COMPARING MIXED & INTEGER PROGRAMMING VS. CONSTRAINT PROGRAMMING BY SOLVING JOB-SHOP SCHEDULING PROBLEMS

Renata Melo e Silva de Oliveira
University of State of Para, Brazil
University of Porto, Portugal
E-mail: prdeig1209651@fe.up.pt

Maria Sofia Freire Oliveira de Castro Ribeiro
University of Porto, Portugal
E-mail: mariaribeiro4@gmail.com

Submission: 04/09/2014
Accept: 15/09/2014

ABSTRACT

Scheduling is a key factor for operations management as well as for business success. From industrial Job-shop Scheduling problems (JSSP), many optimization challenges have emerged since de 1960s when improvements have been continuously required such as bottlenecks allocation, lead-time reductions and reducing response time to requests. With this in perspective, this work aims to discuss 3 different optimization models for minimizing Makespan. Those 3 models were applied on 17 classical problems of examples JSSP and produced different outputs. The first model resorts on Mixed and Integer Programming (MIP) and it resulted on optimizing 60% of the studied problems. The other models were based on Constraint Programming (CP) and approached the problem in two different ways: *a)* model CP1 is a standard IBM algorithm whereof restrictions have an interval structure that fail to solve 53% of the proposed instances, *b)* Model CP-2 approaches the problem with disjunctive constraints and optimized 88% of the instances. In this work, each model is individually analyzed and then compared considering: i) Optimization success performance, *ii)* Computational processing time, *iii)* Greatest Resource Utilization and, iv) Minimum Work-in-process Inventory. Results demonstrated that CP-2 presented best results on criteria *i* and *ii*, but MIP was superior on criteria *iii* and *iv* and those findings are discussed at the final section of this work.

Keywords: Constraint Programming, Mixed an Integer Programming, Job-shop Scheduling Problem, Makespan minimization

1. INTRODUCTION

A typical Job-shop consists on a high-mix low-volume (HMLV) production flow, which simultaneously requires process of operations by the use of shared resources. In this manufacturing context, scheduling and sequencing operations became critical to the efficient use of both time and the machinery involved in a certain production system.

In this context scheduling is a well-known problem that deals with the efficient allocation of resources in order to perform a collection of tasks given a certain time range (DUMITRESCU; STOEAN; STOEAN, 2007). Thus, one of the challenges related to those issues is to reduce lead time by minimizing the amount jobs work in progress (WIP inventories). Then accordingly to Boushaala et al. (2012) and French (1982), a job-shop scheduling problem (JSSP) is complex and hard to be solved because of the following reasons:

i. Each job requires a different sequence of operations to be completed, which generates different jobs under processing simultaneously on different machines,

ii. Processing times for all jobs are known and constant,

iii. All jobs are available for processing at time zero,

iv. Machine absences are not allowed and each machine is continuously available for production,

v. There is only one machine of each type in the shop,

vi. Each machine can perform only one operation at a time on any job,

vii. An operation of a job cannot be performed until its preceding operations are completed

viii. Transportation time between machines is zero,

ix. A job does not visit the same machine twice.

x. There is no restriction on queue length for any machine.

xi. There are no limiting resources other than machines/workstations

xii. The machines are not identical and perform different operations

xiii. Operation cannot be interrupted,

xiv. An operation of a job can be performed by only one machine.

xv. There are capacity limitations which lead to bottleneck problems,

xvi. Due dates must be observed together with the completion times.

This work proposes the optimization on 17 classical Job-shop Scheduling problems (JSSP), under two perspectives: a) 1 Mixed Integer Programming model (MIP), and b) 2 Constraint Programming (CP) Models. Both of those techniques were applied aiming to minimize the makespan by sequencing the permutation of Jobs on the machines regarding the necessary order of processing.

Each model was set to obtain the best possible result given a range of 3600 sec, and after performing the simulations those 3 models performances are analyzed and commented.

Data used on this work was partially extracted from the work of Applegate and Cook (1991) and (BEASLY, 2005) and the approach to solve them was based on the work of Fisher (1973), Applegate and Cook (1991), Zhou (1996) and also by Mastrolilli (2000). Despites the reference works, this article does not aim to reproduce exactly the same results but to discuss the classical mathematical JSSP formulation and the computational solution obtained at IBM ILOG CPLEX environment under the light of Linear Programming perspective.

Then, on the next sections, a brief review of Linear Programming, Mixed Integer Programming and Constraint Programming will be presented, followed by JSSP mathematical model statement.

2. DEFINITIONS

2.1. A Brief Overview of Linear Programing

Linear Programming (LP) was first proposed by George B. Dantzig in 1947 as resource to the need of solving complex planning problems concerning to warlike operations during the World War II. LP is one of the most famous features of

Mathematical Programming, the later is defined by Dave, Dantzig and Thapa (1998) as follows:

“branch of mathematics dealing with techniques for maximizing or minimizing an objective function subject to linear, nonlinear, and integer constraints on the variables”

This fundamental concept is important to define the range of this study as the initial step taken to optimize the JSSP was to build an integer optimization model, composed by some of those elements mentioned above.

Continuing with the definitions, the Linear programming (LP) can be viewed as part of a great revolutionary development, which has given humankind the ability to state general goals and to lay out a path of detailed decisions to take in order to “best” achieve its goals when faced with practical situations of great complexity (DANTZIG, 2002). In order to be linear, an optimization model must satisfy 3 assumptions: proportionality, nonnegativity and additivity, which are described on Table 1.

Table 1: Conditions to linearity

Assumption 1: Proportionality
The quantities of flow of various items into or out the activity are always proportion to the activity levels. i.e.: it concerns to contribution per unit of each decision variable to the objective function.
Assumption 2: Additivity
Relates to the relationships among the decision variables. For each item it is required that the total amount specified by the system as a whole equals to the sum of the amounts flowing into the various activities minus the total amount flowing out. i.e.: The total value of the objective function equals the sum of the contributions of each variable to it.
Assumption 3: Nonnegativity
While any positive multiple of an activity is possible, negative quantities of activities are not. Ex.: A negative quantity of delivery packages cannot be negative.

