Toward Various Exact Modeling the Job Shop Scheduling Problem for Minimizing Total Weighted Tardiness

Namakshenas M [1], **Sahraeian R** [2]

Abstract In this paper, two different mixed integer programming (MIP) and one constraint programming (CP) models are formulated for classical job shop problem with the aim at minimization of the total weighted tardiness as objective function. The proposed models are solved and compared with well-known benchmarks in the job shop literature, using IBM ILog Cplex software. Examination and comparison of these exact models suggest that one formulation performs much more efficiently than others, namely CP model, in triple criteria: First, number of generated variables; Second, solution time and Third, complexity scale.

Keywords: Job Shop, Total Weighted Tardiness, Mixed Integer Programming, Constraint Programming

۱ Introduction

The classical job shop problems are the most prevalent issues in scheduling processes in factories in which high number of products should be produced each with custom orders. The usage of job shop algorithms is not restricted to production and manufacturing environment, but they can be implemented in services and other applications, too. With the rapid development in computational technologies, mixed integer programming (MIP) and other high-tech modeling concepts for solving scheduling problems are significantly receiving attention from researchers. Although, there is no efficient solution methodology due to the NP-hard nature of

[1] Mohammad Namakshenas (✉ e-mail: m.namakshenas@shahed.ac.ir)
Department of Industrial Engineering, College of Engineering, Shahed University, Tehran, Iran

[2] Rashed Sahraeian (✉ e-mail: sahraeian@shahed.ac.ir)
Department of Industrial Engineering, College of Engineering, Shahed University, Tehran, Iran

these problems, mathematical programming is the first step to develop an effective heuristic or algorithm.

Contrary to job shops with makespan objective, literature on solution procedures to the total weighted tardiness job shop scheduling problem (TWT-JS) is very limited. Given the only branch-and-bound algorithm for this problem proposed by (Singer & Pinedo, ١٩٩٧) the remaining approaches mainly based on local search (Essafi, et al., ٢٠٠٨; Mati, et al., ٢٠١١) and shifting bottleneck methods (Pinedo & Singer, ١٩٩٩).

In the widely used three-field notation of Graham, TWT-JS is written as:

$J_{m}\|\sum w_j T_j$, where $T_j = max(0, c_j - d_j)$.

The assumptions made for the classic job-shop problem are summarized here. The processing times are known, fixed, and independent of the sequence. All jobs are ready for processing at time zero. No preemption is allowed, i.e. once an operation has started it must be completed before another operation may be started on that machine. Also, recirculation is prohibited, i.e. all jobs should meet machines only one time and each job should meet all machines. Only one job may be processed on a machine at any instant of time. No restrictions are placed on the routings of jobs.

By this supposition, the problem can be declared as:

Given n jobs to be processed on m machines in a job shop with no restrictions on the routing constraints of jobs, determine the optimal job sequence on each machine in order to minimize the total weighted tardiness.

The rest of paper is organized as follows: In section (٠), first we introduce our notation used throughout the paper, then two integer-based formulation will be discussed. In section (٣) a constraint programming model will be developed in OPL terms. Next section has to do with comparison and analysis of proposed formulations and the last section concludes this study with a summary and conclusion.

٢ Mixed Integer Formulation

In this paper, we use the following notations for MIP formulation procedure:

Parameters & Sets:

n		number of jobs (N: jobs set);
m		number of machines (M: machines set);
p_{jh}		processing time of job j on machine h;
w_j		weight of job j;
d_j		due date of job j;
r_{jlh}	١	if the lth operation of job j requires machine h;
	٠	*o.w.*

Decision variable:

c_j completion time of job j;

s_{jh} starting time of job j on machine h;

Z_j $w_j.max(\cdot, c_j - d_j)$;

E_{kh} $w_k.max(\cdot, c_k - d_k)$ on machine h;

x_{jkh} { \ if job j scheduled in position k on machine h ; • *o.w.*

y_{ijh} { \ if job j follows job i on machine h (not necessarily immediately); • *o.w.*

٣.١ *Disjunctive Approach*

Disjunction based (DJ) formulation is closely intertwined with the disjunctive graph representation of the job shop. However, the relationship between disjunctions is implicit, which allows us to define the starting times s_{jh} with combination of a key binary variable, say y_{ijh}. Also, It suffices to work with these variables on the basis of $i < j$.

