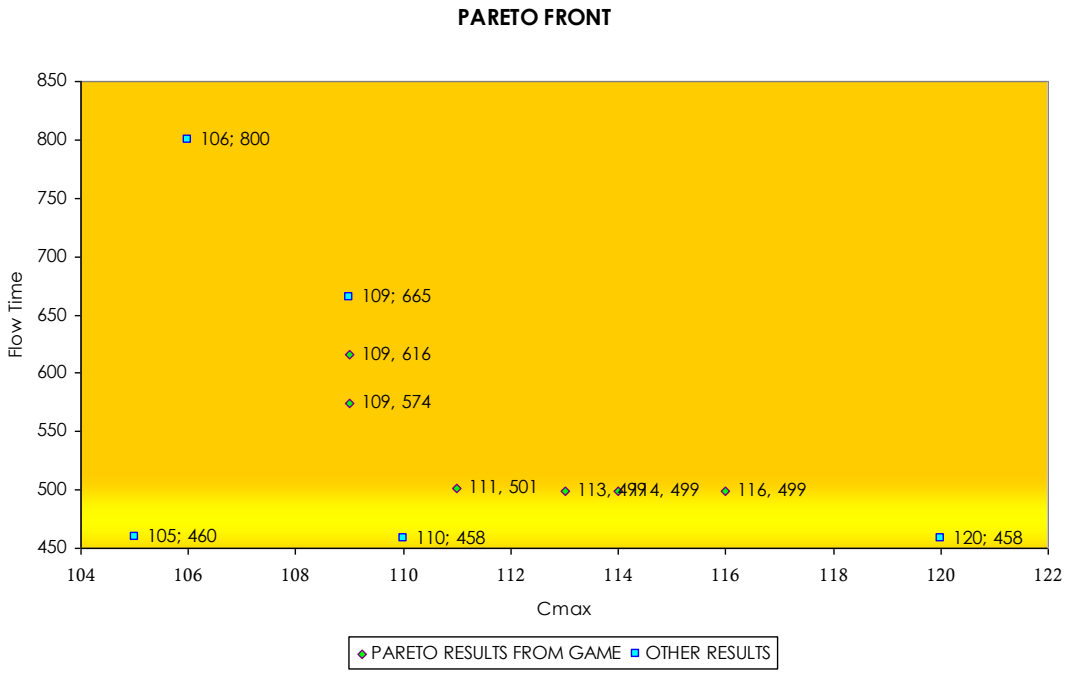

Figure 19. Pareto Front contrasted with results from other MCD* tools

Given the set of solutions for the Scheduling Game as P^{SG} and the set of solutions found by multicriteria scheduling techniques as P^{MC}, variables already defined in the previous chapters, an interpretation can be established according to the hypotheses formulated in this investigation with respect to the Pareto points graphed in figure 19, where the values seem to be close enough the ones found by classical multicriteria techniques. One of the approaches used, the Lexicographical Search Base algorithm shows the best value found so far and the Strict Pareto Points found in the "Scheduling Game" are compared to this value, (for schedules 3, 4 and 5, 24.78%, 8.91% and 8.48% difference from the Flow Time, respectively; and 3.8%, 5.71% and 7.6% difference from the Cmax[*]) which indicates 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.

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

P^{SG}= { (109,635) ; (109, 616) ; **(109, 574)** ; **(111, 501)** ; **(113, 499)** ; (114, 499) ; (116, 499) }

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

---

[*] $\%\ difference = \frac{P^{SG} - P^{MC}}{P^{MC}}$

6.2.6. An extension to the results.

For the previous example, it could be observed that equilibrium was reached where the jobs prefer to stay in their positions during their last move. It has been observed in literature*, that for two up to three machines, Nash equilibrium can be reached for a similar heuristic known as the load balancing mechanism. An extension to these results was made for different setups of problems in order to test the game theoretic influence in the results, where in the long run, the players tend to choose or a not to choose a strategy in particular.

As a further analysis to the game theoretic interpretation of the mechanism proposed during this investigation, 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 β calculated during the game. Given the case that the game was played without taking in consideration the incentives, different types of results would be reached. In order to give interpretations of these situations, a comparison was made for two different setups of the game, one with the payoffs used in the proposed model (a game with incentives) and the other one with the payoffs being only the portion of the criteria each player wishes to reach, for example, for agent 0, the value for Cmax and for each job agent, the resulting value for Total Flow Time (a game without incentives), with the purpose of

* See Literatura Review, Section 2.5

observing how selfish agents can affect the type of solution achieved during the game and the results obtained in the Pareto Front.

This experiment contemplated two values in particular, makespan and total flow time, which will be compared according to the range of solutions achieved for each instance of the experiment. The instances used in the computational experiments were randomly generated and the ranges used for n and m were [10, 30] and [2, 4], respectively. The processing times were generated following a discrete uniform distribution DU(1,50) and 4 replications were considered for each configuration of number of jobs and machines according to the two factors and two levels observed for the experiment (2^2=4 replications).

The results for this simulation have been summarized in the next tables* and graphs, where it can be observed that the behavior of the agents when playing with incentives have shown to be more robust solutions than the ones considering no incentives. In other words, when the agents decided to follow their own criteria (A_0, C_{max} and A_i , Total Flow time), the results where not as *efficient* as in the proposed model.

* In the tables only the Strict Pareto Solutions were taken in consideration for each replication. See graph for a better comparison of the results.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***
1	Mixed Equilibrium (0.389 , 0.397, 0, 0.227)	(828.8857, 266.8307)	174 193	907 896	Nash Equilibrium "AC"	(899.0625, 174.2344)	174	893
2	Mixed Equilibrium (0.401 , 0.195, 0, 0.408)	(853.6832, 231.7396)	174	893	Nash Equilibrium "AC"	(900.44, 174)	174	893
3	Dominated Strategy Equilibrium "A"	(792.2596, 327.9392)	174	889	Nash Equilibrium "AC"	(899.6301, 174.5753)	174	893
4	Mixed Equilibrium (0.235 , 0.476, 0.289, 0.008)	(785.6548, 365.2928)	174	897	Nash Equilibrium "AC"	(934.8261, 175.7681)	175	930

Table 56. Comparison of the results for each replication in the instance m=2, n=10

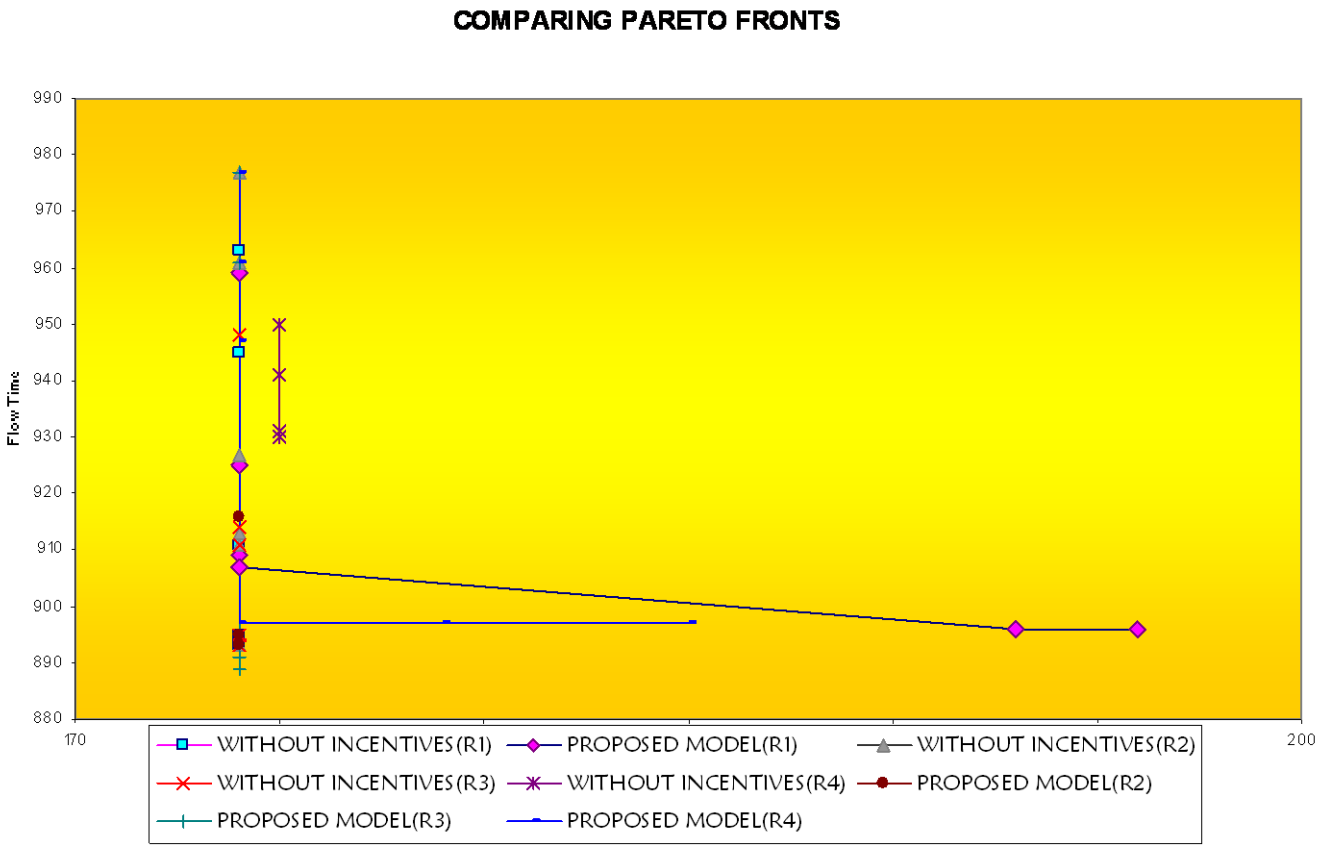

Figure 20. Comparison of the Pareto Fronts generated in the instance m=2, n=10.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Mixed Equilibrium (0.11 , 0.554, 0.335, 0.027)	(2314.149, 497.4392)	276	2499	Dominated Strategy Equilibrium "C"	(2677.227, 276.0833)	276	2622
2	Dominated Strategy Equilibrium "D"	(2359.723, 550.1657)	276 277	2368 2336	Nash Equilibrium "AC"	(2810.455, 278.356)	278	2756
3	Dominated Strategy Equilibrium "D"	(2402.924, 549.538)	276 278	2659 2536	Dominated Strategy Equilibrium "C"	(2702.425, 276.4575)	276	2623
4	Mixed Equilibrium (0 , 0.579, 0.22, 0.2)	(2429.962, 544.81)	276 277 288 291	2783 2675 2651 2635	Dominated Strategy Equilibrium "C"	(2716.067, 278.0552)	278	2646