Source: Adapted from DANTZIG (1996)

Summarizing it in a more scientific verbiage, Linear programming (LP) consists on the mathematical programming technique applied for finding optimal solutions to problems expressed in linear equations and inequalities (BRADLEY;

HAX; MAGNANTI, 1977). Generally LP aims is to find a vector $x \in \mathbb{R}_n$ maximizing (or minimizing) the value of a given linear function among all vectors $x \in \mathbb{R}_n$ that satisfy a given system of linear equations and inequalities. The linear function to be maximized, or sometimes minimized, is called the objective function and it presents the following form (MATOUSEK; GARTNER, 2007):

C T X = c 1 x 1 + · · · + c n x n

Where, $x \in \mathbb{R}_n$ is a given vector

Continuing, those linear equations and inequalities in the linear program are called constraints and a linear program is often written using matrices and vectors, in a way similar to the notation AX = b for a system of linear equations in linear algebra. Therefore, linear programs are problems that can be expressed in canonical form:

$$\mathbf{Max\ C^{T}\ X} \qquad (1)$$

Subject to:

$$\mathbf{A\ X\ \leq\ b} \qquad (2)$$

$$\mathbf{X \geq 0} \qquad (3)$$

The standard form of this kind of problem is:

$$\max c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \qquad (4)$$

Subject to

$$a_1 x_1 + a_{12} x_2 + \cdots + a_{1n} x_n = b_1 \qquad (5)$$

$$a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n = b_m \qquad (6)$$

$$x_1 \geq 0, \ldots, x_n \geq 0 \qquad (7)$$

The way usually pursued to solve such problems is the Simplex Method, which was introduced in the late 1940s, simplex evaluates from vertex to vertex on the boundary of the feasible polygon gradually improving the objective function until an optimal solution is found - or it is established that no solution exists (MATOUSEK; GARTNER, 2007). It is not the aim of this work to discuss step by step the methods to solve Linear Programs – even though Simplex method is very important to solve real world optimization problems. Thus, in order to obtain a detailed explanation of how solve many types of LP trough simplex or graphic method see bradley; Hax and Magnanti (1977) and Taha, (2007).

Once presented a quick overview of Linear programming, on subsection 2.2 the main definitions of mixed and integer programming are briefly presented.

2.2. Basic definitions of MIP

Mixed and integer programming is a part of mathematical programming dedicated to solve problems which require that the variables must be integers numbers. i.e. {0, 1, 2, 3,…, n}. Therefore, it focuses on discrete optimization problems (KAUFMANN, 1977). It is noteworthy that most of the integer problems are complex to be solved as the best solution with integer values of is not always obtained by taking the maximal solution of the program for continuous values and by then suppressing the decimal portion of it.

There are plenty of important issues that can be formulated as integer programming problems and solved by the use of the simplex method, such as i) Scheduling Problems (VANDERBEI, 2008). (e.g.: Equipment Scheduling and personnel scheduling), ii) The Traveling Salesman Problem, and ii) Fixed Costs problems. In the case of the examples i, ii, and iii, they present as property that the integer decision variables are binary. Because of the characteristics described above, the standard integer programming problem is define as:

$$\mathbf{Max\ C^T X} \qquad (8)$$

Subject to:

$$\mathbf{A\,X \le b} \qquad (9)$$

$$\mathbf{X \ge 0} \qquad (10)$$

$$\mathbf{X \in \mathbb{Z}} \qquad (11)$$

However, for problems in which the decision variables may assume any nonnegative integer value, it is necessary to resort to techniques such as the branch-and-bound method. Complementarily, that there is no single technique for solving integer programs and because of that a certain number of procedures have been developed for this purpose. They are broadly classified in 3 groups of three approaches:

i) Enumeration techniques, including the branch-and-bound procedure,

ii) Cutting-plane techniques and,

iii) Group-theoretic techniques. The first item on the list consists on the main resource applied by IBM ILOG CPLEX and because of the relevance of this work, it is important to present further explanations about it. For more detail about item ii and iii, see Bradley; Hax; Magnanti (1977) and Vanderbei (1998).

The Branch-and-bound intends solve an expected large number of correlated LP problems at the search for an optimal integer solution. Marie-France Derhy described this method as based on the principle that the total set of feasible solutions can be portioned into smaller subsets of solutions (DERHY, 2010), such as shown on Figure 1.

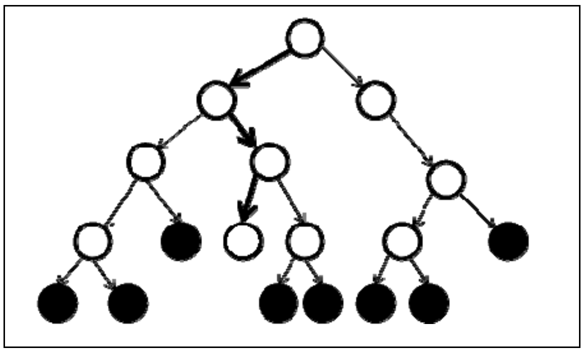
Figure 1- B&B illustrated procedure.
Source: Gurobi Optimization, (2014)

Continuing with the description, then the smaller subsets are evaluated systematically until the best solution is found. Whenever this method is used it is combined to a continuous solution method such as the simplex.

However, as an integer linear program is a LP only constrained by $x \in \mathbb{Z}$, in a minimization problem, the value of the objective function at the linear-program optimum will always be a lower bound on the optimal integer-programming objective, while any other integer feasible point is always a upper bound on the optimal linear-program objective value (BRADLEY, HAX and MAGNANTI, 1977). This process is repeatedly upgraded until an optimal solution if found or until every node is whacked. With this in perspective, it is important to present two more fundamental concepts:

- GAP: the difference between the current upper-bounds (UB) and lower-bounds (LB) is the gap
- INTEGER OPTIMAL: when [UB- LB] ÷ LB = 0, the integer optimal is achieved [6].

Although it presents limitations, MIP has proved to be very effective in modeling and solving both theoretical and practical optimization problems. Additionally, MIP consists on a special case of CP, despites the former represents a very important case of CP where all constraints and the objective function are required to be linear and only integer or real-valued domains are feasible accordingly to Salvagni(2008a) and Barták, (1999). With that idea on focus and considering that the second part of this work betakes Constrain Programming for Job-shop Scheduling Problems, the section is dedicated to present definitions on CP and its main features.

2.3. Basic Definitions of Constraint Programming

This section aims to present definitions related to constraint programming as well as briefly listing the main applications mentioned on literature. Literature Review on Constraint Programming is wide, stating in 1963 with the concept of general logical constraints by Sutherland in 1963 at his interactive drawing system Sketchpad (ACHTERBERG et al., 2008a). Later, during the 1970's further definitions of Constraint logic programming emerged in the artificial intelligence studies. Thereafter, in the following decade the constraint solving was incorporated into logic programming – when the work of Jaffart and Lassez(1987); Colmerauer (1990), among others gained prominence.

Constraint Programming (CP) is the study of computational systems based on constraints. It is an emergent paradigm to declarative model and e□ectively solve large, often combinatorial, optimization problems Salvagni(2008a). Then, because CP builds upon stating constraints and solving them, in this section some definitions related to this field are presented for later comparison to MIP. Summarizing it, a constraint program definition is a triple (BERTHOLD, NATURWISSENSCHAFTEN, 2008):

CP = $(\mathfrak{C}, \mathcal{D}, f)$ and consists of solving (CP) $f^* = \min\{f(x) \mid x \in \mathfrak{D}, \mathfrak{C}(x)\}$ with the set of domains $\mathfrak{D}$ = $\mathfrak{D}_1$× . . . × $\mathfrak{D}_n$, the constraint set $\mathfrak{C}$ = {C 1, … , C m }, and an objective function $f: \mathfrak{D} \rightarrow \mathbb{R}$. We denote the set of feasible solutions by $X_{CP} = \{x \mid x \in \mathfrak{D}, \mathfrak{C}(x)\}$. A CP where all domains $\mathfrak{D} \in \mathfrak{C}$ □ D are finite is called a finite domain constraint program (CP (FD)).