Objective: $$\text{Min} \sum_{j \in N} Z_j \qquad (١)$$

Constraints:

$$\sum_{h \in M} w_{\ j} r_{jmh} (s_{jh} + {}_{\cdot jm} - {}_{\ j} {}_{\ j} \le Z_j , \qquad \forall_{\ j} \in N, \qquad (٢)$$

$$s \quad - {}_{\ j} \ge {}_{\cdot\ j} - M y_{ijh} , \qquad \forall \ {}_{\ j} \in \quad \forall \ \in \qquad < j, \qquad (٣)$$

$$s_{\ j} - \quad \ge {}_{\cdot} \quad - \quad - y_{\ j}) , \qquad \forall \ {}_{\ j} \in \quad \forall \ \in \qquad < j, \qquad (٤)$$

$$\sum_{h \in} \qquad \sum_{\in M} r_{j,l+١,h} s_{jh} , \qquad \forall_{\ j} \in \quad \forall \ \in \quad - n\}, \qquad (٥)$$

$$Z_{\ j} \ s_{\ j} \in \qquad {}_{\ ijh} \in \text{binary}, \qquad \forall \ {}_{\ j} \in \quad \forall \ \in M .$$

Constraints set (٢) is used for linearization of the max function in TWT-JS and based on operational variable, namely r_{jlh}. Constraints set (٣) ensures that if the starting time of job j precedes i on machine h, then there must be delay with duration of processing job j on machine h and it must relax the constraints set (٤) and vice versa. Also, Constraints set (٥) imposes operational precedence of a job on specified machine. To be precise, it declares that starting time of operation l+١ should occur after the completion of operation l for job j.

٢.٢ *Hybrid Approach*

An alternative formulation can be constructed by deploying both the sequence-position variables and the precedence variables, namely hybrid formulation (HB). In order to establish a consistent model using two sets of variables, we define a variable v_{kh} for declaration the starting time of the scheduled job in the kth position for processing on machine h.

Objective: $\text{Min} \sum_{\cdot \in M} E_{kh}$ (٤)

Constraints:

$$E_{kh} \geq \sum_{j \in N} \sum_{=} \sum_{j \in N} x_{jkh} d_j w_j, \quad \forall \in , \forall \in M, \quad (٧)$$

$$\sum_{j \in N} x_{jkh} = 1, \quad \forall \in , \forall \in M, \quad (٨)$$

$$\sum_{k \in N} x_{jkh} = ١, \quad \forall \in \forall \in M, \quad (٩)$$

$$v \quad - \quad - \geq \sum_{j \in N} x \quad - \quad p_{jh}, \quad \forall \in \quad - , , \forall \in M, \quad (١٠)$$

$$M (٢ - \sum_{h \in} \sum_{\in} \sum_{\in} \sum_{\in} \sum_{\in M} r_{jlh} p_{jh},$$

$$\forall \in \forall \in - m\}, \quad (١١)$$

$$Z_j \; v_{\kappa n} \in \quad _{jkn} \in \text{binary}, \quad \forall \in \forall \in M.$$

In hybrid formulation, the first set of constraints (٧) is used to linearized the expression $w_j.max$ (٠, $c_j - d_j$) and this allows to calculate the objective function based on the position of jobs. The second (٨) and third (٩) set of constraints impose position sequence for jobs, the fourth set (١٠) indicate that starting time of consecutive position of jobs must have delay with duration of processing time of the previously positioned job, and eventually last set (١١) impose operational constraints of each job.

٣ Constraint Programming Formulation

It is axiomatic that intrinsic difficulties of exact methods or algorithmic problems in the context of combinatorial optimization often render large number of scheduling problems as NP-hard. From the complexity viewpoint, representation of the solution space in a specific algorithms plays an important role in dealing with such NP class instances. Designed solution methods, even high-tech ones, which are

highly dependent on number of variables and solution space, might yield the optimum in exponential function of time in practice. In order to deal with such drawbacks of conventional mathematical models, we tried and developed a constraint programming (CP) model.

CP has been proven very efficient for solving scheduling problems (IBM Ilog CP, ٢٠٠٩). This approach is highly applicable in two cases: ١) A non-convex solution space that comprised myriad of locally optimal solutions, ٢) Multiple disjunctions or globally-defined constraints, which results in poor information retrieved by a linear relaxation of the problem.