Table 57. Comparison of the results for each replication in the instance m=2, n=20

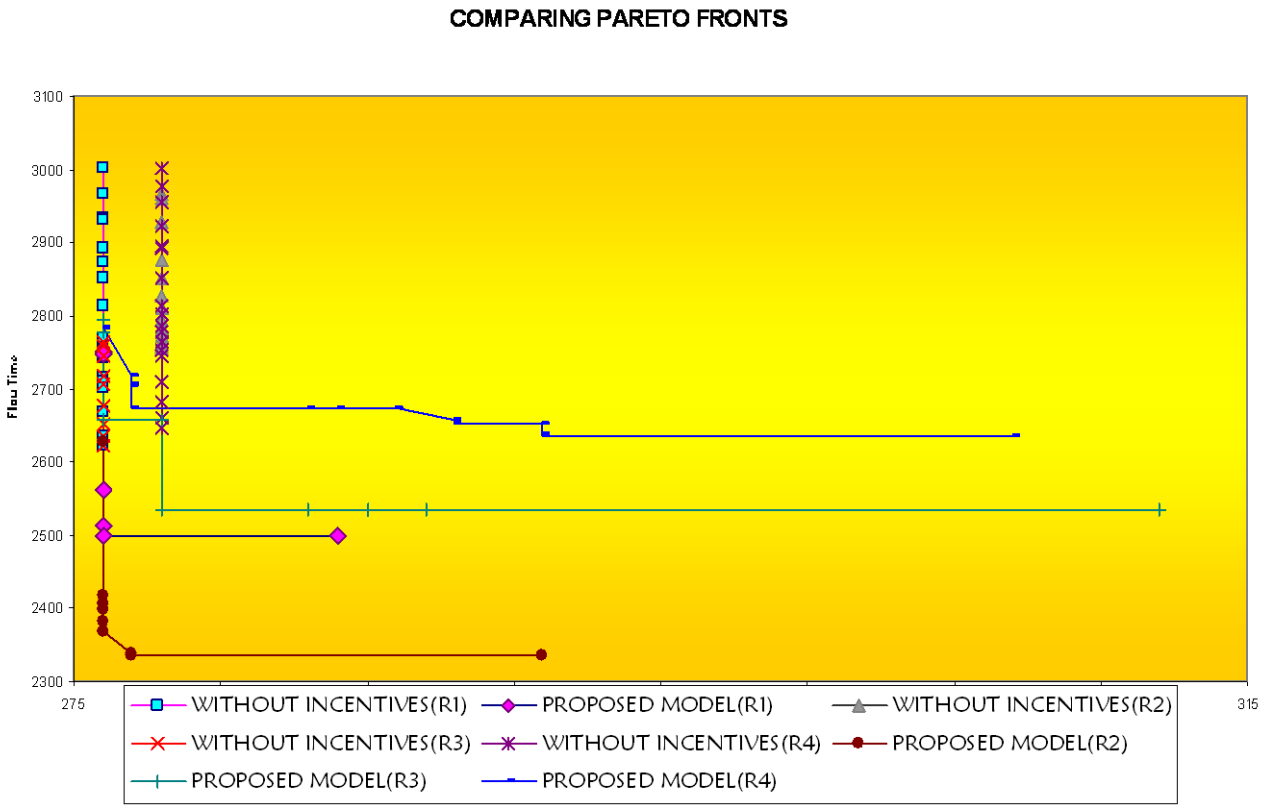

Figure 21. Comparison of the Pareto Fronts generated in the instance m=2, n=20.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Mixed Equilibrium (0.065 , 0.617, 0.237, 0.182)	(4835.527, 782.9151)	380 388 393	5151 5136 5071	Dominated Strategy Equilibrium "C"	(6094.126, 381.3467)	381	5981
2	Mixed Equilibrium (0.238 , 0.404, 0.322, 0.0072)	(5009.041, 689.6577)	380 381 393	5553 5243 5239	Dominated Strategy Equilibrium "C"	(5866.479, 382.0151)	381	5704
3	Mixed Equilibrium (0.006 , 0.629, 0.262, 0.192)	(4522.793, 730.8865)	380 382 385 388 390	4760 4617 4595 4503 4501	Dominated Strategy Equilibrium "C"	(5969.378, 380.2406)	380	5789
4	Mixed Equilibrium (0 , 0.582, 0.345, 0.107)	(5179.07, 761.514)	380 381 389	5605 5393 5377	Dominated Strategy Equilibrium "C"	(5956.931, 380.8133)	380	5618

Table 58. Comparison of the results for each replication in the instance m=2, n=30

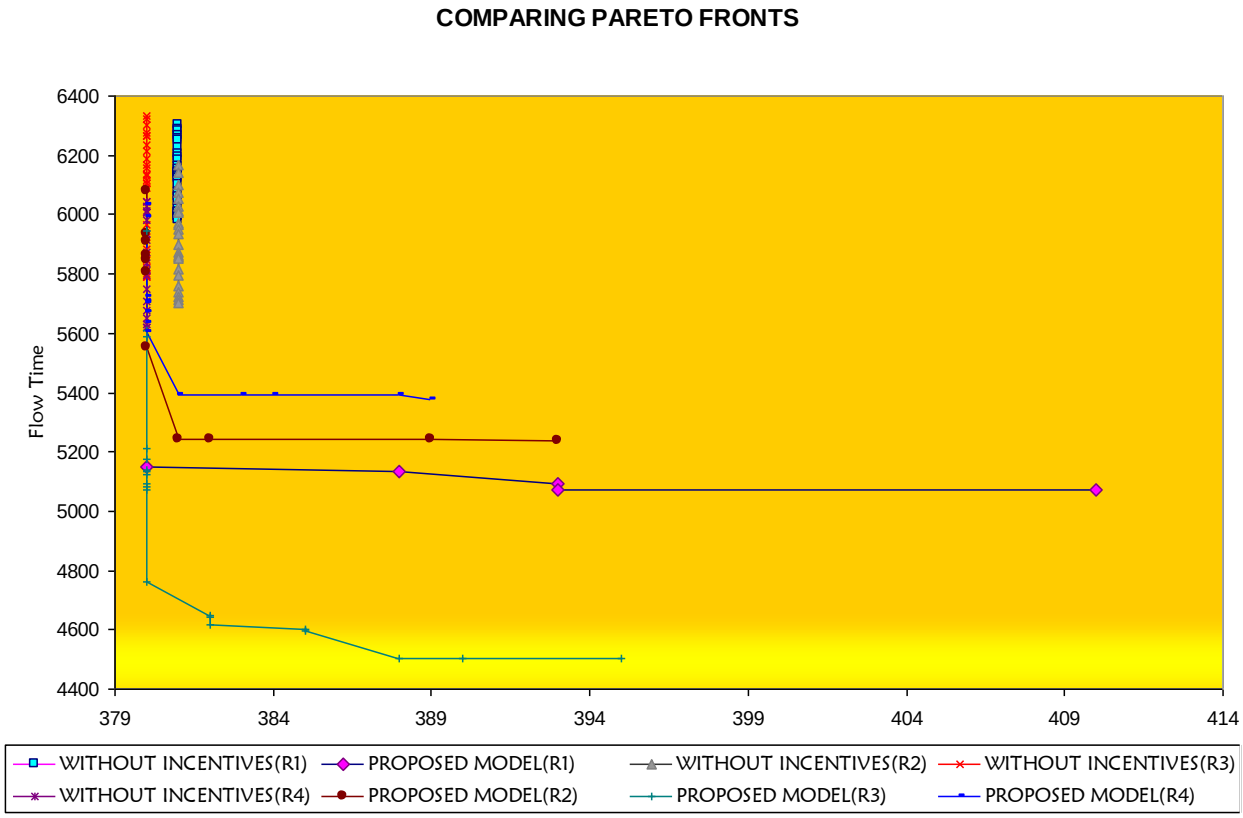

Figure 22. Comparison of the Pareto Fronts generated in the instance m=2, n=30.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Dominated Strategy Equilibrium "A"	(547.3831, 259.9696)	119 120 121	648 642 637	Dominated Strategy Equilibrium "C"	(653.4989, 120.1974)	120	647
2	Mixed Equilibrium (0.318, 0.495, 0, 0.208)	(590.7693, 170.372)	117 123	653 643	Nash Equilibrium "AC"	(693.0394, 120.1296)	119 121 122	694 690 689
3	Mixed Equilibrium (0.518, 0.147, 0, 0.351)	(595.979, 212.9776)	118 119	648 646	Nash Equilibrium "AC"	(654.8666, 117.0833)	117	652
4	Dominated Strategy Equilibrium "A"	(606.2706, 200.9731)	122	651	Dominated Strategy Equilibrium "C"	(692.4401, 121.0175)	121	690

Table 59. Comparison of the results for each replication in the instance m=3, n=10