Complementing the definition, a constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. Furthermore about constraints (BARTÁK, 1999) :

i) They can specify partial information, i.e. there is no obligation of previously declaring variables value,

ii) Constraints are non-directional and they are declarative, i.e. they specify what relationship must hold without specifying a computational procedure to enforce that relationship,

iii) They possess additive propriety. Therefore, the order of imposition of constraints is irrelevant as the conjunction of constraints is in effect,

iv) Usually, constraints from the constraint store share variables.

CP has been successfully applied to a high variety of knowledge fields such as project management, whether industry or hospital scheduling. Further applications exemplified by Wallace (1996), such as Circuit Checking, Real-time control systems.

2.3.1. Constrain Programming Techniques

Constraint problems Techniques (solving technologies) can usually be categorized into 2 broad groups (BARTÁK, 1999): Constraint Satisfaction and Constraint solving.

The first group (Constraint Satisfaction) possesses strong relationship with Artificial Intelligence (AI) for solving Constraint Satisfaction Problems (CSP) which are stated as: a set of variables X={x1 ,...,xn }, where for each variable xi there is a finite set Domain (Di) of possible values. Also, there is a set of constraints that restrict the values that the variables can simultaneously take (LITTLE, TSANG, 1995). The possible values of the domain can whether be numeric or not, and in the case of D assume numeric values, there is no obligation for them to be integer. Therefore, the solution of a CSP will be accomplished when every variable assumes on value from the domain and all constraint are simultaneously satisfied.

A CSP allows multiple solutions depending on the goals can various solutions or only one can be found. Yet, is it still possible to obtain optimal solutions or even only a desirable one. In order to satisfy the constraints of such problems, following approaches are suggested (ACHTERBERG et al., 2008a):

- Consistence Techniques,
- Constraint propagation,
- Stochastic and Heuristics algorithms
- Systematic Search

Moving forward, the Constraint solving category covers the use of optimization based algorithms and mathematical techniques. A Constraint Satisfaction Optimization Problems (CSOP) consists on the same the definition of a standard constraint satisfaction problem (CSP) plus the requirement of finding optimal solutions (LITTLE, TSANG, 1995). Therefore the solution must comply with a previously defined objective function and at the same time it satisfies all the constraints together. In that context, the Branch and Bound (B&B) method is the most recurrent resource, which can be applied to the CSOP and to MIP problems as well (BARTÁK, 1999).

According to the same author, the B&B requires an heuristic function for mapping the partial labelling to a numerical value and in the case of a minimization problem such as the ones studied in this work, it represents an under estimate of the objective function for the best complete labelling achieved.

Thus, this kind of model searches for solutions in a depth first manner and behaves like chronological Backtracking1 except that as soon as a value is assigned to the variable, the value of heuristic function for the labelling is computed. If this value exceeds the bound, then the sub-tree under the current partial labelling is pruned immediately. Another way to address that type of problem is the use of Stochastic and Heuristics algorithms such as Genetic algorithms (GAs). GAs represent a class of stochastic search based on the concept of the evolution in

[1] "Backtracking" is a problem solving method according to which one systematically searches for one or all solutions to a problem by repeatedly trying to extend an approximate solution in all possible ways (FOKKINGA, 2004).

nature which successfully has been applied to combinatorial optimization problems such:

i) the travelling salesman problem (TSP),

ii) the quadratic assignment problem (QAP) and ,

iii) Scheduling Problems.

The plurality of constraint programming techniques is evident, as well as large range of applications. However, it is not the objective of this study to conduct a broad theoretical review on each technique related to this area of knowledge. In fact, it is noteworthy to recapitulate that this works intends to present a comparison between the different results obtained through a MIP model and several CP models aiming to optimize 17 hard Job-shop problems.

In this sense it is relevant to mention the work of Berthold and Naturwissenschaften (2008), in which it is presented the paradigm of Constraint Integer Programming (CIP). This author defends that MIP can be approached as a specific case of Constraint Programming and therefore, it is possible to integrate them. Aiming to do this, the author establishes that most problems of MIP – including the Job-shop Scheduling problems – can be treated as a Constraint Integer problem as long as the constraints are linear. Therefore, by definition, a Constraint Integer Programming (CIP) consists on solving:

$$\text{CIP} = (\mathfrak{C}, I, c) \qquad \textbf{(12)}$$

$$\textbf{where: CIP} \quad \mathbf{c^{*} = min\ \{c^{T}x \mid \mathfrak{C}\ (x)\ ,\ xj \in \mathbb{Z}\ for\ all\ J \in I\ \}} \qquad \textbf{(13)}$$

And $\mathfrak{C}$ = {C1…Cm} is a is a finite set of constraints Ci: $\mathbb{R}$n →{0,1}, i = 1, …, m, a subset I □ N = {1,…, n}, of a variable index set and an objective function vector c □ $\mathbb{R}$n .

A CIP must fulfil the conditions below:

$$\forall \hat{x} \in \mathbb{Z}^{I} \exists (A', B') : \{xc \in \mathbb{R}^{C} \mid \mathfrak{C}(\hat{x}, xv)\} = \{xc \in \mathbb{R}^{C} \mid A' xc \leq b'\} \qquad (14)$$

With

$$C: \quad = \quad N \quad \setminus \quad I, \quad A \qquad (15)$$

$$\in \mathbb{R}^{k \times c}, and\ b' \in \mathbb{R}^{k}\ \ for\ some\ \ k \in \mathbb{Z}_{>0}$$

The first constraint (equation 10) guarantees linearity to the problem after fixing the integer variables therefore, the problem can be solved by enumerating all values of the integer variables and solving the corresponding Linear Programs (BERTHOLD, NATURWISSENSCHAFTEN, 2008). This new paradigm set MIP problem as CIP, which allowed Job-shop scheduling problems (among many others) to be solved with hybrid approaches. Another contribution of ACHTERBERG et al. (2008b) and Berthold and Naturwissenschaften (2008) consists on establishing a parallel on both techniques, which can be observed on Table 2.

Table 2 –differences between MIP and CP

Constraint Programming (CP)
Domains of variables are (arbitrary) sets, Constraints are (arbitrary) subsets of domain space, High flexibility in modelling, natural but very general concept.
Mixed Integer Programming (MIP)
Domains are intervals in $\mathbb{Q}$ or $\mathbb{Z}$ Constraints and objective function are linear, Highly structured, specialized algorithms, restricted modelling
Constraint Integer Programming (CIP)
Linear objective function Arbitrary constraints, but fixing all integer variables always leaves LP (as in MIP)

Source: adapted from ACHTERBERG et al. (2008b)

Once presented main definitions that based this work, on the next section the proceedings of MIP study are presented, followed by the empirical study of CP.

3. JSSP GENERAL STATEMENTS

3.1. The objective function

Inputs of these JSSP consist on a set of Jn (jobs) x Mn (machines), where the due dates are not known, and there where specified two schedule decision criteria:

i) Maximization of the number of jobs,

ii) Minimization of the makespan and,

iii) J = finite set of jobs, J= {ji,….,jn},

iv) M = finite set of machines, M= {mi,….,mn}

For each j and m, let xjm be the starting time of a job j in machine m, and let Pjm be processing time of (j,m), where each Job possesses a predefined sequence of operations through the machines.

