SOLVING PARALLEL MACHINE SCHEDULING PROBLEMS WITH TIME WINDOWS USING CONSTRIANT PROGRAMMING AND TABU SEARCH

Renjie He

System Engineering Research Institute
National University of Defense Technology,
Changsha, Hunan, 410073, P.R. China
He_rj@msn.com

Yuejin Tan

System Engineering Research Institute
National University of Defense Technology,
Changsha, Hunan, 410073, P.R. China
yjtan@nudt.edu.cn

Abstract

Parallel Machine Scheduling Problem with Time Windows (PMSPTW) can be described as scheduling N jobs on M nonhomogeneous parallel machines in order to minimize the penalty costs of jobs unscheduled. The release times, due dates, process times, weights, and sequence-dependent set-up times of the jobs are assumed to exist. An additional feature of the problem is that there are time windows constraints between jobs and machines, and each job must be processed within its time windows. There are few literatures about the PMSPTW at present. To solve the problem, we introduce a Constraint Programming model for PMSPTW and a system for integrating Constraint Programming and tabu search techniques. Typically, Constraint Programming (CP) uses the technique of depth-first branch and bound as the method of solving optimization problems. Although this method can give the optimal solution, for large problems, the time needed to find the optimal can be prohibitive. To overcome this problem, we only use the CP system to check the validity of solutions and determine the values of constrained variables, not to search for solutions. The search is performed by a tabu search procedure to avoid being trapped in local minima. When the procedure needs to check the validity of a potential solution, it is handed to the CP system. For the structure of the tabu search procedure, we discuss the initial solution, tabu list, move operators, and neighborhood searching strategy. In addition to describing basic features of the algorithm, experimental results for random generated benchmark test problems are also presented.

Keywords: *Constraint Programming, Tabu Search, Parallel Machine Scheduling, Time Window*

1. Introduction

Parallel Machine Scheduling Problem has been widely studied in the literature [1,2,7,8]. Here, as for the problem constraints, regular ones (i.e. constraints of jobs' release times, due dates, process times, etc.) have been considered in most of the researches. In these researches, each job can be processed on each of the machines at any time between its release and due date, and the process time for each job is identical for each machine. There are, however, many applications in which each job can only be processed on a subset of machines within several given time windows, and besides, the process time and time windows are machine dependent. The additive constraints make the problem more attractive and complex.

Constraint Programming (CP) is a paradigm for representing and solving a wide variety of problems. Problems are expressed in terms of variables, domains for those variables and constraints between the variables. The problems are then solved using complete techniques such as depth-first search (for satisfaction) and branch and bound (for optimization).

The richness of the language used to express problems in CP makes it an ideal candidate for PMSPTW. However, the PMSPTW is an NP-hard problem, and for problems of practical size computing optimal solutions can be too time consuming. In this paper, we extended the idea of Baker [2] and Rojanasoonthon [3], and describe the use of local search coupled with tabu search meta-heuristic [4,5] as the search technique within a Constraint Programming framework (ILOG Solver[6]) and its application to the Parallel Machine Scheduling Problem with Time Windows (PMSPTW).

The paper is organized as follows. h Section 2 problem description of PMSPTW is given. In Section 3 we describe the Constraint Programming paradigm, how it can be applied to the PMSPTW, and how tabu search can be performed within this framework. In Section 4 the solution technique using tabu search meta-heuristic is discussed. In Section 5, computational test is reported.

2. Problem Description

Parallel Machine Scheduling problem can be described as scheduling N jobs on M unrelated machines. For each job j, it can only be processed on a subset $M_j \subseteq M$ of machines. The processing time of job j on machine $k \in M_j$ is $p_{j,k}$, respectively. We also assume there are time windows constraints between job j and machine k, and job j can only be processed on machine k within one time window without preemption. In order to process job j after job i on machine k, a set up time $s_{i,j,k}$ is required, which depends not only on the machine the two jobs are processed on, but also on the sequence of the two jobs. Jobs have a weight associated with them, which represent the penalty cost of not scheduling the job. An optimal schedule has the following properties:

- each job is processed within its time windows, otherwise, it is considered unscheduled;
- each job is processed exactly once without interruption on one of its permissible machines;
- each machine process at most one job at a time;
- the total costs of the scheduled is minimized.

