TABU SEARCH FOR THE MULTI-MODE RESOURCE CONSTARINED PROJECT SCHEDULING PROBLEM WHITH RESOURCE FLEXIBILITY

Y Kadrou, N M Najid

TABU SEARCH FOR THE MULTI-MODE RESOURCE CONSTARINED PROJECT SCHEDULING PROBLEM WHITH RESOURCE FLEXIBILITY

Y. KADROU1, N.M. NAJID2

[1] IRCCyN, Université de Nantes, 1 rue de la Noë, BP 92101, Nantes F-44000 France

[2] IUT de Nantes, Dept. GMP, 2 avenue du pr. Jean Rouxel 44475 Carquefou Cedex

Abstract

The scheduling problem under study may be viewed as an extension of the standard Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) including Multi-Skilled Labor and will be called as MRCPSP-MS. This problem requires an integration of resource limitation, labor skills, and multiple possible execution modes for each task, and the objective is to minimize the overall project duration. This paper present a new tabu search (TS) algorithm using a powerful neighborhood function based on a flow graph representation in order to implement various search strategies. The search of the solution space is carried out via two types of moves. Furthermore, the TS algorithm is embedded in a decomposition based heuristic (DBH) which serve to reduce the solution space. The effectiveness of the new Tabu Search is demonstrated through extensive experimentation on different standard benchmark problem instances and proves that our results are competitive.

Keywords:

Project scheduling, Heuristics, Tabu Search, Human resources.

1 INTRODUCTION

The scheduling problem under study may be viewed as an extension of the standard Resource-Constrained Project Scheduling Problem [1] and will be called as Multi-mode Resource-Constrained Scheduling Problem with Multi-Skilled Labor (MRCPSP-MS). In this problem, a project composed by a set of activities has to be performed by a limited number of labors with a certain mix of skills. A laborer may have one or more skill profiles (ex: assembler, electrician...) with a particular skill level. Activities can be performed only by workers who have the appropriate skill profile and at least the skill level needed. Hence labor versatility is directly tied to its skill profile and level. To keep the problem general, each activity is associated to a set of labor with the appropriate skill and level. Also, activities can be performed in one of several execution modes. The relationship between the processing time and the execution mode is a simple non-linear relation that decreases with increasing assigned labor intensity. Moreover, each activity can use simultaneously several work centers with different resource consumption according to the selected execution mode. Each work center has a limited capacity; i.e., the maximum number of labor it can get. The decision making process has to include the execution modes of activities, resources assignment, sequence of operations on the resources and work centers regarding both precedence relations and resource limits in order to shorten production cycle (makespan). Obviously, the problem above fits the characteristics of both Multi-mode RCPSP (see, [2], [3] and [4]) and Assignment problem [5] with resource flexibility. From computational point of view, as an extension of the standard RCPSP, this problem is clearly NP-hard in the strong sense [6]. In spite of all the ongoing research on the RCPSP, little has been carried out on the Multi-mode RCPSP with resource flexibility and skill consideration, not even the simpler jobshop scheduling problem with resource flexibility [5]. The flexible jobshop problem (FJSP) is an extension of the classical jobshop scheduling problem which allows an operation to be performed by one machine out of a set. The problem is to assign each operation to a machine (routing problem) and to sort the operations on the machines (sequencing problem). Being an extension of the standard jobshop scheduling, this problem is clearly NP-hard. Dauzère Pérès et al. [7] were the first to present a local search algorithm to solve multi-resource shop scheduling with resource flexibility (or MJSPF for short). This problem is an extension of the flexible jobshop problem which allows an activity to utilize several kinds of resources simultaneously, each of which is selected in a given set. Dauzère Pérès et al. [8] also proposed an extension of MJSPF which allows a resource (often human) to be released before the end of the operation and prevents a set of incompatible resources from being selected. The case of resource allocation has been dealt with in the literature in a number of papers. In particular, a resource allocation in multimodal activity networks is examined by Tereso et al. by applying a Dynamic Programming [9] and Electromagnetism Algorithm approach [10]. In their model, the resources are disjunctive and activities can be processed in different modes where the activity duration depends in its work content and on the amount of resource allocated. The resources required to complete the project are supposed completely polyvalent.