Also, every Jj at each Mm have a nonnegative integer processing time (Pjm) and the instant of a Jj enters into Mm to be process is denominated Xjm.

The objective of this problem consists on minimizing Makespan, which corresponds to the subtraction of completion time of last job and starting time of the first job (Cmax). Then, as the starting time of the first job must be the instant zero, the objective function corresponds to (APPLEGATE, COOK, 1991):

$$\textbf{Min } Z = C_{max} - 0 \qquad \textbf{(16)}$$

$$C_{max} = \text{Max } X_{jm} + P_{jm} \qquad (17)$$

Now that the objective function is defined, the declaration of the constraints is presented on the next section.

3.2. Constraints

The constraints established for this problem are:

$$X_{jm} \geq 0 \quad \text{for all } j \in J,\ m \in M \qquad (18)$$

$$X_{j(t)} \geq X_{j(t-1)} + P_{j(t-1)m} \quad \text{for all } \boldsymbol{t} = 2,\ldots,m \qquad (19)$$

$$X_{im} \geq X_{jm} + P_{jm} \textbf{ or } X_{jm} \geq X_{im} + p_{im} \qquad (20)$$

for all **I**, j ∈ J, m ∈ M

$$Z_x \geq X_{j(t)} + P_{j(t)m} \text{ for all } J \in J \qquad (21)$$

In order to solve this problem with the IBM ILOG CPLEX, a dummy variable was incorporated on constraint (5) so that this problem could be solved with MIP. The binary variable Ym (ij) assumes value one, whenever job i is scheduled on m before job j (7).

$X_{im} \geq X_{jm} + P_{jm} + K.\,(1 - \mathbf{Y}_{\mathbf{m\,(ij)}})$, $\{\mathbf{Y}_{\mathbf{m\,(ij)}} \in: 0 \leq Y \leq 1\}$

Or

$X_{jm} \geq X_{im} + P_{im} + K.\,\mathbf{Y}_{\mathbf{m\,(ij)}}, \{\mathbf{Y}_{\mathbf{m\,(ij)}} \in: 0 \leq Y \leq 1\}$ for all *I*, j ∈ J, (22)

m ∈ M

May K corresponds to some large constant.

Summarizing, the model is: {Min (16), Sub to (18), (20), (21) and(22) }.

4. JSSP FORMULATIONS

4.1. MIP formulation

After stating the mathematical model for the JSSP, in this subsection the MIP formulation to solve this problem is presented.

Objective function **Min C_{max}**

Constraints for all ***I***, j ∈ J, m ∈ M

Nonnegative times:

$X_{jm} \geq 0$ (23)

No-Preemption

$X_{j(t)} \geq X_{j(t-1)} + P_{j(t-1)m}$ (24)

for all ***t*** = 2,…,m

Dummy Variable:

$\mathbf{Y}_{\mathbf{ji-1,\,ji}} \in \{0,1\}$ (25)

for all $\boldsymbol{j}_{\mathbf{i-1}} \in$ **J >** $\boldsymbol{j}_{\mathbf{i}} \in$ J

Sequencing Criteria

$X_{im} \geq X_{jm} + P_{jm} + K.\,(1 - \mathbf{Y}_{\mathbf{m\,(ij)}})$
$X_{jm} \geq X_{im} + P_{im} + K.\,\mathbf{Y}_{\mathbf{m\,(ij)}}$ (26)

L *(sec):* Time limit: = 3.600 (27)

The objective function of this problem is consistent with equation (17), which is minimizing the completion time of all the jobs through finding the best sequencing. The constraints for this problem recall the Job-shop characteristics described on section one (items 1-14), which were translated to equations 23 to 26.

4.2. The CP -1 Model for JSSP

The CP -1 model consists on running the standard algorithm of IBM for default job-shop scheduling problems. This default model was conceived under the paradigm of setting discrete decision variables (processing intervals and sequencing machines) with the objective function of minimizing Makespan. For this kind of problem, IBM set the constraints according to the definitions of the Constraint Programming, and which are consequently aligned to the JSSP rules presented on section 1. After setting the constraints, the next step is to search for a satisfactory solutions, which is performed as illustrated on Figure 2.

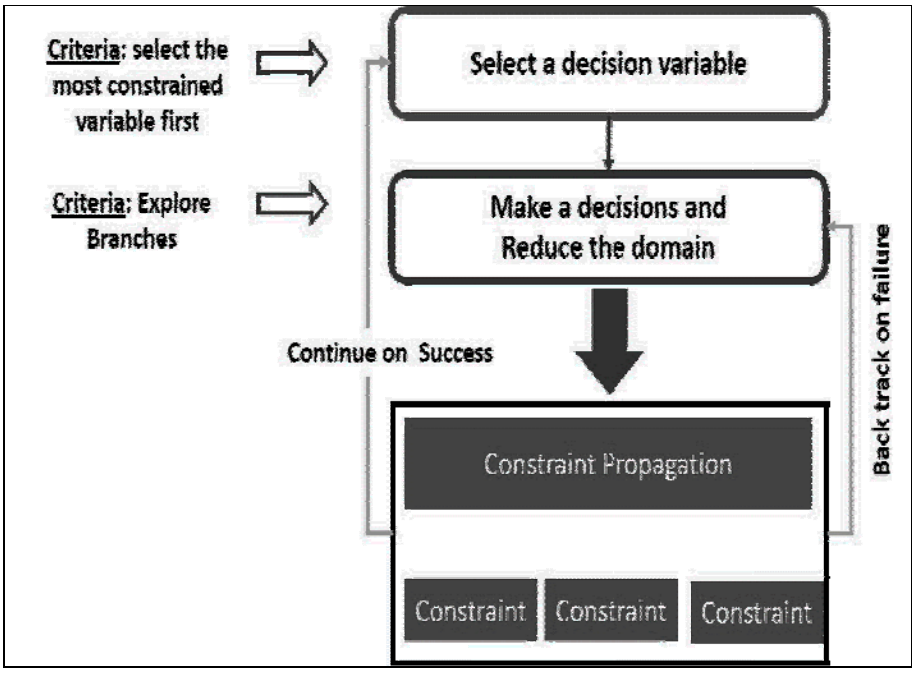
Criteria: select the most constrained variable first
Criteria: Explore Branches
Select a decision variable
Make a decisions and Reduce the domain
Continue on Success
Back track on failure
Constraint Propagation
Constraint
Constraint
Constraint

Figure 2: CP search process
Source:OPL(2009)

This initial model, here called naïve, presents the formulation shown below:

Objective function **Min** Makespan (28)

Constraints

No-Preemption : noOverlap $T_{j,m}$ (29)

Process Sequencing: endBeforeStart $(I_{j,m-1}, I_{jm})$ (30)

$L_{(sec):}$ Time limit: = 3.600 (31)

Where:

$T_{j,m}$ = Tuple operation of $j \in J$, $m \in M$

$I_{j,m}$ = Interval in which j is under processing at m

Eq.(28) does not differ from the other objective function previously discussed in section 3.1, but the constraints are presented with some differences:

i) NoOverlap : this constraint is used to prevent intervals in a sequence from overlapping and (optionally) to enforce a minimal distance between consecutive intervals (IBM, 2014).

ii) EndBeforeStart: this constraint states that the end of a given interval variable Ij,m -1 is necessarily less than or equal to the start of a given interval variable Ijm. (IBM, 2014).

iii) TimeLimit (sec): the solutions search was originally limited per number of fails. However, for this study this parameter was changed to TimeLimit of 3600 sec.