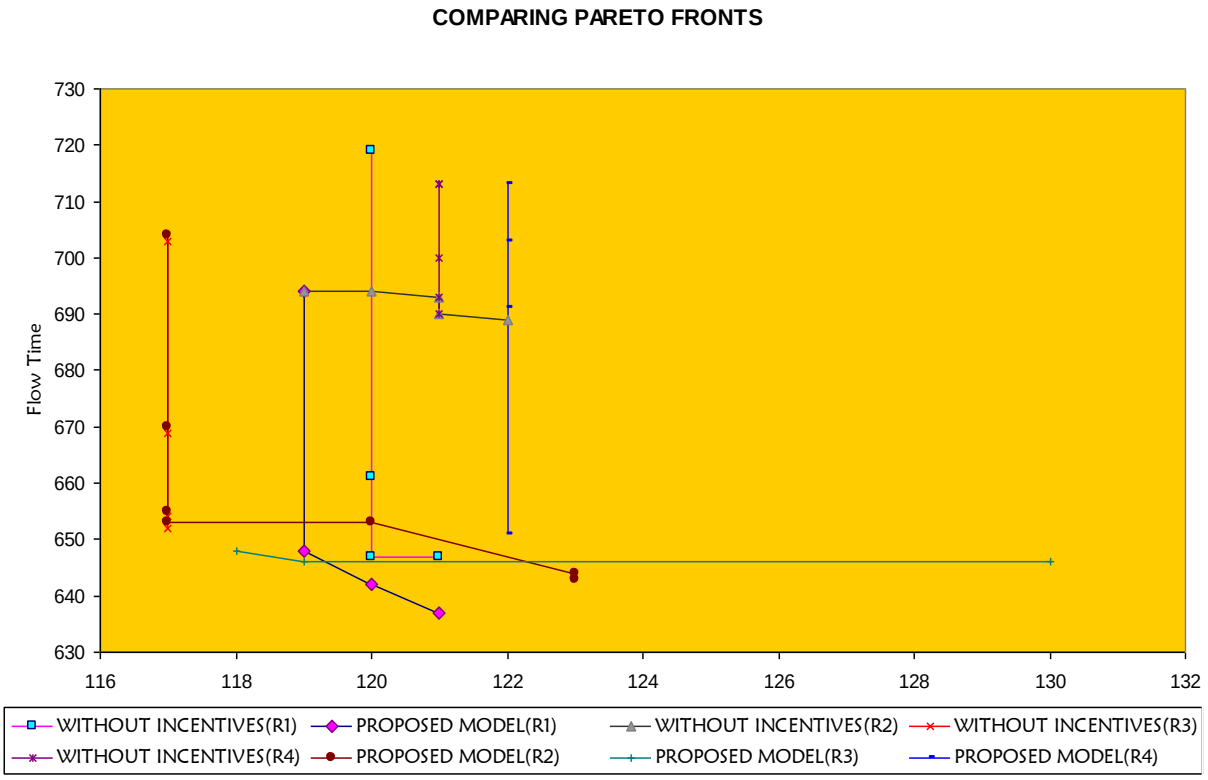

Figure 23. Comparison of the Pareto Fronts generated in the instance m=3, n=10.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Mixed Equilibrium (0.289 , 0.44, 0.248, 0.046)	(1577.921, 357.3643)	186 189	1708 1701	Dominated Strategy Equilibrium "C"	(1835.18, 189.9052)	187	1759
2	Mixed Equilibrium (0.263 , 0.476, 0.173, 0.109)	(1648.947, 329.824)	186 197	1751 1747	Dominated Strategy Equilibrium "C"	(1937.5, 187.5227)	186	1908
3	Mixed Equilibrium (0.244 , 0.598, 0, 0.189)	(1605.99, 347.7964)	185 191 202 206	1762 1756 1745 1739	Dominated Strategy Equilibrium "C"	(1873.534, 185.8609)	184	1811
4	Dominated Strategy Equilibrium "D"	(1543.262, 373.3035)	186 188 194	1703 1644 1638	Nash Equilibrium "AC"	(1873.602, 185.9792)	185	1822

Table 60. Comparison of the results for each replication in the instance m=3, n=20

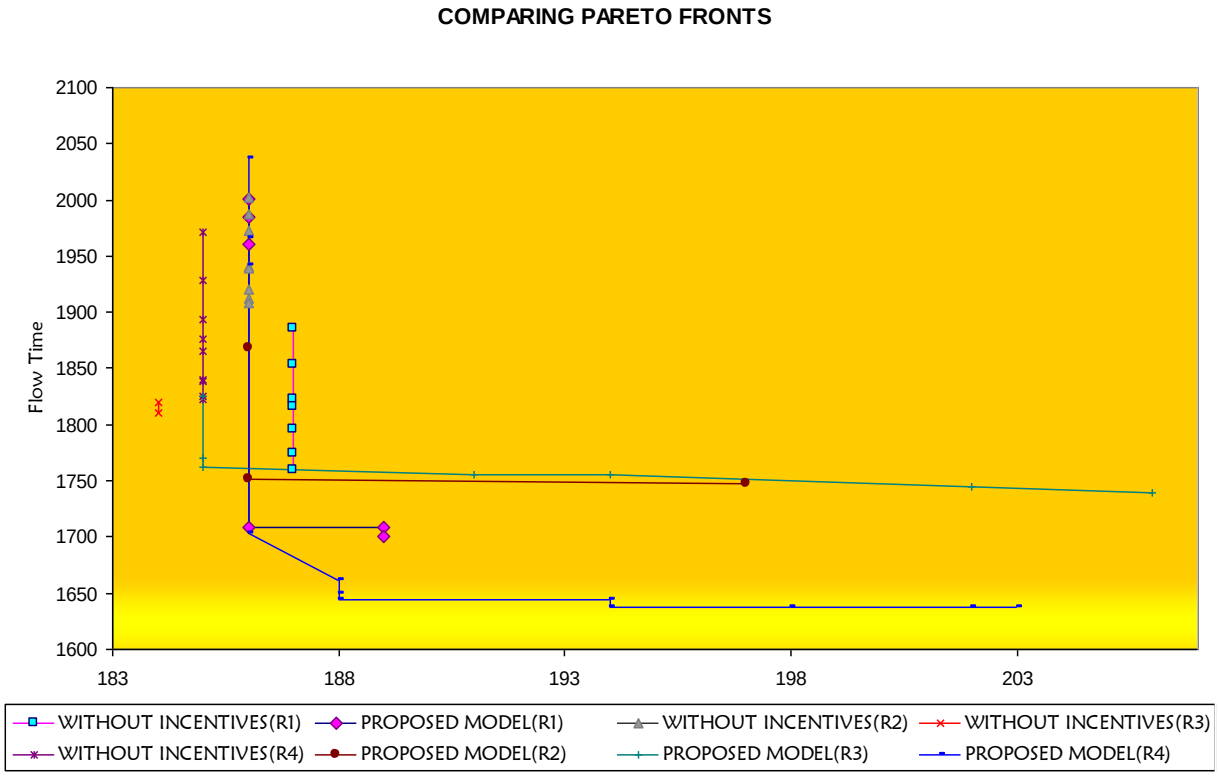

Figure 24. Comparison of the Pareto Fronts generated in the instance m=3, n=20.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Mixed Equilibrium (0.166 , 0.663, 0, 0.303)	(2902.073, 512.0789)	254 255 256 262 264	3219 3183 3088 3013 3011	Nash Equilibrium "AC"	(4195.184, 255.3853)	255	4132
2	Mixed Equilibrium (0 , 0.645, 0.355, 0.05)	(3030.533, 490.2077)	254 255 257 260	3130 3122 3099 3085	Nash Equilibrium "AC"	(4069.489, 254.645)	254	3961
3	Mixed Equilibrium (0.287 , 0.318, 0.257, 0.215)	(3319.317, 466.3869)	254 255 256 257 260 261	4147 3786 3776 3433 3423 3381	Dominated Strategy Equilibrium "C"	(4117.232, 254.3239)	254	4079
4	Dominated Strategy Equilibrium "D"	(3231.594, 520.7142)	255 256 257 260 263 273 297	3609 3436 3329 3324 3306 3212 3210	Dominated Strategy Equilibrium "C"	(3940.544, 254.6729)	254	3633

Table 61. Comparison of the results for each replication in the instance m=3, n=30

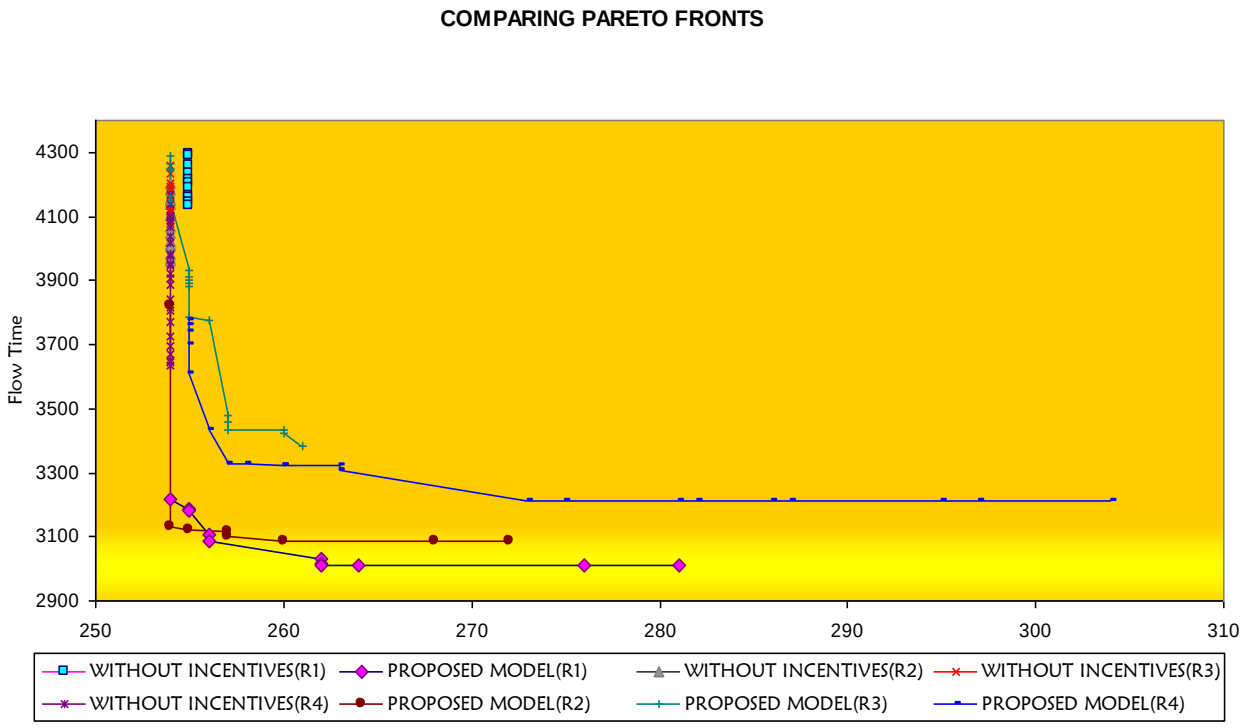

Figure 25. Comparison of the Pareto Fronts generated in the instance m=3, n=30.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***
1	Dominated Strategy Equilibrium "D"	(467.3407, 105.5651)	104	558	Dominated Strategy Equilibrium "C"	(552.3174, 101.381)	101 104	557 546
2	Dominated Strategy Equilibrium "A"	(488.1177, 152.0769)	90	535	Dominated Strategy Equilibrium "C"	(555.356, 92.47675)	91 94	558 546
3	Dominated Strategy Equilibrium "D"	(452.6545, 106.1271)	104	546	Dominated Strategy Equilibrium "C"	(553.8869, 101.9405)	101 104	558 546
4	Dominated Strategy Equilibrium "D"	(462.385, 105.7532)	104	546	Dominated Strategy Equilibrium "C"	(554.761.4401, 91.3709)	89	557

Table 62. Comparison of the results for each replication in the instance m=4, n=10