Although our problem can be viewed as an RCPSP, it is actually more complex. Clearly, the introduction of flexibility considerations and skill constraints complicates the already difficult Multi-mode RCPSP. However, to our knowledge, the literature does not contain any dedicated exact or heuristic procedures for solving the MRCPSP-MS as defined in this paper. The remainder of this article is organized as follows: Section 2 clarifies the terminology and the project representation used below. In Section 3, a lower bound of the problem is presented. We then give a detailed description of our TS algorithm is Section 4. In Section 5 we evaluate the effectiveness of the algorithms by applying the compiler to a number of standard benchmarks. Finally, Section 6 presents our conclusions and suggests several directions for future research.

2 PROBLRM DESCRIPTION AND FLOW NETWORK REPRESENTATION

The MRCPSP-MS can be stated as follows. Assume a project defined by an acyclic directed graph *G: = (A, E)*, with vertex set *A (0, 1, 2,…, n, n+1)* corresponds to the set

of non-preemptable activities to be scheduled where the dummy activities *0* and *n+1* mark the beginning and the end of the project; respectively. The set of arcs *E: = {(i, j): i, j∈A, i ≠ j}* represents precedence constraints between activities. Let's A_i^- (A_i^+) denote; respectively, the set of direct predecessor (successor) of activity *i* (denote the size of A_i by $|A_i|$). Specifically, an activity can only start after all its predecessor activities have been completed. To be processed, the activities require a set of labors $\mathcal{K} = \{1,\ldots,K\}$ and a set of work centers $\mathcal{W} = \{1,\ldots,W\}$. Each work center $w \in \mathcal{W}$ has a limited capacity. Let's Z_w denotes the work center capacity which is maximum number of labors the work center *w* can receive to process a subset of activities simultaneously. Various resources such as machines, tooling, fixtures, with similar processing characteristics are grouped together in a work center and labors also must be available to perform activities. Each labor $k \in \mathcal{K}$ can perform a pool of activities and cannot be assigned to more than one activity at the same time. Lets $A_k \subseteq A$ denotes the set of activities that labor *k* can process and let $\mathcal{K}_i$ denotes the set of labors (labors with the required skill profile and level) able to perform activity *i*. For each activity $i \in A$ a set $\mathcal{M}_i = \{1,\ldots,M_i\}$ of (execution) modes is available. Each activity has to be performed in exactly one mode. The processing time of activity *i* being executed in mode *m* is denoted by $d_{i,m}$. Thus, when processed in mode $m \in \mathcal{M}_i$, activity *i* requires the reservation of $z_{i,m}^w$ units of work center $w \in \mathcal{W}$ and the allocation of $a_{i,w}^m$ workers among the set of labor $\mathcal{K}_i$ during $d_{i,m}$ units of time. We assume that the amount of work center units engaged to process activity *i* is equal to the number of allocated workers; i.e., $z_{i,m}^w = a_{i,m}^w$. One immediately deduces that the total number of assigned workers denoted $a_{i,m}$ for the execute activity *i* in mode *m* is given by

$$a_{i,m} = \sum_{w \in \mathcal{W}} a_{i,m}^w = \sum_{w \in \mathcal{W}} z_{i,m}^w \qquad (1)$$

The processing time value is based on a speed rate α_m that is mode dependent, but independent of the resource allocation. The time to process an activity *i* in mode *m* is the multiplication of the processing time in mode 1 by the speed rate of mode *m*, $\alpha_m \in]0, 1]$. In what follows, we set the speed rate such as for two different modes $m' > m''$, we have $\alpha_{m'} < \alpha_{m''}$. For the sake of simplicity an execution mode *m* of activity *i* is defined by $d_{i,m} \left| < z_{i,m}^1, \ldots z_{i,m}^H > \right.$ where $d_{i,m}$ its duration and $< z_{i,m}^1, \ldots z_{i,m}^H >$ the work centers units required for its execution.