That implies that this search is not guaranteed to return the optimal solution, but the best one found within the limit available (IBM, 2014).

4.3. The CP -2 Model for JSSP

CP-2 model resembles definitions from section 3.1 and 3.2, from which is added the constraint of time limit Eq. (36). This proposition can be observed next.

Objective function

Min C_{max}

$$C_{max} = \text{Max } X_{jm} + P_{jm} \tag{32}$$

Constraints

Nonnegative times:

$$X_{jm} \geq 0 \quad \text{for all } j \in \text{J}, m \in \text{M} \tag{33}$$

No-Preemption

$$X_{j(t)} \geq X_{j(t-1)} + P_{j(t-1)m} \tag{34}$$

for all $t = 2,\dots,m$

Process Sequencing:

$$X_{im} \geq X_{jm} + P_{jm} \vee X_{jm} \geq X_{im} + p_{im} \tag{35}$$

$$L_{(sec):} \text{ Time limit: } = 3.600 \tag{36}$$

The sequencing criteria for CP-2 model does not differ mathematically from the one presented on model MIP excepting for discharging the use of dummy variables and, of course, by the use of the CP solver.

4.4. experimental results the test problem

Aiming to verify if the proposed models could successfully solve a JJSP, one short test problem was ran on IBM ILOG CPLEX for each model, before any attempt to run one of the hard proposed by (BEASLY, 2005).

This short problem was based on the JSSP description by i) Slack, Chambers and Johnston(2010), ii) (Boushaala et al., 2012)and French (1982) and it consists on a 3 jobs x 4 machines, in which, every job had to follow a predefined. The problem times and routes are presented on Table 3.

Table 3: Problem to test the model

Jobs	Machine Sequence	Processing Times in h (machine, job)
0	0,1,2	P_{01}=9, p_{11}=10, p_{21}=14
1	1,0,3,2	p_{12}=8, p_{02}=5, p_{32}=5, p_{22}=6
2	0,1,3	P_{03}=9, p_{13}=7, p_{33}=5

This problem required 48 seconds to be solved on a computer with processors 4 Intel® Core™ i7-4700MQ and 8GB of RAM, and it presented the following results (

Table 4).

Table 4: MIP test solution

Test JSSP	Size	Optimal C_{max}	MIP C_{max}	GAP (%)	D
MIP	3x4	39	39	0	0
CP1			39	-	0
CP2			39	-	0

D = deviation from best C_{max} I % [2]

The Cmax calculated through MIP matched the optimal solution and it is important to mention that the MIP model has reach optimality as the GAP calculated by IBM ILOG CPLEX equals to zero. With this in view, it can be inferred that the MIP model is functional and can be applied to more complex problems.

[2] D = $[C_{max\ (model)} - C_{max\ (optimal)}] \div C_{max\ (optimal)}$

Similarly to the model test performed on MIP, the results obtained though IBM default model CP-1 - here defined as naïve model - took 3 x 10-3 seconds to solve this problem and to find the best Cmax value.

Next, the CP-2 model is presented as an alternative to CP-1 model in the search for solutions. For testing Model CP2, the instance TEST was run in 1.05 seconds until the best Cmax was found.

Thus, as every model presented in this section has shown capability to find optimal results for the JSS test problem, their application on hard problems was proceeded. The results of this final stage of work, can be observed on section **Erro! Fonte de referência não encontrada.**.

Moving forward, as the job-shop scheduling problems (JSSP) are both scheduling and sequencing problems it is important for the operations management to provide a way of viewing the sequencing and timing instantaneously. For that reason, the Gantt chart was chosen to illustrate the results obtained for this initial problem - Figure 3.

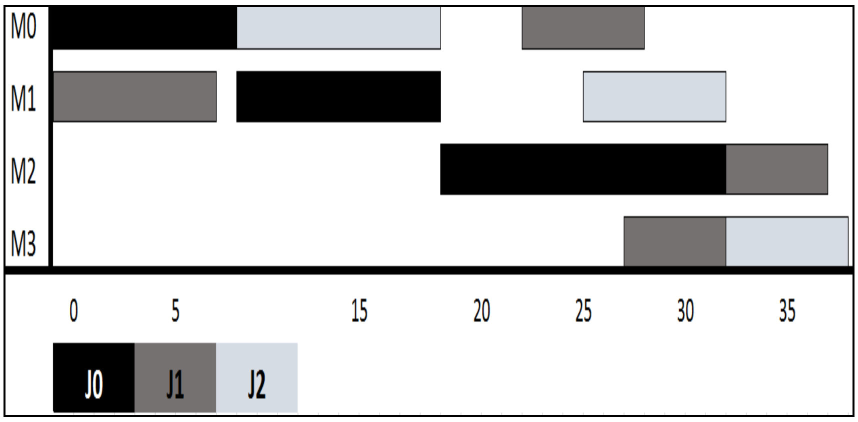

Figure 3: Gantt chart for JSSP test 1 (machines x time)

The Gantt chart consists on a bar graphic which pictures the schedule of a certain set of operations in the appropriate sequence. Through this chart it is possible to observe both start and finish times of each operation at the jobs involved on the problem.

Those start and finish times were initially calculated via IBM ILOG CPLEX. But, in order to pedagogically illustrate this JSSP, the following discussion will be supported by the Gantt chart.

Then, calculating the total waiting time of each job through the graph: J0 presented no waiting time, efficiently flowing all long the processing.

J1 had 15h of waiting time from machine 1 to machine 0. J2 had the sequencing with higher and also more frequent waiting time among machines: M0 (9 h), M1 (8h), M3 (1h). Therefore, the total waiting time for this job was 18 h.

Additionally, from the chart it is possible to extract the available time of each machine in this timeframe: i) M0 is free for 5hs between J2 and Job1, ii) M1 has 14h not occupied as between J0 and J2 there is 7hs and once J2 leaves this machine there are still 6h free to be used, iii) M2 has 18hs unoccupied before J0 starts to be processed, iv) and M3 has 28h free before J1 starts to be processed and also 1hs between J1 and J2.

Still, it is relevant to expatiate that because the sequencing results of CP are not identical to the one obtained from MIP model, Figure 3 will suffice to illustrate the obtained sequencing results. Once this solution for the test problem presented no deviation from its optimal, the next step taken was to run further complex problems in order to observe how it fits them.

5. EXPERIMENTAL ANALYSIS

5.1. Studied instances

At this stage of work, there were selected 17 differently sized JSSP to be optimized by the algorithms written on IBM ILOG CPLEX. The sizes of the chosen problems are presented on Table 5.