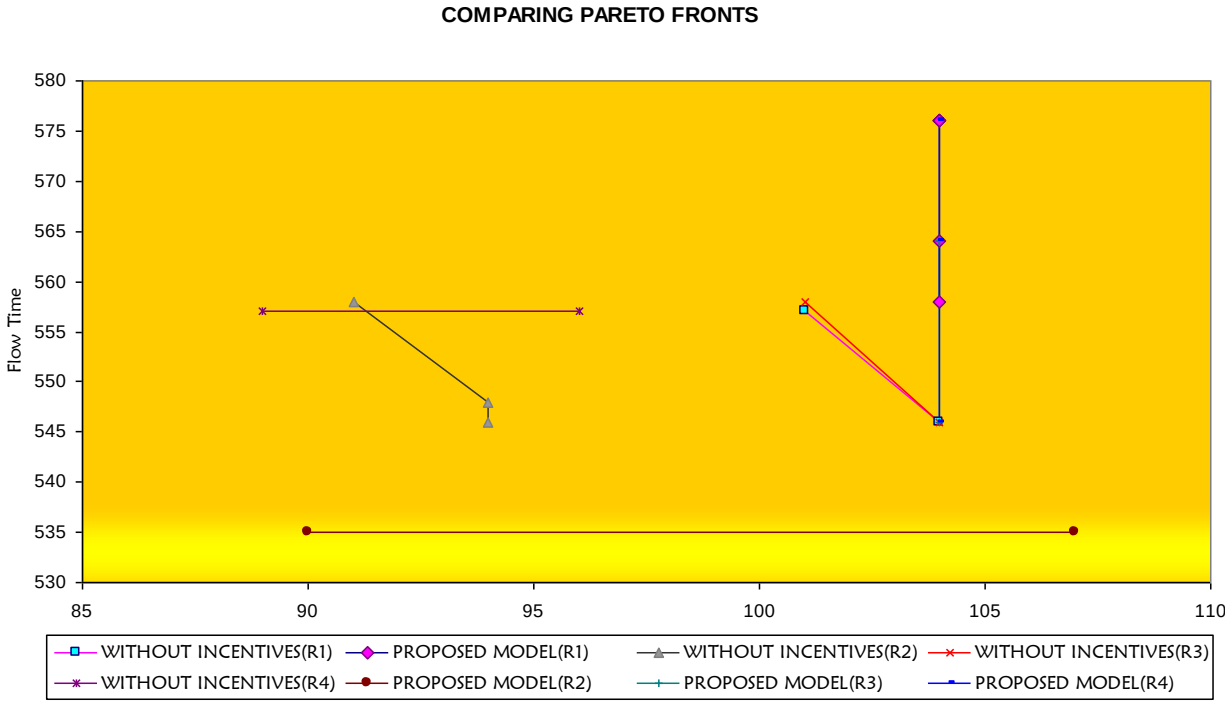

Figure 26. Comparison of the Pareto Fronts generated in the instance m=4, n=10.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***	**Type of Sol.**	**AvgCost (Ai,A0)**	**Cmax***	**Flow***
1	Dominated Strategy Equilibrium "D"	(128.216, 258.4572)	139 143 147	1346 1331 1330	Nash Equilibrium "AC"	(1526.455, 146.4815)	146	1518
2	Dominated Strategy Equilibrium "A"	(1298.394, 278.0583)	143 144	1371 1345	Nash Equilibrium "AC"	(1527.651, 146.5303)	146	1518
3	Mixed Equilibrium (0.079 , 0.743, 0.065, 0.127)	(1296.951, 296.3047)	147 158 160 168	1387 1386 1384 1383	Dominated Strategy Equilibrium "C"	(1477.574, 146.0606)	145	1439
4	Mixed Equilibrium (0.102 , 0.703, 0.3, 0.217)	(1243.722, 321.4114)	140 143 146 148	1419 1337 1334 1333	Dominated Strategy Equilibrium "C"	(1502.203, 142.469)	140 143	1500 1488

Table 63. Comparison of the results for each replication in the instance m=4, n=20

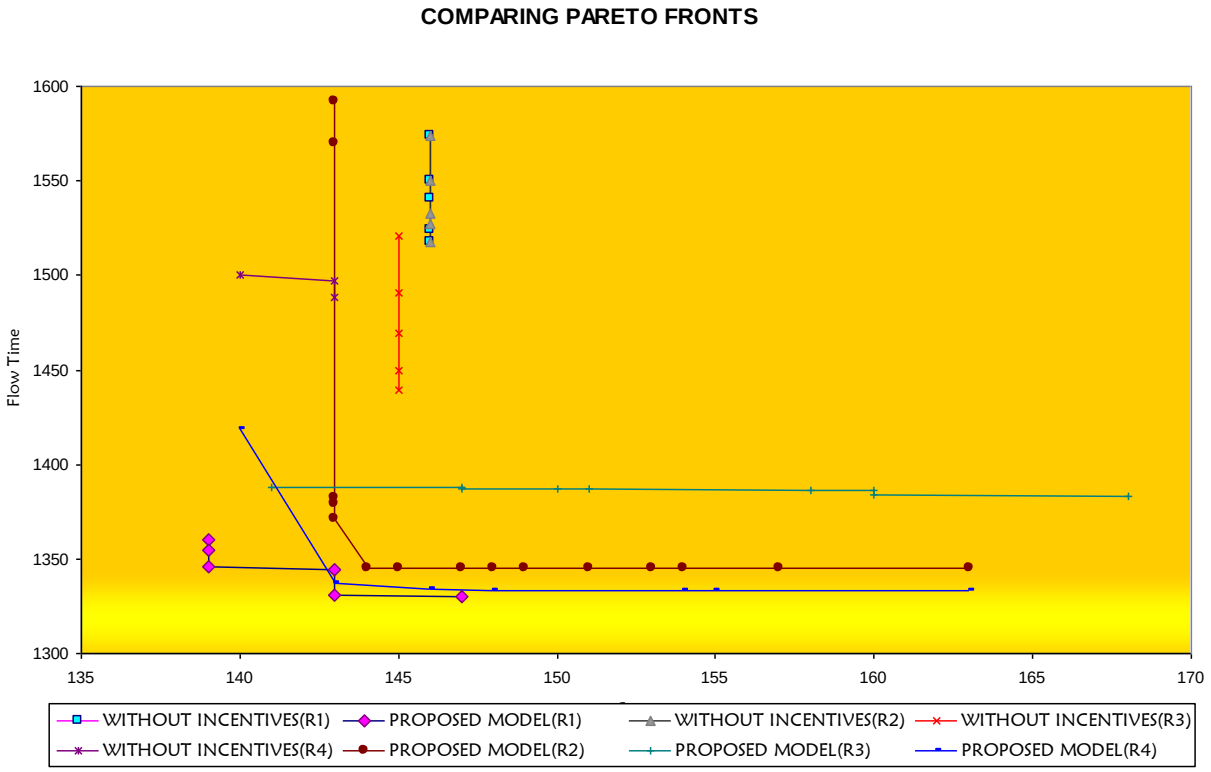

Figure 27. Comparison of the Pareto Fronts generated in the instance m=4, n=20.

	GAME WITH INCENTIVES (Proposed Model)				GAME WITHOUT INCENTIVES (Selfish Agents)			
r	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*	Type of Sol.	AvgCost (Ai,A0)	Cmax*	Flow*
1	Mixed Equilibrium (0.166, 0.504, 0.165, 0.235)	(2327.192, 357.1109)	193 196 198 201 205 213	2415 2413 2410 2402 2391 2388	Dominated Strategy Equilibrium "C"	(3087.931, 192.1463)	192	3053
2	Mixed Equilibrium (0.26 , 0.335, 0.245, 0.153)	(2343.506, 359.6672)	192 193	2392 2389	Dominated Strategy Equilibrium "C"	(3066.612, 192.7381)	192	3014
3	Mixed Equilibrium (0.117 , 0.512, 0.288, 0.141)	(2349.193, 354.3929)	191 192	2472 2380	Dominated Strategy Equilibrium "C"	(2908.359, 192.7368)	192	2753
4	Mixed Equilibrium (0.051 , 0.79, 0.159, 0.062)	(2244.355, 379.7274)	191 192 193 195	2888 2305 2305 2300	Dominated Strategy Equilibrium "C"	(3108.067, 195.6067)	195	3064

Table 64. Comparison of the results for each replication in the instance m=4, n=30

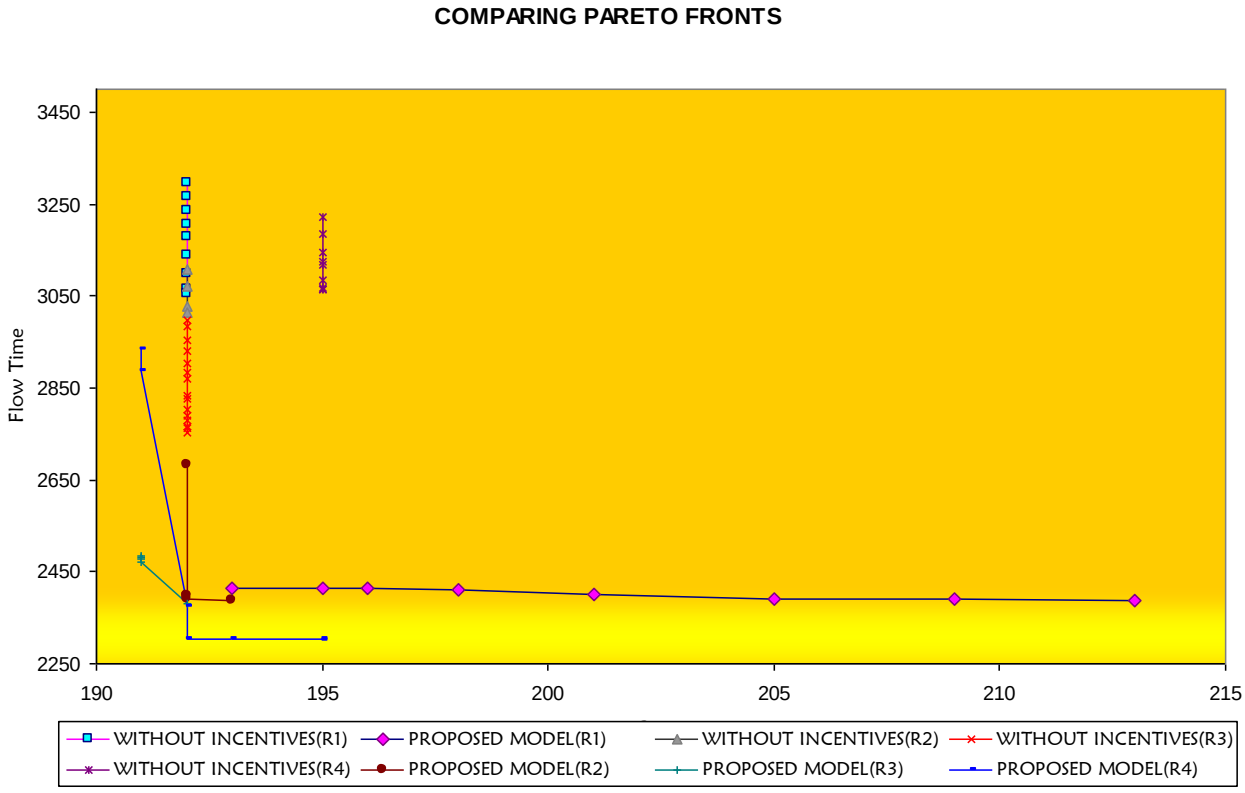

Figure 28. Comparison of the Pareto Fronts generated in the instance m=4, n=30.