Table 1: An example of a project scheduling problem

Activity	*Mode1*	*Mode2*	*Mode3*	*Labor*
1	15 |<0,1>	13|<0,3>		1, 3, 5
2	10 | <1,1 >	7 | <1,2 >	6 | <2,3 >	1,2, 3, 4, 5
3	10 | <1,1 >	7 | <2,1 >	5 | <2,2 >	1, 2, 4, 5
4	7 | <1,1 >	5 | <1,2 >	2 | <2,3 >	1, 2, 4, 5
5	5 | <1,1 >	4 | <1,2 >		2, 3, 5
6	13 | <1,1 >	6 | <1,2 >		1, 2, 3, 4
Work center capacity : (1, 2); (2, 3)				

We shall assume in this paper that no preemption and no overlapping between two consecutive activities, setup times are negligible or are included in the fixed duration and no ready times or due dates are imposed on any of the project activities. Consider as example the project given in table 1, composed by six activities, five laborers and two work centers which can receive individually at most two and three laborers; respectively. Let us consider the case in which activity 3 is performed in mode 2 described by the vertex 7|<2, 1>: The corresponding processing time d_{12} equals seven units of time. Two units of work center 1, $z_{3,2}^1 = 2$ and one unit of work center 2, $z_{3,2}^2 = 1$, are used at each point in time at which activity 3 is being executed. Furthermore, two workers and one worker must be assigned to work center 1 and 2, respectively. These laborers have to be selected from the set of labors $\mathcal{K}_1$ = {1, 2, 4, 5} able to perform activity 3. The acyclic directed graph *G (A, E)* shown in Figure 1 (a) gives the precedence relation between activities. The duration used in the CPM graph correspond to the duration when the shortest mode, $m = \mathcal{M}_i$, is used. The length of the longest weighted path from 0 to any operation is referred to as the head of the operation $r_{i,m}$ and equals the earliest start time of *i*. The tail $q_{i,m}$ of the operation *i* is the length of the longest path from *i* to *n+1* minus the processing time of *i* ($d_{i,m}$). The head and the tail of each activity can be computed easily using backward and forward recursion on the graph. Obviously, the C_{max} when relaxing resources constraints (labors, work center) represents a simple lower bound on the project completion time (length of the critical path). Figure 1 (b) gives an optimal solution of our scheduling problem example computed using Xpress-MP with a makespan equal to 21 time units. The Gantt chart of Figure 1 (b) displays activities and labor assignment over time. Each block represents an activity on the vertical axis the number of labors assigned, where the laborers allocated to each task are indicated inside the block and the length of a block denotes the duration of the activity.