Table 5: JSSP instances dimensions

Problems	**Quantity**	**Size**
LA06	1	15 X5
FT06	1	6X6
LA01, LA 02, LA05, LA08, LA03, LA04	6	10x5
ABZ5, ABZ6,LA19,LA20, ORB2, ORB5, MT10 (FT10), ORB 1, ORB3	9	10x10

FT06 optimal value was published by Fisher and Thompson (1973) and the others were previously published by Applegate and Cook (1991) and also

Zhou,(1996) and with a different approach by Mastrolilli (2000). The calculated Cmax, and their deviations from the optimal solution, which were obtained though models MIP, CP-1 and CP-2 are presented on Table 6.

5.2. Experimental Results

The experimental results were organized in a way such as the interpretation of the reader was facilitated. Therefore: i) Colum Opt shows the Optimal Cmax value for the problem, ii) Colum SMIP corresponds to the solution obtained by MIP model, iii) in the sequence, iv)SCP1 displays the solution btained by model CP-1, v) SCP2 exhibit the solution obtained by model CP-2. Finally, every Colum containing the symbol vi) Dn, shows the deviation from optimal Cmax. Deviations were calculated taking as an example footnote 2, on page 227.

Table 6: Results per model

Instance	Opt	S_{MIP}	D_{MIP}	S_{CP1}	D_{CP1}	S_{CP2}	D_{CP2}
ABZ5	1234	1238	-	1277	0,035	1239	0,004
ABZ6	943	943	-	948	0,005	943	-
FT06	55	55	-	55	0,000	55	-
LA01	666	666	-	666	0,000	666	-
LA02	655	655	-	662	0,011	655	-
LA03	597	597	-	647	0,084	597	-
LA04	590	590	-	655	0,110	590	-
LA05	593	593	-	593	0,000	593	-
LA06	902	926	0,03	1559	0,728	926	0,027
LA08	863	863	-	863	0,000	863	-
LA19	842	842	-	884	0,050	842	-
LA20	887	887	-	934	0,053	902	0,017
MT10	572	593	0,04	1062	0,857	937	0,638
ORB1	1059	1102	0,04	1079	0,019	1077	0,179
ORB2	860	888	0,03	907	0,055	888	0,032
ORB3	930	1038	0,12	1067	0,147	1024	0,101
ORB5	886	926	0,05	983	0,109	887	0,001

D_{model} = deviation from best C_{max} I %

Despites the limitation of time which was set in 3600 seconds for both models, the MIP achieved optimality on 6 problems: ABZ6, LA01, LA02, LA03, LA04, LA20. Other 4 problems were granted as optimized, but they presented small gaps3.

[3] Tolerance: Problems with gaps between 0.1% and 10% on their solutions were considered nearly optimal.

Concerning to those last problems, there should be a clarification: IBM ILOG is configured by default for a 10% tolerance on GAPS measure and that is why during the run of those 4 JSSP the algorithm was interrupted before 3600 sec. Those instances were: ORB01, ORB02, ORB03 and ORB05.

The CP-1 model presented the smaller time of solution processing, within a range from 0s03 sec (LA06) to 1,06 sec (ABZ06), with ORB 01 as an outlier (1390 sec). In fact, none problem solution was interrupted before exceeding the time limit. However from 17 problems only 3 presented no deviation from the best Cmax value of literature review. Continuing with CP-1 outputs, 6 instances presented solutions close to the target (optimal), with deviations inferior to 10%, which were : ABZ5 , ABZ6 , LA03 , LA19, LA20, and ORB2.

Although the satisfactory results, some other instances presented the highest deviation from the desired Cmax value, among the 3 models, such as: ORB3 (0,147), LA04 (0,11), LA02 (0,11) ORB5 (0,11) and especially LA06 (0,728), MT10 (0,856) that presented values superior than 70%.

Moving the analysis to model CP-2, the general processing time were much higher in comparison to its predecessor. The processing time range stands from 0,005 (LA08) to 1926 (ORB01).

On the other hand, deviation from the target were much lower:

i) There were successfully solved 9 problems: with zero deviations from optimal value (ABZ6, FT06 , LA01, LA02 , LA03 , LA04 ,LA05, LA08 , LA19) and There were 6 problems solved with divergence between 4% and 0,1%, see Table 6,

ii) There are only 2 solutions with diversion higher than 10% from optimal, which are: ORB01 (0,17) and MT10 (0,638).

At the stage of research, the standard model CP-1 was discharged as the best choice, because although it presented the best solving time, it failed to solve 53% of instances. Still, it was able to fully solve 3 problems: LA05, FT06 and LA01 as well as it nearly solved 35% of this set of problems4. For that reason, the Gantt Charts of

[4] Tolerance for CP models: problems presenting deviations from optimal between 0.1% and 10% on their solutions were considered nearly optimal.

CP-1 as well as its processing time were suppressed of the next subsection, but it can be observed on the Attachment Section.

By the MIP processing, most problems were optimized (or at least nearly optimized) in less than 400 sec. However, ORB1, ORB3 and ORB5 have reached the time limit of 3600 with gaps superiors to 11%. Because the tolerance level of gaps was set on 10%, it was considered that these problems were not successfully solved.

Moving forward to CP-2 model, its processing times obtained were very much lower than the previous one, by solving 88% of instances under 500 sec. The exceptions were ORB01 (1.926sec), OBR02 (871sec) and ORB03 (1.117sec), but notwithstanding the time, solutions presented deviations inferior to 10% on the first 2 orbs. To observe all processing times, see **Erro! Fonte de referência não encontrada.**.

In terms of time performance and solving success, model CP -2 has demonstrated to be most adequate to solve this particular set of problems. However MIP model also presented satisfactory results. Due to the times and success rates of MIP and CP-2 models have been considered virtually equivalent, a further exploration of results were elaborated. Thus, in the following subsection some further comments related to the operations management perspective are presented to aid the choice of the most appropriate model for the studied set of solutions.

5.3. Complementary Analysis

This complementary analysis of the 2 selected models is illustrated with the example of instance FT06. This instance is a 6 x 6 system, with optimal Cmax = 55h.

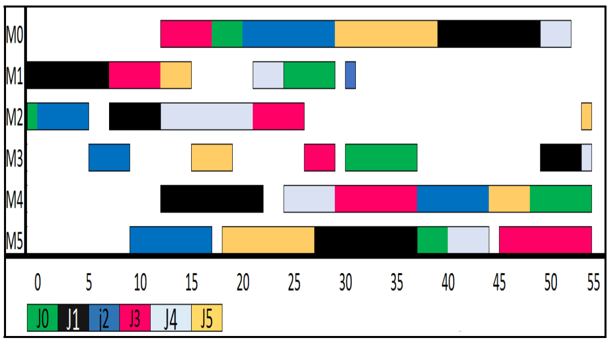

Figure 4: Gant chart for instance ft06 on MIP

Both models have arrived on the best results for its completion time, however the presented sequences differ from one another - differences on the sequencing can be observed on Figure 4 and Figure 5.