A pure disjunctive programming with interval variables s_{ij} (starting time of job i on machine j in a directed graph) is represented as follows:

Objective: $\text{Min} \sum_{j \in N} Z_j$ (١٢)

Constraints:

$w_j (s_{ij} + \quad - \quad) \leq Z_j$, $\forall \in \quad \forall \in M$, (١٣)

$s_{ik} - \quad \geq p_{ij}$, $\forall \in \quad \forall \quad \in \quad \quad < \quad > \rightarrow < \quad >$, (١٤)

$(s \quad - \quad \vee \quad - \quad \geq p \quad \vee \quad$, $\forall \quad \in \quad \forall \in M$, (١٥)

$s_{ij}, Z_j \geq ٠$, $\forall \in \quad \forall \in M$. (١٦)

In this formulation, the first set of constraints (١٣) is considered for linearization of the objective function. The second set (١٤) ensure that operation (*i,k*) cannot start before the completion of operation (*i,j*). The third set of constraints (١٥) observe ordering among operations of different jobs that have to be assigned on the same resource.

The CP model, in contrast to IP model, is highly contingent upon the CP package used for modeling the problem because of the discrepancies in functional structures of various modeling languages (Edis & Ozkarahan, ٢٠١١). In this study, ILOG's OPL Studio 11.1 is used as the modeling language. Also, Table ١ presents the syntax definition of the constraints used in OPL model.

Table ١ Syntax definition for declared OPL local or global constraints

Syntax	*Definition*
cum	Accumulation over interval variables
precedes	Declares precedence constraint on interval variables
requires	Declares resource requirement
wait	Declares precedence constraint on the assignment of resource variables

A basic OPL modeling framework involves a set of intervals (jobs) that need to be assigned on a set of resources (e.g. machines, operators, etc.). An interval variable is a decision variable with three components, i.e. a start and end time and a

duration, logically linked together. For interval variables, the search methodology does not enumerate the values in the variables domain and the search space is usually independent of the problem domain size. The search space is estimated as the number of possible orderings of the *n* interval variables and the complexity of problem is estimated as *n.log۲(n)*.

Then, the OPL formulation can be written as follows:

Objective: Min $cum_{i \in N}\{ \quad , \quad max(C[i]-d[i],0)\}$ (۱۷)

Constraints:

$C[i] \quad \_ \quad OPR[i,nb\_resource].end, \quad \forall \in N,$ (۱۸)

$OPR[i,nb\_resource]$ **precedes** $C[i], \quad \forall \in N,$ (۱۹)

$OPR\ i\ j$ **precedes** $OPR\ i\ j+1], \quad \forall \in \quad \forall, \in \quad -n\},$ (۲۰)

$OPR\ i\ j \quad MAC\ OPR\ i\ j \quad j], \quad \forall \in \quad \forall, \in M,$ (۲۱)

$MAC\ OPR\ i\ j \quad j \quad MAC\ OPR\ i\ j \quad j+1],$ (۲۲)

$\forall \in \quad \forall, \in \quad -n\},$

In this OPL formulation, we deployed two key variables, namely *OPR* as interval variable and *MAC* as resource variable or machines. The objective allows the solver to accumulate over sequences of interval variables and to select the minimum value corresponding to that combination. The first set of constraints (۱۹) declares that jobs on their last operations cannot override *C*[*i*] interval. This auxiliary variable helps the modeler to impose a global deadline constraint for the entire planning horizon. The constraints set (۲۰) adjusts the precedence conception, and (۲۱) indicates all interval variables should be dedicated to *unary* resource variable (*MAC*). Note that there is no *alternative* resource for interval variables, therefore they must be dedicated to only one source. The last set of constraints (۲۲) suggests that resource assignment should not be overlapped.

۴ Test Problems and Experiments

In order to evaluate the capabilities of proposed models, it should be tested on difficult problem instances. Also, our attempt should be focused on conditions under which the solution procedure is most severely challenged. Moreover, the ideal condition for conducting such experiment could exist in usage of data which are representative of real scheduling problems, but providing such pure data is almost impossible.

We obtained our job shop problem benchmarks among well-known test problems in the literature, e.g. ABZ۰۵, ABZ۰۶, MT۱۰, etc. This benchmarks initially intended for makespan case are revised by adding a due date and a weight for each

of jobs. The latter are constructed following scheme of Pinedo & Singer (۱۹۹۹): They suggested that ۲۰% of the customers are very important, ۶۰% of them are of average importance, and the remaining ۲۰% are of less importance. Therefore, we assume that w_j = (۴,۴,۲,۲,۲,۲,۲,۱,۱) for a typical ۱۰×۱۰ (job×machine) instance. According to following equation, the due date of job j is set to be equal to the floor of the sum of the processing times of its operations multiplied by a due date tightness factor β.