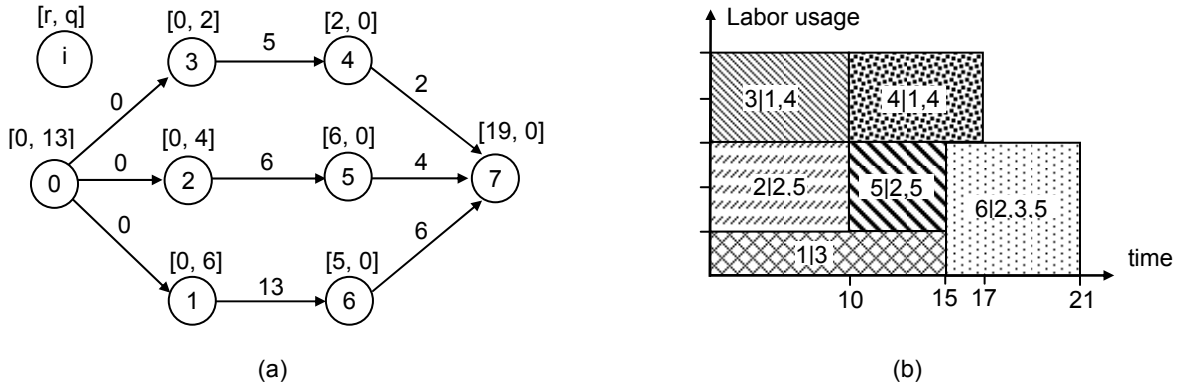

Figure 1: Project example and Gantt chart representation

Every instance of MRCPSP-MS can be represented by the flow graph *G (A, E, K, W)*. The graph *G* can be decomposed into three sub-graphs G_A, G_E, G_W where $G_E(A, E)$ represents the set of conjunctive (directed) arcs connecting operations subject to precedence constraints. $G_K(A, K)$ is the set of conjunctive arcs connecting operations to be processed by the same laborers and $G_W(A, W)$ the flow network representing the way in which work center units are passed on between the various project activities. A schedule on a disjunctive graph *G(A, E ,K)* consists in finding a set of orientations that minimizes the length of the longest path (critical path) such that the resulting solution directed graph is acyclic (there are no precedence conflicts between operations). The sub-problem *G (A, E, W)*, is an extension of the known RCPSP where each activity has multiple execution modes. Since work centers are cumulative resources, it's difficult to model disjunction between activities requiring the same work center to be scheduled simultaneously. This difficulty can be deal with if each work center *w* is defined as a set of identical disjunctive resources T_w (denote the size of T_w by $|T_w|$), such that $Z_w = |T_w|$ and each elementary resource *z*, $z \in T_w$ can be assigned at most one activity and the disjunctive arc $(i, j) \in W$ can be seen as an union of Z_w disjunctive arcs. The way in which work center units are passed on between the various project activities can be represented by a resource flow network. To illustrate disjunction related to conflicting demand of work centers, we use the resource flow network presented by [xx], in which rather than considering the disjunction on each resource unit of work center, the amount of work center units transferred between two activities is grouped on the same edge since all units of the same work center are equivalent same (identical parallel machines). Hence, the set of arcs $G_W(A, W)$ is constructed in such a way that the partial graph $G_W(A, W)$ is a transportation network where each arc u= $(i, j) \in G_W$ is associated with a vector capacity $\Psi_{i,j} =< f_{i,j}^1, \ldots f_{i,j}^H >$, $f_{i,j}^w$ representing the number units of work center $w \in \mathcal{W}$ that are directly transferred from activity *i* (when it finishes) to activity *j* (when it starts). In order to reduce the number of transportation arcs and simplify the flow network representation, all resources transferred between two activities can be grouped on the same resource flow arc $f_{i,j} =< d_{i,m} | \Omega_{i,j} | \Psi_{i,j} >$, where $d_{i,m}$ is the duration of activity *i* when processed in mode *m*. $\Omega_{i,j}$ and $\Psi_{i,j}$ are the set of labors and work centers units transferred from *i* to *j*, respectively. Considering again our example, Figure 2 gives a feasible solution of the scheduling problem and illustrates the associated flow network based on the new model-building. The conjunctive arcs (thin arrows) of *E = {(3,4), (2,5), (1, 6)}* represent precedence constraints between activities and resource transportation arcs (bolded arrows) of $K \cup W$ correspond to the labors and work center units transferred between activities.

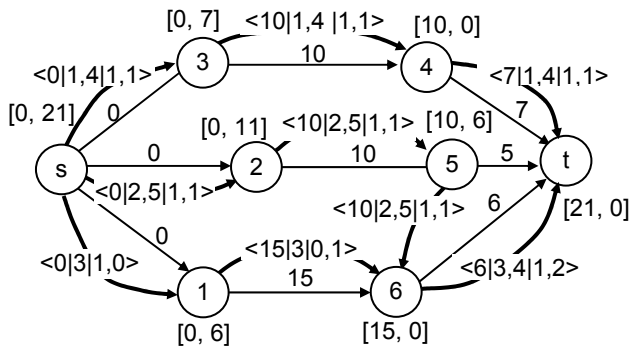

Figure 2: Flow network representation

3 LOWER BOUND

In this section, three makespan lower bounds are presented, which can be realized efficiently with small computational effort. The following three lower bounds exploit the problem structure of MRCPSP-MS.