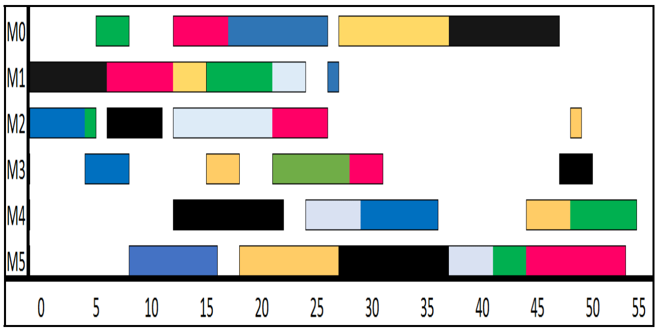

Figure 5: Gant chart for instance FT06 on CP-2

In other words, if only the initial criterion is maintained. (Min Cmax), both models can be considered equally eligible, but with subtle difference in accuracy between MIP and CP2.

For that reason, some other important performance issues were incorporated in to this study:

i) Reducing Work-in-process and

ii) Enlarging Resource Utilization (or reducing idle resources). There are many other performance criteria to guide decisions related to scheduling priority as defined by Brown, Blackmon and Cousins (2011):

iii) Level of customer service,

iv) Due time,

v) Factory efficiency.

The criteria chosen as a reference to support this section correspond i) & ii). Reducing work-in-process (WIP) and lead time stands out as one of the most critical objectives of Operations Management, such as defined by Slack, Nigel and chambers (2010) especially in job shop manufacturing.

The Work-in-process (WIP) inventories are goods at an intermediate stage between raw materials inputs to the process and finished goods. The design of the production process will greatly influence the level of work-in-process inventory(BROWN; BLACKMON; COUSINS, 2011).

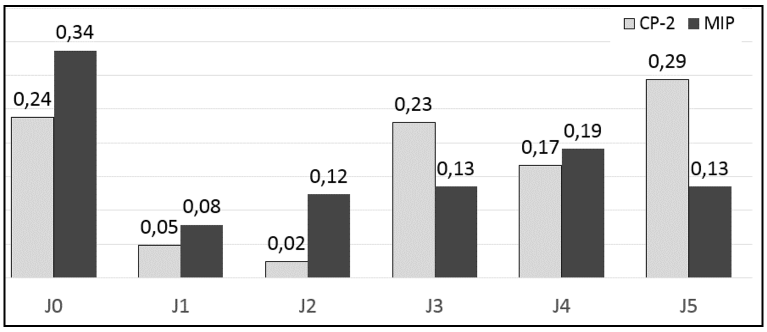

Figure 6 – Job´S WIP on ft06 in a 55h cycle

Considering the definitions presented above, Figure 6 illustrates how much time Jobs stay as WIP inventories in the sequencing of MIP Vs CP-2 model. From that graph it can be observed that J0 spends 34% (18,7hs) of the cycle time waiting for being processed at MIP, and 24% (13,2h). Considering a tolerance level per job of 10% for WIP, j0 results cannot be considered efficient. Referring to J1, the total waiting time in both cases are underneath the tolerance level of 10%. Summarizing, if the average WIP of each model are compared MIP, presents the lower WIP inventory time (14,8h or 26% of the cycle time), while CP-2 puts jobs on hold for 21h or 38% of the cycle time.

Maximizing (or at least) Enlarging the Greatest Resource Utilization (GRU) incurs on prioritizing sequences of activities that will result in a minimum idle time (ALHARKAN, 2005). That is also a measure of efficiency for many authors, as it attempts to minimize the waste of expensive means of production.

Thus, changing the perspective of the job for the machines, MIP sequencing also presents better performance in most of machines, with exception of M3 (Figure 7). However, CP-2 also surpasses 50% of the cycle time with this machine idle for the same machine

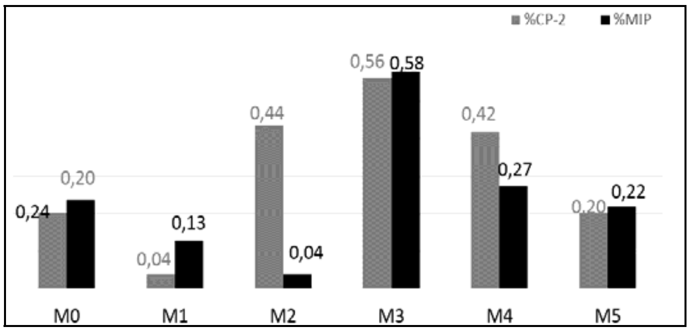

Figure 7 – machine´s idle time

Considering the average time machines are idle, there can be noticed that MIP solution is more efficient than CP-2, because CP-2 put machines on idle on average 17hs (that corresponds to 30% of 55hs) while MIP on average does not allocate tasks to the machines for 13 hours (24,5%). Concluding this analysis, the evaluation of those two models leads to the following implications:

- Considering the Objective function: Min Cmax

CP2- Model was considered the best model among three to solve this particular set of problems, as it successfully presented the optimal value at 88% of the studied instances as well as it fail to solve only one problem. In addition to that, this model has demonstrated the second best timing.

As a disadvantage, this model has shown inferior performance on reducing the WIP inventory and also on reducing idle machine times.

MIP model was considered the second best option because it presented a larger processing time, and because it successfully optimized 60% of the studied instances. This model also presented good (though not optimal solutions) for 23% of the JSSP and finally, it fail to optimize 12% of the problems. As an advantage, MIP model presented better performance at the use of machinery resources and also by presenting a shorter WIP inventory timing.

Once concluded the experimental analysis, the final considerations of this study are presented on Conclusions section.

6. CONCLUSIONS

This work aimed to present a comparison among 3 optimizing models, which are 1 MIP model and 2 CP Model, both had as objective to optimize 17 classical hard job-shop scheduling problems.

Through the study of those problems, it has been proved that the 2 algorithm succeed on optimizing the majority of problems as well as they observed the 16 premises of JSSP presented on section 1.

CP-2 has proven to be the most appropriate model to be faster on finding a close to optimal solution for 10X10 problems, while MIP was faster to find it on 10x5 problems.

Considering the computational processing time range of 3600sec, in spite of 79% slower, MIP has demonstrate more accuracy of results then CP-2 in 82,35% of the studied problems. Therefore, it is evidenced the trade-off between response speed and accuracy between those 2 models.

Another analysis should be effectuated by a decision maker who wished to choose between MIP and CP algorithms: the performance criteria of a given productive system. Those trade-off analysis enter into the domain of Operations Strategy, which is not the focus of this work and because of that it is suggested for the reader to see the work of Slack, Nigel and Lewis (2009) on the field of Operations Strategy.

As future works opportunities it is suggested to be taken a multi-criteria study aiming to simultaneously optimized Cmax, GRU and WIP or even any other performance criteria of Productive Systems in order to pursue better results. Another possibility is to deconstruct this work and redo it with the use of heuristics techniques, which would require more sophisticated tools in the search for new solutions.

Finally, this work is concluded with the confidence of not only presenting 2 functional Optimization algorithms for JSSP but also with certainty of having contributed to the demonstration of the many insights to the industrial management that MIP and CP can bring along.

7. ACKNOWLEDGMENTS

Thanks are due to CAPES for funding this work through the Program Science without Borders (BEX 19131127) and to Universitty of State of Para (UEPA).