For the first configuration of 2 machines with 10 jobs scheduled, it seemed as if the model that did not consider incentives, was the one that gave the best solution. Yet as the number of jobs and number of machines increased, the opposite is shown and in a greater scale. The solutions improved in very high percentages, being the greatest one the configuration of 30 jobs and 3 machines with a 22.1% of difference* in total flow time. Other four values for total flow time were observed to be improved in a high percentage, by the proposed model. The difference in C_{max} is not interesting in this analysis because the difference was taken from the best makespan of both values and in both models it is observed that the makespan is not that critical. Yet, if this analysis goes further, it can be observed how, for a group of Strict Pareto Solutions, one criteria is worsen by the others improvement. The most critical difference can be seen during the instance of 30 jobs in 3 machines, where the makespan was worsen by 16.47% while the total flow time was improved by 11.05%.

Another important conclusion is given by the robustness of the solution, where it is clearly observed that the model with incentives shows a more robust solution than the one without them. On the other hand, the model that did not consider incentives had almost on all the instances only one Strict Pareto Solution. A summary of these results can be seen in table 65.

* The values used for this calculation where the ones that shown to have a better makespan.

$$\%improvement = \frac{value\ for\ game\ without\ incentives - value\ for\ game\ with\ incentives}{value\ for\ game\ without\ incentives} * 100\%$$

			% improvent from the proposed model*		No. of Strict Pareto solutions found (with incentives, without incentives)	
m	*n*	*r*	Cmax	Flow	With	Without
2	10	1	0,00	-1,56	2	1
		2	0,00	0,00	1	1
		3	0,72	14,08	1	1
		4	0,57	3,55	1	1
	20	1	0,00	4,70	1	1
		2	0,72	14,08	2	1
		3	0,00	-1,37	2	1
		4	0,72	5,18	4	1
	30	1	0,26	13,88	3	1
		2	0,26	2,65	3	1
		3	0,00	17,77	5	1
		4	0,00	0,23	3	1
3	10	1	0,83	-0,15	3	1
		2	1,68	5,91	2	3
		3	-0,85	0,61	2	1
		4	-0,83	5,65	1	1
	20	1	0,53	2,90	2	1
		2	0,00	8,23	2	1
		3	-0,54	2,71	4	1
		4	-0,54	6,53	3	1
	30	1	0,39	22,10	5	1
		2	0,00	20,98	4	1
		3	0,00	-1,67	6	1
		4	-0,39	0,66	7	1
4	10	1	-2,97	-0,18	1	2
		2	1,10	4,12	1	2
		3	-2,97	2,15	1	2
		4	-16,85	1,97	1	1
	20	1	4,79	11,33	3	1
		2	2,05	9,68	2	1
		3	-1,38	3,61	4	1
		4	0,00	5,40	4	2
	30	1	-0,52	20,90	6	1
		2	0,00	20,64	2	1
		3	0,52	10,21	2	1
		4	2,05	5,74	4	1

Table 65. Improvement in the results and robustness of the solution presented by the proposed model.

A type of report was generated by the program during this simulation and is shown in Appendix B. It serves as an important interpretation to the type of solutions found in the long run which have been also summarized in the tables. An important conclusion can be stated from this report and it is that the model that does not consider incentives tends to choose the strategy “AC”, which means that the

agents prefer to stay on their current positions and machines. This is the behavior that is expected from selfish agents, according to game theory, where cooperation between them is not even considered. Yet, in the proposed model, the incentives indirectly make them cooperate and act in different ways. Almost all the instances for this model resulted in mixed equilibrium, which reflects how randomness affects the procedure. By observing the reports, it can be seen how solutions get better iteration after iteration with a few exceptions, until the possible movements the agents can make have all being recorded and there is no incentive to move to a new position.

7. RESEARCH ASSOCIATED COSTS

DESCRIPTION	VALUE
1.Researchers (2)	$ 18.000.000
2. Computing Equipment	
2.1. Computer	$ 2.600.000
2.2. Software Licence	$ 560.000
Total Computing Equipment	$ 3.160.000
3. Database Magazines for paid articles (10 art, each for US$30)	$ 690.000
4. Preparation of final document	
4.1. Ink charging (3)	$ 45.000
4.2. Paper (6 packets)	$ 72.000
Total Computing Equipment	$ 117.000
5. Indirect Costs Associated (energy, internet, transportation)	$ 500.000
TOTAL COSTS ASSOCIATED	**$ 22.467.000**

The description of these costs corresponds to a year of research, where the researchers develop the application in Microsoft Excel while investigating the different solutions obtained, resulting in the modification of procedures and ending in a final procedure solution. These values are estimated but correspond to a process that the researcher needs to follow.

8. ACTIVITIES SCHEDULE

ID	Task Name	Duration	Start	Finish
1	FIRST MEETING WITH THE DIRECTOR	1 day	Mon 06/02/06	Mon 06/02/06
2	PRELIMINARY SEARCH OF ARTICLES OVER THE INTERNET	10 days	Mon 13/02/06	Fri 24/02/06
3	PRELIMINARY SEARCH OF BOOKS	5 days	Mon 13/02/06	Fri 17/02/06
4	BIBLIOGRAFICAL REVIEW	15 days	Mon 10/04/06	Sun 30/04/06
5	VARIOUS THESES REVIEW	5 days	Mon 27/03/06	Fri 31/03/06
6	INTRODUCTION, BACKGROUND INFORMATIONS AND TERMINOLOGY	3 days	Mon 15/05/06	Wed 17/05/06
7	IDENTIFICATION OF THE PROBLEM, OBJECTIVES AND JUSTIFICATION	3 days	Thu 18/05/06	Mon 22/05/06
8	"MULTICRITERIA SCHEDULING" INVESTIGATION	40 days	Thu 01/06/06	Wed 26/07/06
9	"GAME THEORY" INVESTIGATION	40 days	Mon 08/05/06	Fri 30/06/06
10	FIRST CORRECTIONS MADE WITH THE DIRECTOR	3 days	Mon 05/06/06	Wed 07/06/06
11	REVIEW OF ALGORITHMS INVOLVING PARALEL MACHINES AND MULTICRITERIA OPTIM. TECHNIQUES	15 days	Mon 05/06/06	Fri 23/06/06
12	LITERATURE REVIEW	10 days	Thu 25/05/06	Wed 07/06/06
13	HIPOTHESES DEFINITION AND VARIABLES TO MEASURE THEM	2 days	Thu 08/06/06	Fri 09/06/06
14	DEFINITION OF THE METHODOLOGY	2 days	Thu 08/06/06	Sun 11/06/06
15	CORRECTION WITH THE DIRECTOR PREVIOUS TO THE FIRST DOCUMENT DELIEVERED TO IND. ENG. DEP	1 day	Mon 12/06/06	Mon 12/06/06
16	PRESENTATION OF PROPOSAL TO JUDGES	1 day	Mon 19/06/06	Mon 19/06/06
17	ADVANCED SEARCH OVER THE INTERNET AND DATABASE ARCHIVES	30 days	Mon 21/08/06	Fri 29/09/06
18	LEARNING HOW TO USE LEKIN SCHEDULING SYSTEM	1 day	Wed 20/09/06	Wed 20/09/06
19	DEFINING AND WORKING ON THE ALGORITHMS OR HEURISTICS CHOSEN	25 days	Mon 02/10/06	Thu 02/11/06
20	ASSESSMENT FROM THE DIRECTOR	3 days	Mon 30/10/06	Wed 01/11/06
21	CALCULATIONS AND RESULTS OBTAINED AND COMPARISONS	6 days	Fri 03/11/06	Fri 10/11/06
22	COMPARISON OF THE RESULTS - PARETO GRAPHS	1 day	Mon 13/11/06	Mon 13/11/06
23	SYNTHESIS OF THE RESULTS OBTAINED	1 day	Tue 14/11/06	Tue 14/11/06
24	LAST CORRECTIONS DONE WITH THE DIRECTOR	1 day	Wed 15/11/06	Wed 15/11/06
25	DOCUMENT REVIEW BY THE CORRECTOR	14 days	Thu 16/11/06	Tue 05/12/06
26	DOCUMENT DELIEVERED CORRECTED AND CD TO THE INDUSTRIAL ENG. DPT.	1 day	Wed 06/12/06	Wed 06/12/06
27	FINAL PRESENTATION DEFENDING THE ARGUMENTS PRESENTED	1 day	Mon 29/01/07	Mon 29/01/07

Project: PROYECT SCHEDULE
Date: Fri 01/12/06

Task | Rolled Up Task | External Tasks
Progress | Rolled Up Milestone | Project Summary
Milestone | Rolled Up Progress | Group By Summary
Summary | Split | Deadline

9. CONCLUSIONS AND FURTHER RESEARCH

9.1. CONCLUSIONS

Throughout the research we were able to approach scheduling in a rather new way, with regards to it, we have had encountered references about certain topics as controlling agents, mechanism design, which were the baseline for what we have accomplished so far, but we always had something different in our mind; which was the multicriteria approach, as a starting research; bicriteria, and we found nothing in literature concerning that bicriteria approach accounting intelligent agents. Basically, this topic is just opening and the perspectives for it are wide, our scope is very small for what can be done.