3. Constraint-Based Modeling of PMSPTW

In our model, we refer the set of jobs as J, machines as M, starting and ending jobs of the sequence on all machine as a dummy job 0. $J^0 = J \cup \{0\}$ is all jobs. M_j is the set of machines on which job j can be processed on.

3.1 Path Constraint

To model the PMSPTW, it is desirable to have a special constraint type for dealing with accumulation of time along schedules. For this, we use a path constraint. the constraint is of the form:

$$N_j = i \Rightarrow t_i \geq t_j + q_{j,i} \qquad (1)$$

where N_j is a flow decision variable representing the next job of job j performed by the same machine, t_j is a time decision variable representing the start time of job j, $q_{j,i}$ is a travel time representing the process time of job j and the setup time between job j and i.

3.2 Time Window Constraint

Modeling time windows constraint is an easy job in ILOG Solver. Suppose job j can only be processed within time window (10, 15). This can be express by the instruction $10 \leq t_j \leq 15$. Note that not all time windows consist of a continuous block time. there are situations where the acceptable time window occur in pieces. To model the disjoint time windows, we use && (and) and || (or) to formulate a composite constraint. For example, job j can start after 8 and before 10, or after 14 and before 16. this can be modeled as a composite constraint:

$$(8 \leq t_j) \,\&\&\, (t_j \leq 10 || 14 \leq t_j) \,\&\&\, (t_j \leq 16)$$

For a given job, the time windows are different from machine to machine. We have tried several models to circumvent this difficulty. The most efficient seems to use predefined IloIfThen constraint class. An instance of

IloIfThen represents a condition constraint. Generally, a condition constraint is composed of an if part (the conditional statement or left side) and a then part (the consequence or right side). Using IloIfThen, we can easily define specific time windows according to the machine the job is processed on.

3.3 Search Strategy

Solutions to CP problems are usually found using complete methods such as depth-first search and branch and bound. However, for scheduling problems of practical size, complete search methods cannot produce solutions in a short and reliable time period. By contrast, local search methods have proved successful in this regard. Local search operates by changing small parts of the solution, for instance moving a job from one machine to another. This type of operation involves retracting previous decisions and making new ones. In contrast to depth-first search, retraction can be done in any order, not simply in the opposite order the decisions were made.

To solve this problem, we only use the CP system to check the validity of solutions and determine the values of constrained variables, not to search for solutions. The search is performed by a tabu search procedure. When the procedure needs to check the validity of a potential solution, it is handed to the CP system.

4. Search Procedure

We use tabu search to control the search process. Tabu search is a special meta-heuristic that has been widely applied combinational optimization problem. It is an iterative process in which the best overall solution is kept as the final result. The computation is performed in two phase. The first phase involves the sequential construction of feasible solution one element at a time. At each iteration, all feasible elements are ranked according to an importance function that take into account the present state, and one is selected from a candidate list. In the second phase, a tabu search process is performed. It should be noted that any implementation of tabu search is problem -oriented and needs particular definitions of structural elements such as move, neighborhood, memory structure, aspiration function, the neighborhood searching strategy, stopping rule, etc. In next section, we will discuss these aspects in detail.

4.1 Phase 1

Starting with an empty schedule, jobs are individually selected and inserted into one possible position in the sequence on 1 of m machines. The process is repeated until no more jobs can be inserted. The overall process is described as follows.

Step1: start with empty schedule;

Step2: Let L be the list of unscheduled jobs;

Step3: take the most important job j from L;

Step4: insert j in a sequence on 1 of m machines, if there is no feasible position, the job is unscheduled;

Step5: remove job j from list L;

Step6: if L is not empty, go to step3, else the process terminated.