REFERENCES

ACHTERBERG, T. et al. (2008) **Constraint Integer Programming**: A New Approach to Integrate CP and MIP. In: [s.l: s.n.]. p. 6–20.

ACHTERBERG, T. et al. (2008b) **Constraint Integer Programming : A New Approach to Integrate CP and MIP** (M. A. Perron, Laurent ; Trick, Ed.)Integration of AI and OR Techniques inConstraint Programming for Combinatorial Optimization Problems- 5th CAIPOR. **Anais**...Paris: Springer, Disponível em: <http://download.springer.com/static/pdf/854/bok:978-3-540-68155-7.pdf?auth66=1401445197_6c2a7a5412d043e9d7236f3536cf3058&ext=.pdf>

ALHARKAN, I. M. (2005) **Algorithms for Sequencing and Scheduling**. 1. ed. Riyadih: King Saud University, p. Alharkan, Ibrahim

APPLEGATE, D.; COOK, W. (1991) A computational study of the job-shop scheduling instance. **ORSA Journal on Computing**, v. 3, p. 149–156.

BARTÁK, R. (1999) **Constraint Programming : In Pursuit of the Holy Grail**Proceedings of the Week of Doctoral Students (WDS99). **Anais**...Prague: MatFyzPress.

BEASLY, J. E. (2014) **OR-Library**. Disponível em: <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt>. Acesso em: 10 fev.

BERTHOLD, T.; NATURWISSENSCHAFTEN, D. DER. (2008) **Constraint Integer Programming**. Paris: Zuse Institute Berlin.

BOUSHAALA, A. et al. (2012) **Genetic Algorithm based on Some Heuristic Rules for Job Shop Scheduling Problem**3rd International Conference on Industrial Engineering and Operations Management. **Anais**...Istanbul,: IEE, Disponível em: <http://iieom.org/ieom2012/pdfs/75.pdf>

BRADLEY, S. P.; HAX, A. C.; MAGNANTI, T. L. (1977) **Mathematical Programming: An Overview**Boston: Addison-Wesley Longman, Inc., Disponível em: <http://web.mit.edu/15.053/www/>

BROWN, S.; BLACKMON, K.; COUSINS, H. M. (2011) **Operations Management**. 1. ed. Boston: Elsevier, p. 449

COLMERAUER, A. (1990) **An Introduction to Prolog III** (J. W. Lloyd, Ed.)Computational Logic. **Anais**...Brussels: Springer, Disponível em: <http://link.springer.com/book/10.1007/978-3-642-76274-1>

DANTZIG, G. B. (1996) **Linear Programming**WaPrinceton University Press. Disponível em: <http://bioinfo.ict.ac.cn/~dbu/AlgorithmCourses/Lectures/Dantzig1963-1.pdf>

DANTZIG, G. B. (2002) Linear Programming. **Operations Research**, v. 50, n. 1, p. 42–47, fev.

DAVE, U.; DANTZIG, G. B.; THAPA, M. N. (1998) **Linear Programming-1: Introduction.** 1. ed. New York: Springer-Verlag, v. 49, p. 1226

DERHY, M.-F. (2010) Integer Programming : The Branch and Bound Method. In: **Linear Programming, Sensitivity Analysis & Related Topics**. 1. ed. New York: Pearson Education, p. 464.

DUMITRESCU, D.; STOEAN, C.; STOEAN, R. (2007) **Genetic Chromo-Dynamics for The JobShop Scheduling Problem**International Conference Knowledge Engineering Principles and Techniques. **Anais**...Romania: KEPT, Disponível em: <http://www.cs.ubbcluj.ro/~studia-i/2007-kept/307-DumitrescuStoean.pdf>

FISHER, H.; THOMPSON, G. L. (1973) Probabilistic learning combinations of local job-shop scheduling rules. In: J.F. MUTH AND G.L. THOMPSON (Ed.). **Industrial Scheduling**. 1. ed. New Jersey: Prentice Hall PTR, p. 128–139.

FOKKINGA, M. M. (2004) An exercise in Transformational Programming : **Science of Computer Programming**, v. 16, p. 19–48.

FRENCH, S. (1982) **Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop**. [s.l.] Ellis Horwood.

GUROBI OPTIMIZATION, I. (2014) **Gurobi Optimization**. Disponível em: <http://www.gurobi.com/resources/getting-started/mip-basics>. Acesso em: 31 maio.

IBM. (2014) **IBM ILOG CPLEX Optimization Studio**. Disponível em: <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r6/index.jsp?topic=/ilog.odms.ide.help/OPL_Studio/opllangref/topics/opl_langref_scheduling_sequence.html>. Acesso em: 5 out.

JAFFART, J.; LASSEZ, J. (1987) **Constraint Logic Programming** POPL '87 Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. **Anais**...New York: Spriger, Disponível em: <http://dl.acm.org/citation.cfm?id=41635>

KAUFMANN, A. (1977) **Integer and Mixed Programming: Theory and Applications -**. 1. ed. Pari: Academic Press Limited, p. 390

LITTLE, J.; TSANG, E. (1995) **Foundations of Constraint Satisfaction.** 2. ed. San Diegoeg: Academic Press Limited, v. 46, p. 666

MASTROLILLI, M. ; G. L. M. (2000) Effective Neighborhood Functions for the Flexible Job Shop Problem: Appendix (2000). **Journal of Scheduling**, v. 3, n. 1, p. 3–20.

MATOUSEK, J.; GARTNER, B. (2007) **Understanding and Using Linear Programming**. 1. ed. Berlim: Springer-Verlag, p. 229

OPL, I. B. M. I. (2009) **Optimization modeling with IBM ILOG OPL**IBM, Disponível em: <https://www.google.pt/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CC8QFjAA&url=http://lak.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/cplex/Workbook.pdf&ei=aE6KU4jCL6Kg0QXq8oCoDA&usg=AFQjCNFOVRPUBVjWh5wDN140Sx-ApTHVdg&sig2=ypBxl3aZGj0QnuLIH5Al0w&bvm=bv.67720277,d.d2k>

SALVAGNI, D. (2008) **And Programming Heuristic Methods for Constraint for Mixed Integer Linear Programs Mixed**. [s.l.] UNIVERSIT `A DEGLI STUDI DI PADOVA.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. (2010) **Operations Management with MyOMLab**. 6. ed. New Jersey: Prentice Hall PTR, p. 728

SLACK,NIGEL; LEWIS, M. (2009) **Operations Strategy**. 2. ed. Harlow: Prentice Hall PTR, p. 528

TAHA, H. A. (2007) **Operations Research: An Introduction**. 8th. ed. London: Pearson Education.

VANDERBEI, R. J. (1998) **Linear Programming: Foundations and Extensions**. New York: Springer, v. 49, p. 93–98

VANDERBEI, R. J. (2008) **INTEGER PROGRAMMING foundations and extension**. 3rd. ed. new: Springer, p. 469

WALLACE, M. (1996) Practical applications of constraint programming. **Constraints**, v. 1, n. 1-2, p. 139–168, set.

ZHOU, J. (1996) A constraint program for solving the job-shop problem. **Principles and Practice of Constraint Programming — CP96 Lecture Notes in Computer Science**, v. 1118, p. 1–15.