This research considered two conflictive criteria, makespan and total flow time among identical parallel machines, where two types of agents, each one looking after one criterion (makespan for Agent 0, total flow time for each job agent) started establishing tradeoffs through a 2x2 payoff matrix driven by the decisions that improved their condition. Once a game was played by the system controlling agent starting from the machine with the highest load and randomly selecting a job agent whose job belonged to that machine, and whose aim would be to wait less in queue started, a main setting was established: to drive agents to a point of

equilibrium. The only way to achieve this was through payoffs regarding opportunity costs that could compensate selfishness among agents. By establishing those concerns as baselines, a model was designed taking into account what could represent gains in terms of overall performance, and so drive agents to decisions that later would transform into swaps and switches among machines and timeslots within the system to finally reach schedules that could satisfy a level of effectiveness given by the Pareto Front. Once a Pareto Front was constructed, it had to be confronted with results obtained through different approaches in MCDM in order to test the closeness of these results found by the designed model.

Regarding the stated research's main objective, the outcomes that brought the model and the solutions that were gathered through the guidelines of game theory and scheduling, were tested and found a robust set of solutions that constructed a Pareto Front, but can be rather conservative with respect to best results found so far in the MCDM approach, yet, they are fairly close. For these reason, it can be stated that the sequencing of jobs found throughout the model can generate alternate sources of solutions for these types of configurations, which depends finally on the decision maker's choice.

The dynamic tradeoffs between the two intelligent agents and the fact that each one had a specific objective (which in terms of scheduling were conflictive),

allowed us to demonstrate that it is possible to consider more than one objective for these configurations of productive systems.

Apart from all the interesting consequences of this investigation, there were also some aspects that limited our research: the equations used for calculating the costs of the game, the randomness that affected our results, the fact that sometimes Nash equilibrium may not lead to overall performance because it is conservative since it is always based to minimize the maximum load, or maximize the minimum gain; it has been well proven, that Nash Equilibrium lacks strength in that sort of problems, e.g. the prisoner dilemma. By just considering single movement in timeslots at a time may also slant the potential outcomes, but it is a low risk gateway. Another fact that may limit our model is the way it has been played which generates solutions that start randomly but in the long run they may all tend to converge at some point, yet it is hard to tell whether all problems may converge, or what types of problems are the ones that converge.

As points in the Pareto Front were confronted, it can be said that scheduling and game theoretic interaction techniques enabled to approach to solutions where more than one criterion may be important, thus, we may consider accepting the alternate set of hypothesis stated in the corresponding section. The reason for this lies basically because although the solutions found constituted a Pareto Front, they tend to vary when finding the set of Pareto Solutions, due to the randomness of

this mechanism and the nature of game theory, where Nash equilibrium solutions have been demonstrated to be conservative, yet socially efficient.

At the same time, the way the alternate hypotheses were established, makes them more acceptable because they fit to most of the findings of our project, that is, they can be suitable for new topics and highlights the importance of the analyzing elements that later may serve as inputs for further research.

7.2. FURTHER RESEARCH

Doors towards further research are more than open. The results from this study can be sharpened in order to make solutions more robust and get to equilibria faster by considering more accurate equations concerning payoffs within the dynamic matrix. We have had certain limitations concerning how those functions worked all along, and we believe that in order to improve the solutions and outcomes, the following step is to study those parameters deeper and improve the variables that drive the decisions from agents and tradeoffs.

Branches that may originate from Game theoretic principles and approaches to productive systems can be said to be endless, since so many considerations can be taken into account. For instance, this same problem may have a big research potential if set up times are considered. Other considerations may also be

approached through the same principles, for example considering other objectives; working with due dates, or dead lines, different release dates, related and unrelated machines, matching outcomes for general flow shops or even workshops, etc. Game Theoretic principles are suitable to any environment where an intelligent agent has decisions to make that must satisfy someone else, who is also acting rationally.

BIBLIOGRAPHY

AGNETIS, Alesandro, MIRCHANDANI Pitu, PACCIARELLI Dario, PACIFI Andrea. “Scheduling Problems with Two Competing Agents”, 2004.PDF Document. p 229-242.

ALTMAN, Eitan; BASAR, Tamer; JIMENEZ Tania; and SHIMKIN Nahum.”Routing into two parallel links: Game-Theoretic Distributed Algorithms”. PDF Document. 20p.

ARCHER, Aaron, TARDOS Eva “Truthful Mechanisms for One-Parameter Agents” Symposium on Foundations of Computer Science. Computer Society PDF Document, 2002. PDF Document. 10p.

BERENBRINK, Petra, FRIEDETZKY, Tom, GOLDBERG, Leslie A., GOLDBERG, Paul, HU, Zengjian and RUSSEL, Martin. “Distributed Selfish Load Balancing”. Canada, Natural Sciences and Engineering Research Council of Canada. May 2, 2006. (PDF document) 17p.

BRAUN, Tracy D; SIEGE, Howard Jay; and BECK, Noah. “A Comparison of Eleven Static Heuristics for Mapping Class of Independent Tasks onto

Heterogeneous Distributed Computing Systems". Journal of Parallel and Distributed Computing , 2001.PDF Document 28p.

CARLSSON, Christer and FÚLLER, Robert. "Multiple Criteria Decision Making: The Case of Interdependence". 12p. http://citeseer.ist.psu.edu/cache/papers/cs/607/ftp:zSzzSzftp.abo.fizSzpubzSziamsrzSzco95p.pdf/carlsson95multiple.pdf.

CONWAY, Richard, et.al. "Theory of Scheduling". USA: Dover Publications, 1967. 309p.

COMPANY, Ramon, "Planeación y Programación de la producción" España: Barcelona, 1989. 39p.

DANIELS Richard L, **"**Analytical Evaluation of Multi-Criteria Heuristics**".** USA, Management. Science. Vol. 38, No. 4. Apr, 1992. p 501-513.

EYAL Even-Dar, KEESELMAN Alex, and YISHAY Mansour "Convergence Time to Nash Equilibra". School of Computer Science. Tel Aviv University. March 2006. PDF Document. 17p.

FINKE G, GORDON V and, PROTH J.M. “Scheduling with due dates (Annotated Bibliography of complexity and algorithms. Les cahiers du laboratoire Leibniz. Grenoble, France, 2002.PDF Document. 58p.

FUDENBERG, Drew and TIROLE, Jean. “Game Theory”. Massachusetts: MIT Press, 1991. 579p.

GHANDI Rajiv, “Dependent Rounding and its Applications to Approximation Algorithms”. Rutgers University, Canden, New Jersey Journal of the ACM, v 66, n 3, May 2006 p. 324-360.

GUTIERREZ, Eliécer and MEJÍA, Gonzalo. “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.”. Universidad de los Andes: Enero 25, 2006. (Pdf document form the World Wide Web) 6 p. http://triton.uniandes.edu.co:5050/dspace/bitstream/1992/812/1/Evaluacion+de+algoritmos+geneticos+para++el+problema+de+maquinas+en+paralelo.pdf

GUTPA, Jatinder; HO, Jonny; and WEBSTER, S. “Bicriteria optimization of the makespan and mean flow time on two identical parallel machines”. Bell State University. Muncie, Colombus State University, and Syracuse University, USA, 2000.PDF Document. 11p.

GUTPA, Jatinder N.D; and HO, Johnny C. “Minimizing Flow Time subject to Optimal Makespan on Two Identical Parallel Machines” Journal Pesquisa Operacional. V 20, n 1, June 2000. USA.

JANSEN, Klaus and MONALDO Mastrolilli “Approximation Schemes for Parallel Machine Scheduling with Controllable Processing Times”. Institut für Informatik und Praktische Mathematik, Universität zu Kiel, Germany. 2000.

MARIE, Michael and SEWANI, Anil “Combinatorial Algorithms & Data Structures”. Lecture 23: 4.14.05, Spring 2005, 4p.

KOUTSOUPIAS, Elias and PAPADIMITRIOU. ”Worst-case Equilibra”. Document in PDF. Berkley University and UCLA. PDF Document. 10p.

KOUTSOUPIAS, Elias. “On Mechanisms for Scheduling on Unrelated Machines”. Department of Informatics and Telecommunications, University of Athens, Summarized Presentation in PDF. August 2006. 23p.

KUMAR A, MARATHE M, PARTHASARATHY S,and SRINIVASAN A, “Aproximation Algorithms for Scheduling on multiple Machines”. Foundations of Computer Science, Department of Computer Science, Virginia Tech, University of Maryland, 2005. Summarized Presentation. PDF Document. 53p.

KUTANOGLU, Erhan and WU, S. David. “An Auction-Theoretic Modeling of Production Scheduling to Achieve Distributed Decision Making”.2003. (PDF document). 36p.

KUTANOGLU, Erhan and WU, S. David. “On combinatorial auction and Lagrangean Relaxation for distributed resource Scheduling”. USA: Leigh University.1998 (PDF document). 35p.

KUTANOGLU, Erhan and WU, David. “An Incentive Compatible Mechanism for Distributed Resource Planning”. Document in PDF. January 5, 2001. PDF Document.28p.

KWOK Yu-Kwong. ”Non-Cooperative Grids: Game-Theoretic Modeling and Strategy Optimization”, University of Southern California, Los Angeles, USA, December 2004. PDF Document. 30p.

LAI .T .C and SOTS Y.N. “Sequencing with uncertain numerical Data for akespan” USA: The Journal of Operational Research Society 1999 Vol 50 No3 (PDF document) 15p.

MULLER Tomas, “Interactive Heuristic Search Algorithm”. Department of Theoretical Computer Science. Praha, Czech Republic. 2003, 10p.

NISAN, Noam. “Algorithms for Selfish Agents: Mechanism Design for Distributed Computation”. Jerusalem: Institute of Computer Science. (PDF document) 17p.

OWEN, Guillermo. “Game Theory”. San Diego: Academic Press.1995. 447p.

PARKES David, “Price-Based Information Certificates for Minimal-Revelation Combinatorial Auctions”. Division of Engineering and Applied Sciences, Harvard University, USA. PDF Document. 2004. 20p.

PINEDO, Michael. “Scheduling: Theory, Algorithms, and systems”. New Jersey. Upper Saddle River.2002. 586p.

PINEDO, Michael and CHAO, Xiuli. “Operations Scheduling: With Applications in Manufacturing and Services.” USA: McGraw-Hill, 1999. 310p.

RESAT ALI, Tutuncuoglu, “Sequencing with earliness and tardiness penalties”. Department of Industrial Engineering, Bilkent University, Ankara, Turkey, April 22 1999, 11p.

RONEN, Amir. “Solving Optimization Problems Among Selfish Agents”. Jerusalem: Hebrew University, 2000. (PDF document) 92p.

SCHULTZ Andreas S and SKUTELLA Martin, “Scheduling Unrelated Machines by Randomized Rounding”, May 1999.