Critical Path Bound (LB1): The project duration when relaxing resource constraints and assigning shortest modes to the activities represents a simple lower bound on the project completion time. In the previous example, LB1 is equal to 15.

Resource Capacity Bound (LB2): MRCPSP-MS can be relaxed by discarding precedence constraints skill requirement and work center capacity constraints. A bound value is computed as the total workload divided by the amount of available labor. For our example LB2 = 22.

$$LB_2 = \sum_{i \in A} \left\lceil \min_{m \in M_i} \{a_{i,m} d_{i,m}\} \Big/ K \right\rceil \quad (2)$$

Work Center Capacity Bound (LB3): For each work center w $\in$ W we compute a lower bound based on the same principle used in the computation of LB2. Let's Aw be the set of activities which have to be processed by the work center w. For our example LB3 = 15. LB3 can be expressed as follow:

$$LB_3 = \max_{w \in \mathcal{W}} \left(\sum_{i \in A_w} \left\lceil \min_{m \in M_i} (z_{i,m}^w d_{i,m}) \right\rceil / Z_w \right) \quad (3)$$

The makespan lower bound is then the maximum between these three bounds:

$$LB(C_{\max}) = MAX\,(LB_1, LB_2, LB_3) \quad (4)$$

4 TABU SEARCH ALGORITHM

Tabu search algorithm uses a neighborhood search procedure to iteratively move from a solution *x* to a solution $x' \in N(x)$ in the neighborhood of *x*, until some stopping criterion has been satisfied. At each iteration, TS examines all the solutions close to the current solution by performing a set of candidate moves as defined by the neighborhood structure. In order to escape from a local optimum and avoid cycling, moves that would bring back to a recently visited solution are kept in *tabu list*, and forbidden or declared tabu for a certain number of iterations If all the candidate moves are tabu, then the oldest element on the tabu list is removed until a permissible move can be selected. Recently, the theory and the practice of tabu search were extensively improved by Glover & Laguna [11] and by Hertz, Taillard & De Werra [12] by aspiration criteria and intensification and diversification schemes.

4.1 The Neighborhood structure

The majority of the local methods, applied to MRCPSP, the assignment and the sequencing of operations on the resources are treated separately. Either directly, i.e. assignment and sequencing are considered independently or indirectly in a local search algorithm where reassignment and resequencing are two different types of transitions. The proposed Tabu Search algorithm is distinguished mainly by two aspects. Firstly, the proposed neighborhood function is based on the flow network representation described above, where the decisions of resequencing activities and re-assignment resources and execution modes are not differentiated. Secondly, the TS algorithm uses a sophisticated set of moves for neighborhood exploration and evaluates the performance of each visited solution without making the move. Since our objective is to minimize the duration of the project, these movements are

applied only on the set of critical activities. An activity is defined to be a critical activity if and only if:

$$r_{i,m} + d_{i,m} + q_{i,m} = r_{n+1} \quad (5)$$

The neighborhood *N(x)* of a visited solution *x* is explored by applying one of the following two types of moves:

Change_Alloc(i): Change the resource allocation of the activity *i* by replacing the critical incoming and outgoing resource arcs of activity *i* by another eligible resource arcs. Such reinsertion has to be performed only by making some of the resource units initially transferred from an activity *u* to an activity *v* to be transferred from *u* to *i* and then from *i* to *v*. This move is applied only on the set of critical activities verifying:

$$r_{i,m} > \max_{p \in A^-}(r_p + d_p) \text{ or } q_{i,m} > \max_{s \in A^+}(q_s + d_s) \quad (6)$$