4.2 Phase 2

Phase 2 is a tabu search procedure. The main features of the procedure are presented as follows:

- Move operators

In combinatorial optimization problems, local optimality is defined in terms of neighborhoods. In this paper, we have considered 4 move operators (Figure 1).

1) Relocate a job from its position to another position on the same or different machine.
2) Interchange 2 job positions on the same or different machine.
3) Insert a unscheduled job into the schedule of 1 of m machines.
4) Delete a job from machines schedules.

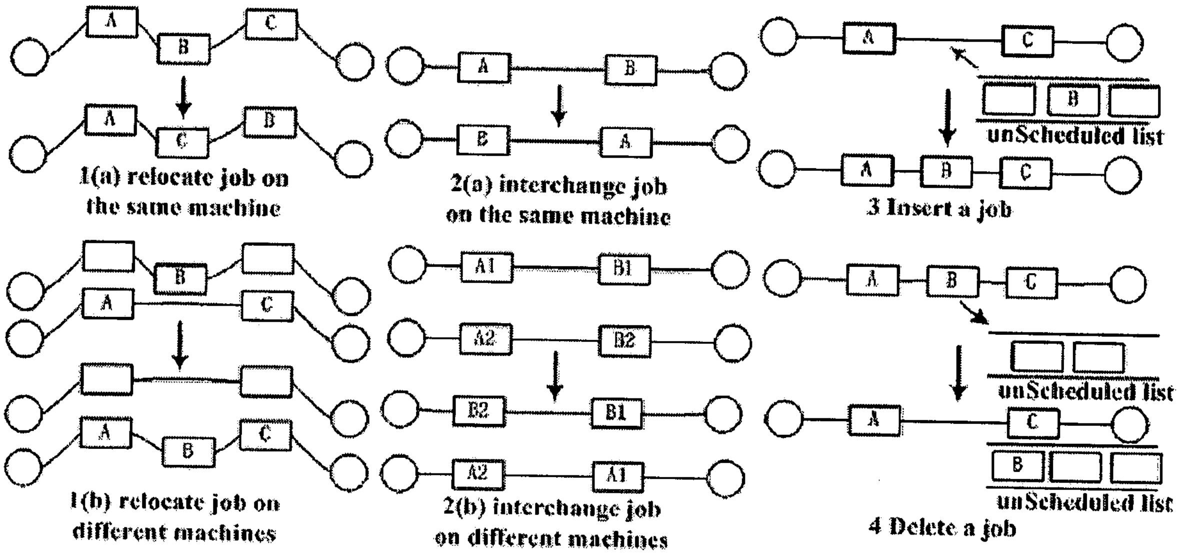
1(a) relocate job on the same machine
2(a) interchange job on the same machine
3 Insert a job
unScheduled list
1(b) relocate job on different machines
2(b) interchange job on different machines
4 Delete a job
unScheduled list
unScheduled list

Figure 1 Four class move operators

- Tabu list

For the PMSPTW, moves can be characterized by two sets of directed arcs: arcs added and arcs removed (for insert and delete move, we can treat them as a special arc of machine-job pair). For example, the relocate operator (Figure 1 (1b)) removes three arcs and adds three new arcs. We use two tabu lists of fixed and equal size: one for storing the most recently added arcs, and the other for storing the most recently removed arcs. The tabu status of a move m is decided as follows: The arcs that are removed by m and occur in the " recently added" tabu list are counted. Likewise, the arcs that are added by m and occur in the " recently deleted" tabu list are counted. The sum of these is then compared to a threshold dependent on the move type. If the threshold has been met or passed, the move is tabu, and is not tabu otherwise.

- Aspiration rule

An aspiration criterion was used which could change the tabu status of a move. We use the common best accept rule: if the move would result in a solution better than the current best, then it was accepted even if tabu.