VITTORIO, Bilo; FLAMMINI, Michele; and MOSCARDELLI, Luca. “Pareto Approximations for the bicriteria scheduling problem”. Journal of Parallel and Distributed Computing, v 66, n 3, March, 2006, p 393-402 Taken from: Compendex.

APPENDIX A

PARTIAL RESULTS FOR SCHEDULING GAME
Number of Machines: 2
Number of Jobs: 10

ITERATION: 1
Cmax: 109
Flow time: 635
Strategy:BC
Job Agent: 9

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
3	30	38	46
4	19	57	103
7	21	78	181
8	23	101	282

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
2	46	46	46
5	4	50	96
9	6	56	152
6	36	92	244
10	17	109	353

ITERATION: 2
Cmax: 109
Flow time: 616
Strategy:BC
Job Agent: 10

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
3	30	38	46
4	19	57	103
7	21	78	181
8	23	101	282

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
2	46	46	46
5	4	50	96
9	6	56	152
10	17	73	225
6	36	109	334

ITERATION: 3
Cmax: 109
Flow time: 574
Strategy:BC
Job Agent: 5

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
3	30	38	46
4	19	57	103
7	21	78	181
8	23	101	282

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
2	46	50	54
9	6	56	110
10	17	73	183
6	36	109	292

ITERATION: 4
Cmax: 117
Flow time: 574
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
2	46	54	62
4	19	73	135
7	21	94	229
8	23	117	346

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
3	30	34	38
9	6	40	78
10	17	57	135
6	36	93	228

ITERATION: 5
Cmax: 128
Flow time: 563
Strategy:BD
Job Agent: 4

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
2	46	54	62
3	30	84	146
7	21	105	251
8	23	128	379

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
5	4	4	4
4	19	23	27
9	6	29	56
10	17	46	102
6	36	82	184

ITERATION: 6
Cmax: 122
Flow time: 569
Strategy:BD
Job Agent: 8

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
2	46	54	62
3	30	84	146
7	21	105	251
10	17	122	373

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
5	4	4	4
4	19	23	27
9	6	29	56
8	23	52	108
6	36	88	196

ITERATION: 7
Cmax: 115
Flow time: 569
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (current)

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Ji	Pi	Ci	PartialFi
5	4	4	4
2	46	50	54
9	6	56	110
8	23	79	189
6	36	115	304

ITERATION: 8
Cmax: 115
Flow time: 529
Strategy:BC
Job Agent: 9

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
2	46	56	70
8	23	79	149
6	36	115	264

ITERATION: 9
Cmax: 111
Flow time: 529
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
2	46	73	108
7	21	94	202
10	17	111	313

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
3	30	40	54
8	23	63	117
6	36	99	216

ITERATION: 10
Cmax: 115
Flow time: 529
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
2	46	56	70
8	23	79	149
6	36	115	264

ITERATION: 11
Cmax: 115
Flow time: 506
Strategy:BC
Job Agent: 8

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
8	23	33	47
2	46	79	126
6	36	115	241

ITERATION: 12
Cmax: 120
Flow time: 506
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
2	46	103	195
10	17	120	315

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
8	23	33	47
7	21	54	101
6	36	90	191

ITERATION: 13
Cmax: 115
Flow time: 506
Strategy:AD
Job Agent: 2

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
5	4	4	4
9	6	10	14
8	23	33	47
2	46	79	126
6	36	115	241

ITERATION: 14
Cmax: 115
Flow time: 508
Strategy:BC
Job Agent: 9

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
8	23	33	49
2	46	79	128
6	36	115	243

ITERATION: 15
Cmax: 122
Flow time: 508
Strategy:AD
Job Agent: 8

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
8	23	50	85
7	21	71	156
10	17	88	244

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
3	30	40	56
2	46	86	142
6	36	122	264

ITERATION: 16
Cmax: 115
Flow time: 508
Strategy:AD
Job Agent: 3

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92
7	21	78	170
10	17	95	265

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
8	23	33	49
2	46	79	128
6	36	115	243

ITERATION: 17
Cmax: 114
Flow time: 508
Strategy:AD
Job Agent: 6

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
3	30	57	92

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
8	23	33	49

7	21	78	170	2	46	79	128
6	36	114	284	10	17	96	224

ITERATION: 18
Cmax: 114
Flow time: 499
Strategy:BC
Job Agent: 7

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
7	21	48	83
3	30	78	161
6	36	114	275

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
8	23	33	49
2	46	79	128
10	17	96	224

ITERATION: 19
Cmax: 116
Flow time: 499
Strategy:AD
Job Agent: 7

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
8	23	50	85
3	30	80	165
6	36	116	281

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
7	21	31	47
2	46	77	124
10	17	94	218

ITERATION: 20
Cmax: 113
Flow time: 499
Strategy:AD
Job Agent: 6

Schedule for Machine 1 (current)

Ji	Pi	Ci	PartialFi
1	8	8	8
4	19	27	35
8	23	50	85
3	30	80	165
10	17	97	262

Schedule for Machine 2 (sec)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
7	21	31	47
2	46	77	124
6	36	113	237

ITERATION: 21
Cmax: 111
Flow time: 501

Strategy:BD
Job Agent: 7

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
7	21	29	37
8	23	52	89
3	30	82	171
10	17	99	270

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
4	19	29	45
2	46	75	120
6	36	111	231

ITERATION: 22
Cmax: 115
Flow time: 501
Strategy:AD
Job Agent: 4

Schedule for Machine 1 (sec)

Ji	Pi	Ci	PartialFi
1	8	8	8
7	21	29	37
4	19	48	85
3	30	78	163
10	17	95	258

Schedule for Machine 2 (current)

Ji	Pi	Ci	PartialFi
9	6	6	6
5	4	10	16
8	23	33	49
2	46	79	128
6	36	115	243

APENDIX B

CONFIGURATION: m=2, n=20 INCENTIVES INCLUDED ON PAYOFF MATRICES.
RESULTS ON THE EXPECTED RESULTS FOR EACH PLAYER IN THE SCHEDULING GAME

Iteration	Strategy	Cmax	flow	Costs Job Agent	Costs Agent 0
1	BC	287	3002	2919,024	582,4412
2	AD	287	2968	2842,644	579,364
3	AD	303	2968	2772,279	523,7333
4	AD	287	2968	2362,986	849,3374
5	AD	289	2968	2831,391	523,2143
6	BD	293	2968	2705,227	465,2171
7	AD	286	2975	2776,42	414,9189
8	AD	304	2975	2938	594
9	BC	297	2975	2881,189	775,6602
10	AD	297	2960	2472,514	649,0746
11	BC	281	2960	2873,532	666,3042
12	AD	281	2975	2924,677	558,8477
13	AD	280	2975	2926,656	542,9681
14	AD	281	2975	2924,677	558,8477
14	AD	281	2975	2924,677	558,847
14	AD	281	2975	2093,894	722,1053
15	AD	280	2975	2653,535	464,3265
16	BD	288	2975	2365,537	856,5783
17	BC	277	2986	2936	552,8361
18	AD	277	2976	2936	552
19	AD	278	2976	2936	552
20	BD	277	2976	2806,05	680,3846
21	BD	305	3006	2813,292	519,5366
22	AD	310	3001	2880,111	555,1265
23	AD	280	3001	2964	566
24	AD	283	3001	2808,78	534,2667
25	AD	286	3001	2814,8	524,3077
26	AD	312	3001	2964	630
26	BC	312	3001	1785,862	452,4
27	AD	312	2970	2932,915	644,4774
28	AD	313	2970	2926,504	669,7419
29	BC	302	2970	2839,064	710,7218
30	AD	302	2933	2896	610
30	BC	302	2933	2816,551	664,4
31	AD	302	2891	2854	610
31	AD	302	2891	2704,61	522,1367
31	AD	302	2891	2704,61	522,1367
31	AD	302	2891	2696,574	583,7248
32	BD	305	2891	2852,065	662,5
33	AD	297	2899	2822,694	553,3269
34	AD	278	2899	2853,41	476,0895
35	BD	289	2899	2856,46	687,1111
36	AD	286	2902	2865	578
36	BC	286	2902	2852,121	683,2222
37	AD	286	2887	2172	452,4333
38	BC	277	2887	2476,778	583,1579
39	BC	277	2869	2698,984	507,0089
40	AD	277	2842	2702,9	537,92
41	AD	288	2842	2378,96	541,5769
42	AD	281	2842	2802	556
42	AD	281	2842	2467,694	582,4116
42	AD	281	2842	2662,056	490,1429
42	AD	281	2842	2662,056	490,1429
42	AD	281	2842	2766,593	538,5663

43	AD	294	2842	2458,111	516,4512
44	AD	287	2842	2496,677	543,6348
45	BD	294	2842	2688,818	585,1328
45	AD	294	2842	2806,164	588,7421
46	AD	298	2842	2264,942	456,1224
46	AD	298	2842	2765,478	554
46	BC	298	2842	2698,852	567,3037
47	AD	298	2807	2770	602
47	AD	298	2807	2768,521	605,3721
48	AD	297	2807	2769,784	607,785
48	AD	297	2807	2229,102	455,7551
48	AD	297	2807	2762,867	575,8344
48	BC	297	2807	2586,875	576,7325
49	AD	297	2769	2724,867	575,8344
49	AD	297	2769	2389	516,8536
49	AD	297	2769	2732	600
49	BD	297	2769	2620,584	484,0086
50	AD	285	2799	2579,521	602,9615
50	AC	285	2799	2760,133	622,4928
51	AD	285	2799	2457,534	592,2817
51	BD	285	2799	2744,316	594,3497
52	AD	302	2782	2742	598
52	AD	302	2782	2531,295	476,7731
52	BC	302	2782	2603,218	702,4783
53	BC	302	2739	2525,771	604
54	AC	302	2704	2179,548	635,7895
54	BD	302	2704	2328,451	552
55	AC	298	2708	2194,903	596
55	AD	298	2708	2536,134	482,3492
56	AD	284	2708	2527,25	540,4407
57	AD	298	2708	2428,934	589,6479
58	BC	309	2708	2370,97	517,9014
59	AD	309	2674	2612,799	583,4598
60	AD	291	2674	2421,811	434,9189
61	BC	298	2674	2468,313	777,1373
62	AD	298	2645	2454,683	640,967
63	AD	299	2645	2608	604
63	BC	299	2645	2546,038	608,6786
64	AD	299	2629	2327,444	462,5916
65	AD	276	2629	2579,478	528,0849
66	BC	282	2629	2492,117	587,1147
67	AD	282	2598	2407,805	595,4747
68	BC	283	2598	2355,702	548,3125
69	AD	283	2572	2349,886	495,7273
70	AD	287	2572	2383,892	612,5
70	AD	287	2572	2510,096	553,1893
71	BD	283	2572	2521,369	535,2437
72	AD	290	2565	2504,077	547,3251
72	AC	290	2565	2480,725	559,3326
72	BC	290	2565	2230,5	448,8095
73	AD	290	2546	2506	574
73	BC	290	2546	1571,308	386,6667
74	AD	290	2529	2349,102	502,9661
74	AD	290	2529	2468,077	547,3251
74	BC	290	2529	2347,449	473,4188
75	AD	290	2506	2398,832	497,4468
76	BD	280	2506	2455	577,9819
77	BC	291	2495	1853,833	480,7826
78	AD	291	2468	2217,2	501,446
79	AD	279	2468	2420,072	560
80	AD	280	2468	2351,536	532,1978