$$d = \left\lfloor \sum \right\rfloor \qquad \beta = 1.3, 1.5, 1.6.$$

۴.۱ *Computational Results and Analysis*

Table displays the size of the MIP and CP formulations for all three alternatives. As the table indicates, the CP formulation contains the smallest number of constraints and variables in comparison with the other formulations. The hybrid formulation (HB) has the highest number of variables. It can be immediately inferred that the last set of constraints possess high portion of constraints in this model, but it is obviously not a tight enough set to take advantage of this property. The disjunction formulation (DJ) has a moderate number of constraints by a wide margin, which seems to account for its efficiency to solve larger problems. In spite of the fact that the HB formulation is conceived to be very tight, the size of the formulation finally contributes to its longer computation times. Table ۲ signals the fact that number of binary variables in DJ and HB formulation almost tend to grow exponentially by the rise of problem size. Moreover, solving the CP model is roughly analogous to solving a pure integer model, because solution procedures for facing with CP models only have been built and designed on the integer base.

Table ۲ Model comparison based on the number of generated variables and constraints

	Variables											
	Binary			Integer or interval			Non-negative			Constraints		
Problem size (job × machine)	DJ	HB	CP	DJ	HB	CP	DJ	HB	CP	DJ	HB	CP
۴×۴	۲۴	۶۴	۰	۰	۰	۲۰	۲۱	۴۰	۰	۶۴	۲۵۴	۱۶
۶×۶	۹۰	۲۱۶	۰	۰	۰	۳۶	۴۳	۸۹	۰	۲۱۶	۱۳۳۰	۳۶
۸×۸	۲۳۴	۵۱۲	۰	۰	۰	۷۲	۷۳	۱۳۳	۰	۵۱۲	۳۸۳۲	۶۴
۱۰×۱۰	۴۵۰	۱۰۰۰	۰	۰	۰	۱۱۰	۱۱	۱۹۲	۰	۱۰۰۰	۹۳۹۰	۱۰۰
۱۲×۱۲	۷۹۲	۱۷۲۸	۰	۰	۰	۱۵۶	۱۵۷	۲۹۱	۰	۱۷۲۸	۱۹۵۷۲	۱۴۴
۱۵×۱۵	۱۵۷۵	۳۳۷۵	۰	۰	۰	۲۴۰	۲۴۱	۴۵۳	۰	۳۳۷۵	۴۸۱۳۵	۲۲۵

Our experiments involved solving each test problem using ILog CPLEX ۱۱.۱ on a ۲.۶۶ GHZ Intel Core ۲ Quad Processor (۴ MB Cache) with ۶ GB of memory. We restricted the solver with a ۳۶۰۰-s time limit in order to terminate a specific run if the optimal solution had not been verified in that period of time. The runs information are summarized in Table ۲. For each formulation, the table displays the number of problems solved within the time limit for the full set of ۳۰ problems, the average time (in second), and the maximum time. We also kept track of the maximum and average gap for the cases in which an optimal solution was not found by the solver and for better addressing the convergence of the model. (A gap is the difference between the solution returned by truncated run and the optimum, computed as a ratio to the optimum.) The results indicate that the HB formulation is very weak, it is also inefficient compared to other formulations. The CP formulation solves most of the benchmarks. Also, we found that DJ formulation converges as fast as CP to the optimum in some cases, but it takes so much time to prone and explore the branches and prove optimality. As the table displays, tightness factor has dramatic impact on average and maximum time to solve a problem, because it restricts the search space and triggers fast convergence. There remains no skepticism about the performance of CP: it outperforms other formulations in all measures.

Table ۲ Summary of computational results

Tightness factor (β)	Model	Problem solved	Average time (sec)	Maximum time (sec)	Average gap (%)	Maximum gap (%)
۱.۲	DJ	۲۷	۹۸۵.۱۳	۳۶۰۰+	۱۱.۱۱	۳۰.۰۰
۱.۲	HB	۱۱	۲۵۴۹.۶۷	۳۶۰۰+	۴۶.۲۵	۱۰۰.۰۰
۱.۲	CP	۳۰	۷۴۸.۰۲	۱۹۸۵.۲۲	۰.۰۰	۰.۰۰
۱.۵	DJ	۲۱	۲۹۱۴.۶۵	۳۶۰۰+	۷۲.۱۲	۹۷.۰۱
۱.۵	HB	۵	۳۶۰۰+	۳۶۰۰+	۱۰۰.۰۰	۱۰۰.۰۰
۱.۵	CP	۲۲	۲۷۷۱.۳۶	۳۶۰۰+	۲۵.۹۷	۴۳.۱۲
۱.۶	DJ	۱۷	۳۶۰۰+	۳۶۰۰+	۱۰۰.۰۰	۱۰۰.۰۰
۱.۶	HB	۲	۳۶۰۰+	۳۶۰۰+	۱۰۰.۰۰	۱۰۰.۰۰
۱.۶	CP	۲۲	۲۷۸۹.۲۷	۳۶۰۰+	۶۵.۷۴	۱۰۰.۰۰

Also, in order to intuitively validate the proposed models including MIP and CP, we used a graphical chart similar to Gant chat (Fig. ۱).