Change_mod(i, m, m'): Change the current execution mode *m* of the activity *i* by *m'*$\in \mathcal{M}_i$, satisfies *m'* ≠ *m*. Performing this movement imply removing and adding the following flow units :

$$f_{i,\sup} = \left\langle K_{\sup} \middle| W_{\sup} \right\rangle = \left\langle K_{i,m} - K_{i,m'} \middle| W_{i,m} - W_{i,m'} \right\rangle \quad (7)$$

$$f_{i,add} = \left\langle K_{add} \middle| W_{add} \right\rangle = \left\langle K_{i,m'} - K_{i,m} \middle| W_{i,m'} - W_{i,m} \right\rangle \quad (8)$$

$$f_{i,\sup} \cap f_{i,add} = \varnothing \quad (9)$$

These two flows are disjoined, because the amount of the consumed resources can only increase or decrease while passing from a mode to another. An arc *(s, t)* $\in \Gamma_i$ is feasible for the insertion of activity *i*, if the resulting graph is acyclic, and the following necessary and sufficient feasibility conditions are satisfied:

Proposition 1: An arc *(s, t)* such that $s \notin \{A_i^+, K_i^+, W_i^+\}$ and $t \notin \{A_i^-, K_i^-, W_i^-\}$ is feasible for the insertion of activity *i* if :

$$f_{s,t} \cap f_{add} \neq \{\varnothing\} \quad (10)$$

$$r_s < \underset{j \in V_i'^+}{Min}\left(r'_j + d_j\right) \quad (11)$$

$$r_t + d_t < \underset{j \in V_i'^-}{Max}\left(r_j\right) \quad (12)$$

Indeed, condition (11) checks that the flow of the arc (s, t) may satisfy at least part of the resource requirement of activity *i* while condition 11 and 12 ensure the acyclic property of the new generated flow graph *G'* after the insertion of activity *i*.

The procedures used to perform these two movements employ similar mechanisms, which substitute all the critical incoming resource arcs of the considered activity by new feasible set of arc Γ_i which minimizes or degrades less the makespan of the project. The REINSERT algorithm is as follows:

Algorithm REINSERT (i, $f_{\sup}$, f_{add})

1. Decide on which incoming and outgoing set of arcs of activity *i* the flow $f_{\sup}$ must be removed.
2. Create a new routing arcs between the predecessor and successor of activity *i* that minimize the project makespan.
3. Define the set of eligible arcs for the activity insertion
4. For each feasible insertion arc compute a lower bound of the project makespan increase.
5. Select a subset of eligible arcs that satisfy capacity constraints and flow conservation such as the resulting makespan is minimal.

For that, it would be necessary to be able to evaluate all the eligible insertion arcs. However, a complete evaluation, i.e. calculation of the starting/finishing times of all the tasks, takes a considerable time. This led us to define a lower bound which can evaluate the quality of a move without actually making it. The lower bound of the makespan after selecting the feasible arc (s, t) is defined as follows:

$$Lb(C_{\max}) = r'_i + p_{i,m'} + q'_i \quad (13)$$

$$r'_{i,m'} = \max\left(\max_{p \in V_i'^-}(r_{p,m} + p_{p,m}), r_{s,m} + p_{s,m} \right) \quad (14)$$

$$q'_i = \max\left(\max_{f \in V_i'^+}(p_{f,m} + q_{p,m}), p_{t,m} + q_{t,m} \right) \quad (15)$$

Where $V_i'^{-/+} = \left\{A_i^{-/+} \cup K_i'^{-/+} \cup W_i'^{-/+}\right\}$ the set of incoming / outgoing is arcs of activity *i* after deleting the flow $f_{\sup}$.