81	AD	290	2468	2402,473	576,6053
82	BC	277	2468	2295,062	572,7797
83	AD	277	2457	1761,737	378,6897
84	BC	289	2457	2151,113	661,9032
85	AC	289	2435	2362,386	610,2091
85	AD	289	2435	2386,648	556,6957
85	AD	289	2435	2169,365	592,0225
85	BC	289	2435	2282,512	640,9872
86	AD	289	2417	2349,655	531,9029
87	BC	276	2417	2233,136	625,2245
88	AD	276	2406	2178,604	590,7317
88	BC	276	2406	2304,249	568,2353
89	BC	276	2397	2313,323	524,0714
90	AD	276	2382	2106,313	586,4151
91	AD	280	2382	2068,809	393,9804
92	BC	283	2382	2304,273	603,9018
93	AD	283	2376	2327,028	542,644
93	AD	283	2376	2328	552
94	BC	277	2376	1929,564	449,6118
95	BC	277	2379	1814,5	369,3333
95	AD	277	2379	2205,037	424,2329
95	BC	277	2379	2320,484	544,2071
96	BC	277	2371	1810,5	369,3333
96	BD	277	2371	2127,131	426,5083
97	AD	286	2362	2218,615	502,9702
98	BD	282	2362	2242,753	609,1892
99	AD	276	2368	2153,408	576,6667
99	AD	276	2368	2203,169	579,0909
100	AD	286	2368	2171,467	480,2813
100	AC	286	2368	1827,091	483,4762
101	AD	286	2368	2248,18	534,7399
101	AC	286	2368	1827,091	483,4762
101	AD	286	2368	2320	556
102	AD	278	2368	2214,827	536,8315
103	AD	282	2368	2246,896	554,9467
103	BC	282	2368	1763,5	376
103	AC	282	2368	1819,818	476,7143
103	BD	282	2368	2317,833	582,7541
103	AD	282	2368	2193,277	513,6989
103	BD	282	2368	2317,833	582,7541
103	AD	282	2368	2193,277	513,6989
103	AC	282	2368	1819,818	476,7143
103	AC	282	2368	1819,818	476,7143
103	AD	282	2368	2246,896	554,9467
103	AC	282	2368	1819,818	476,7143
103	BC	282	2368	1763,5	376
103	AD	282	2368	2193,277	513,6989
103	AD	282	2368	2226,023	527,6923
103	AD	282	2368	2246,896	554,9467
103	AD	282	2368	2174,074	479,2188
103	AD	282	2368	2226,023	527,6923
103	AD	282	2368	2246,896	554,9467
103	BD	282	2368	2317,833	582,7541
103	BC	282	2368	1763,5	376
103	AD	282	2368	2226,023	527,6923
103	BD	282	2368	2317,833	582,7541
103	AD	282	2368	2246,896	554,9467
103	AD	282	2368	2174,074	479,2188
103	AD	282	2368	2328	580
103	BC	282	2368	1763,5	376
103	BC	282	2368	2261,273	583,4483

104	AD	282	2349	2263,196	559,4381
105	BC	279	2349	2266,438	533,448
105	AD	279	2349	2301	552
105	AD	279	2349	2261,273	537,7353
106	BC	282	2349	2275,493	561,0538
107	BC	282	2338	2262,351	576,1946
107	AD	282	2338	2144,074	479,2188
107	AD	282	2338	2298	580
107	AD	282	2338	2163,277	513,6989
107	BC	282	2338	2262,351	576,1946
107	AD	282	2338	2252,196	559,4381
107	BD	282	2338	2233,129	533,9333
108	AD	281	2339	2164,277	516,6451
108	AD	281	2339	2164,277	516,6451
108	BC	281	2339	1752	374,6667
108	AD	281	2339	2144,096	478,9531
108	AD	281	2339	2253,268	558,5641
108	BC	281	2339	2263,336	571,0645
108	AD	281	2339	2238,778	531,4815
109	AD	287	2339	1801,913	629,5161
110	AD	285	2339	2222,699	552
110	AD	285	2339	1767,478	471,1176
111	AD	277	2339	2229,775	518,544
111	BC	277	2339	2283,176	542,7513
112	AD	277	2336	2288	552
112	AD	277	2336	2233,581	532,3171
113	AD	291	2336	2108,804	566,7222
113	BD	291	2336	2221,298	601,3671
113	BD	291	2336	2221,298	601,3671
113	AD	291	2336	2231,11	546,7105
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	1900,172	404,05
113	AD	291	2336	1900,172	404,05
113	AD	291	2336	2231,11	546,7105
113	BD	291	2336	2221,298	601,3671
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	1900,172	404,05
113	BD	291	2336	2284,188	610,7241
113	AD	291	2336	2141,726	577,4458
113	AD	291	2336	1898,8	575,2059
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	1900,172	404,05
113	AD	291	2336	2141,726	577,4458
113	AD	291	2336	2231,11	546,7105
113	BD	291	2336	2288	598,5634
113	BD	291	2336	2221,298	601,3671
113	AD	291	2336	2231,11	546,7105
113	AD	291	2336	2108,804	566,7222
113	BD	291	2336	2221,298	601,3671
113	BD	291	2336	2284,188	610,7241
113	AD	291	2336	2108,804	566,7222
113	AD	291	2336	2231,11	546,7105
113	BD	291	2336	2221,298	601,3671
113	BD	291	2336	2284,188	610,7241
113	AD	291	2336	2108,804	566,7222
113	BD	291	2336	2288	598,5634
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	1898,8	575,2059
113	BD	291	2336	2221,298	601,3671
113	BD	291	2336	2288	598,5634
113	AD	291	2336	1900,172	404,05

113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	2231,11	546,7105
113	AD	291	2336	1900,172	404,05
113	BD	291	2336	2284,188	610,7241
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	2141,726	577,4458
113	AD	291	2336	1900,172	404,05
113	BD	291	2336	2284,188	610,7241
113	AD	291	2336	2141,726	577,4458
113	AD	291	2336	2247,322	575,6108
113	AD	291	2336	1900,172	404,05
113	AD	291	2336	2231,11	546,7105
113	AD	291	2336	2231,11	546,7105
113	BD	291	2336	2221,298	601,3671
113	AD	291	2336	1898,8	575,2059

CONFIGURATION: m=2, n=20 NO INCENTIVES INCLUDED ON PAYOFF MATRICES
RESULTS ON THE EXPECTED RESULTS FOR EACH PLAYER IN THE SCHEDULING GAME

Iteration	Strategy	Cmax	flow	Costs Job Agent	Costs Agent 0
1	AC	287	3002	3002	287
2	AD	287	3002	3002	276
3	BC	276	3002	2967	276
4	BC	276	2967	2933	276
5	BC	276	2933	2930	276
6	AC	276	2930	2930	276
7	BC	276	2930	2893	276
8	BC	276	2893	2873	276
9	AC	276	2873	2873	276
10	AC	276	2873	2873	276
10	AC	276	2873	2873	276
10	BC	276	2873	2851	276
11	AC	276	2851	2851	276
11	AC	276	2851	2851	276
12	BC	276	2851	2814	276
13	BC	276	2814	2812	276
14	AC	276	2812	2812	276
14	BC	276	2812	2770	276
15	BC	276	2770	2753	276
16	AC	276	2753	2753	276
17	AC	276	2753	2753	276
18	BC	276	2753	2742	276
19	BC	276	2742	2715	276
20	AC	276	2715	2715	276
21	BC	276	2715	2710	276
22	AC	276	2710	2710	276
23	AC	276	2710	2710	276
23	AC	276	2710	2710	276
23	AC	276	2710	2710	276
23	AC	276	2710	2710	276
23	BC	276	2710	2702	276
24	AC	276	2702	2702	276
24	AC	276	2702	2702	276
24	AC	276	2702	2702	276
24	AC	276	2702	2702	276
24	AC	276	2702	2702	276
25	AC	276	2702	2702	276
25	AC	276	2702	2702	276

25	AC	276	2702	2702	276
25	AC	276	2702	2702	276
25	AC	276	2702	2702	276
25	AC	276	2702	2702	276
25	BC	276	2702	2668	276
26	BC	276	2668	2639	276
26	AC	276	2668	2668	276
26	BC	276	2668	2639	276
26	AC	276	2668	2668	276
26	AC	276	2668	2668	276
26	BC	276	2668	2639	276
26	AC	276	2668	2668	276
27	AC	276	2668	2668	276
27	AC	276	2668	2668	276
27	AC	276	2668	2668	276
28	BC	276	2668	2639	276
28	AC	276	2668	2668	276
28	AC	276	2668	2668	276
28	AC	276	2668	2668	276
28	BC	276	2668	2639	276
28	AC	276	2668	2668	276
28	BC	276	2668	2636	276
29	BC	276	2636	2607	276
29	AC	276	2636	2636	276
29	BC	276	2636	2622	276
30	BC	276	2622	2593	276
30	AC	276	2622	2622	276
30	AC	276	2622	2622	276
30	AC	276	2622	2622	276
30	AC	276	2622	2622	276
30	AC	276	2622	2622	276
30	BC	276	2622	2593	276
30	AC	276	2622	2622	276
30	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	AC	276	2622	2622	276
31	BC	276	2622	2593	276
31	AC	276	2622	2622	276
32	BC	276	2622	2593	276
32	AC	276	2622	2622	276
32	BC	276	2622	2593	276
32	BC	276	2622	2593	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	BC	276	2622	2593	276
32	AC	276	2622	2622	276

32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	BC	276	2622	2593	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	BC	276	2622	2593	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276
32	AC	276	2622	2622	276