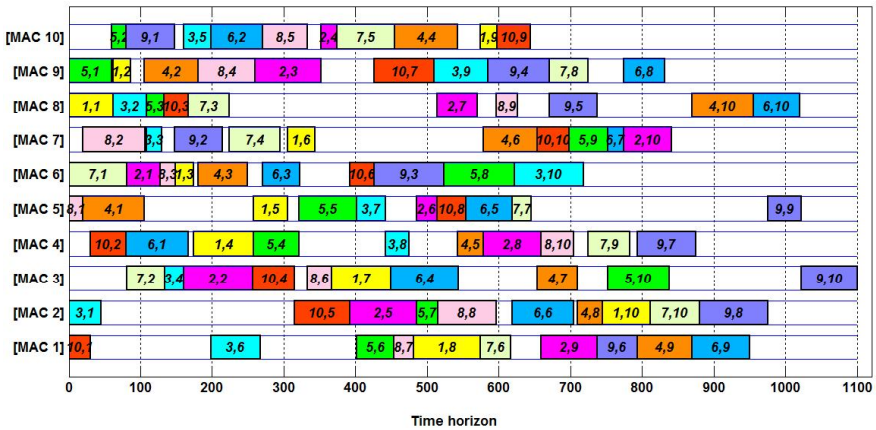

Fig. 1 Gant chart for ABZ9 instance optimal schedule; obtained by proposed CP (First and second indexes respectively indicate the job's number and operation's number for processing that job.)

5 Conclusions

That a scheduling problem can be formulated through mathematical programming does not insinuate that there is an available satisfactory standard solution procedure. TWT-JS is a very hard problem either based on enumeration or on heuristics.

We conducted some computational experiments based on two integer model and one OPL model, namely CP, using state-of-the-art software package, CPLEX11.1. We tested the efficiency of our proposed formulations by well-known job shop 10×10 benchmarks each with different computational complexity. The CP formulation solves nearly all difficult problems in a reasonable amount of time, because search space is usually independent of the problem domain size. Also, we found that formulation based on disjunction concept (DJ) provides a highly consistent perspective. It produces least number of variables and constraints in comparison with HB formulation and yields a tight model. When it fails to prove optimality after an hour of computation time, it preserves optimality gap.

For scheduling researchers, the implication is that standard optimization approaches are able to solve moderate-sized scheduling problems in a reasonable amount of time. Specialized algorithms would seem to be preferable only for larger problems in size. Furthermore, the attention paid to CP formulation may be justified if they provide insight into combinatorial perspective, but CP model are desirable when mathematical models come to fail in description of highly complex search spaces.

۶ References

Edis, E. B. & Ozkarahan, I., ۲۰۱۱. A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. Engineering Optimization, ۴۳(۲), p. ۱۳۵–۱۵۷.

Essafi, I., Mati, Y. & Dauzère-Pérès, S., 2008. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers & Operations Research, Volume ۳۵, p. ۲۵۹۹–۲۶۱۶.

IBM Ilog CP, ۲۰۰۹. Detailed Scheduling in IBM ILOG CPLEX Optimization Studio with IBM ILOG CPLEX CP Optimizer. [Online] Available at: http://cpoptimizer.ilog.com

Mati, Y., Dauzère-Pérès, S. & Lahlou, C., 2011. A general approach for optimizing regular criteria in the job-shop scheduling problem. European Journal of Operational Research, Volume ۲۱۲, p. ۳۳–۴۲.

Pinedo, M. & Singer, M., ۱۹۹۹. A Shifting Bottleneck Heuristic for Minimizing the Total Weighted Tardiness in a Job Shop. Naval Research Logistics, Volume ۴۶, pp. ۱-۱۷.

Singer, M. & Pinedo, M., ۱۹۹۷. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. IIE Transactions, ۳۰(۲), p. ۱۰۹–۱۱۸.