- Neighborhood searching strategy

We classify the moves into three categories: improve moves (move operator 3), equal moves (move operator 1,2), decline moves(move operator 4). It is quite natural that a move to be performed should be selected from improve moves. Usually among them, the move yielding minimal cost value is the one performed. If there are no improve moves available, we random select a non-forbidden move from equal moves and decline moves. When he problem size grows huge, to evaluate all moves is also a time-consuming job,. To overcome this problem, a random sampling neighborhood procedure is applied, only the random selected subsets of moves are evaluated during the search process.

5. Computational Results

The algorithm is implemented in C++ on a P4 1.5G, 256RAM personal computer. Two factors are considered: number of jobs, number of machines. For a given job, the permitted machines on which the job can be processed on is random selected, the process times are drawn from uniform distribution on (10, 50). The number of time windows between jobs and machines are drawn from uniform distribution (1, 5). The weights of jobs are drawn from uniform distribution (1, 5). The schedule time horizon is (0,200).

The factors and their levels are as follows:

n : 100, 200, 400

m : 2, 4, 6, 8, 10

Ten replications were conducted for each problem configuration and a total of 150 problem instances were examined. The computational results for the experiment are included in Table 1. Phase 2 runs are limited to 200 iterations.

From Table 1, we can observe that:

- Using constraint programming and tabu search, we can solve the PMSPTW efficiently and
- Using random sampling method, we can get the same good result within much shorter time.

Table 1 Computational results

n	*m*	*Phase 1*	*Phase 2 (all moves)*	*CPU(s)*	*Phase 2 (random sample)*	*CPU(s)*
100	2	214/87	173/78	142	180/79	20
	4	166/69	109/59	186	114/55	57
	6	109/51	53/39	223	51/31	69
	8	117/41	26/16	186	35/18	72
	10	69/30	18/13	180	23/13	72
200	2	462/182	407/172	366	413/173	75
	4	431/172	343/155	489	345/150	133
	6	386/153	264/130	675	269/129	194
	8	349/140	198/110	776	197/109	238
	10	305/122	166/93	896	168/95	280
400	2	957/385	892/375	685	898/372	133
	4	895/364	770/344	1278	780/347	263
	6	871/350	687/319	1771	695/318	325
	8	853/336	659/305	1938	661/302	309
	10	880/360	735/338	1443	743/337	284

6. Conclusions

The methodology presented in this paper represents an attempt to solve a variant of the nonhomogeneous parallel machine scheduling problem with time windows using constraint programming and tabu search. The same approach would still work for more complex variations. As Constraint Programming naturally separates the description of the problem from the way it is solved, it is also possible to implement more sophisticated solving techniques using search engines based on local search and meta-heuristics.

References

[1] Cheng, T. C. E., Sin, C. C. S, "A state-of-the-art review of parallel-machine scheduling research", *European Journal of Operational Research*, Vol 47, p271-292, 1990.

[2] S.Rojanasoonthon, JF Bard, SD Reddy., "Algorithms fro parallel machine scheduling: a case study of the tracking and data delay satellite, system", *Journal of Operational Research Society*, Vol. 54, p 806-821, 2003.

[3] Bruno De Backer, Vincent Furnon, Paul Shaw., "Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics", *Journal of Heuristics*, Vol. 6, p501-523, 2000.

[4] Glover,F, "Tabu Search, Part I", *ORSA Journal on Computing*, Vol. 1, No.3, p190-206. 1989.

[5] Glover,F, "Tabu Search, Part I", *ORSA Journal on Computing*, Vol. 2, No.1, p4-32. 1990.

[6] ILOG Inc. ILOG Solver 5.3 User's manual. 2003

[7] Virginie Gabrel, "Scheduling jobs within time windows on identical parallel machines: New model and algorithms", *European Journal of Operational Research*, Vol. 83, p320-329, 1995.

[8] Zhi-Long Chen, Chung Yee Lee, "Parallel machine scheduling within a common due window", *European Journal of Operational Research*, Vol. 136, p512-527, 2002.