4.2 Starting solution

An initial solution is generated by a very simple heuristic using a serial scheduling scheme and the SLK-SFM priority rule. The serial schedule generation scheme (SSGS) consists of *g = 1… n* stages, in each of which one activity is selected and scheduled. Each stage is made up of three steps. Step (1) at each decision point the activity from the set of eligible activities (all their predecessors have been scheduled) with the smallest slack is selected, slack calculation being based on the shortest possible duration for the not-yet-scheduled activities and the selected duration for already-scheduled activities. Step (2) SFM rule chooses the mode with the shortest feasible duration and the selected activity during step 1 is scheduled at the earliest precedence and resource feasible starting time. Step (3) states that the workers must be assigned to the activities in order to balance the workload on the resources.

4.3 Tabu search mechanisms

The aspiration criterion used was that a tabu move would be accepted if it produced a solution that was better than the best solution found to date. If a move that is set tabu would lead to the better solution better than the best solution found to date, the tabu status is revoked, and the corresponding move is selected. This aspiration criterion is known as global aspiration by objective.

The *long* and *intermediate term memory* components were supplied by using frequency-based memory. Essentially, frequency-based memory stores information about the frequency that a move with a specific attribute has occurred. After a specified amount of time, the long term memory is called to ensure that the search process examines solutions throughout the entire solution space (diversification). After the same amount of time the intermediate term memory was then called to intensify (*intensification*) the search by finding a new starting point which tended to share common features solutions examined.

The diversification and Intensification can accomplished during the insertion process by selecting the arcs which lead to the more or less encountered situation. This can be done by modifying the lower bound of the feasible insertion

arcs which makes arcs containing less (diversification) or more (Intensification) frequently encountered attributes more attractive.

4.4 Restricting the solution space

It was shown by [13] that nearly 90% of the time of resolution is taken by the evaluation of the neighborhood. Consequently, it is interesting to constrain the neighborhood as possible in order to decrease the complexity of resolution. Hence, we investigate the impact of a solution space restriction by iteratively solving decomposed sub-problems of the main problem under study. A decomposition approach for RCPSP has been proposed by [14]. This approach consists of splitting each problem instance into smaller sub-problems. The search is then continued on the sub-problems, and the resulting sub-schedules are reincorporated in the schedule of the main problem. The decomposition-based heuristic (DBH) used in the TS presented above consist of making moves only on a set of activities overlapping a predetermined time interval $[t_s, t_f]$. In this way, the set of eligible moves is reduced and only activities belonging to the space interval are explored.

5 COMPUTATIONAL EXPERIMENTS

To evaluate the performance of the TS algorithm proposed in this study, we apply the algorithm to several established benchmark instances taken from literature. The problem tackled in this paper can be seen as an extension the standard flexible job-shop scheduling problem (FJSP) [15], the basic RCPSP and its multi-mode version [16, 17, 18]. The search is stopped when the number of sequences evaluated reaches a given maximal value nMax = 2000. All numerical experiments are also conducted with a unique parameter setting for all problem sets. The computational experiments have been performed on the following benchmarks:

1. Brandimarte Instances [15] : The data set of FJSP which consists of ten problems with number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15, and number of operation for each job ranging from 5 to 15.
2. Patterson and Alvarez instances : 110 “easy” Patterson instance [16] (from 6 to 51 activities) and the 48 Alvarez instances with 103 activities [17].
3. Kadrou and Najid (ND) instances [18]: The data set of MRCPSP-SL with 50, 100, 150 and 200 activities. Each instance set was divided into three subsets (100 instances per subset) differing by graph complexity, resource profile, and execution mode flexibility (easy, medium, and hard subsets), which are 1200 instance in total.

Table 2 compares the proposed TS to the best results of the tabu search of Mastrolilli and Gambardella [5] and with the different genetic algorithms proposed by Pezzella et al. [19] and Chen et al. [20]. The first column reports the instance name; the second column reports the best known lower bound. The third and fourth column report our best makespan and the average and maximum computational times required for each run (i.e., 2000 iterations). The remaining columns report the best results of the three algorithms we compared with, together with the relative deviation with respect to our TS. The relative deviation is defined as:

$$dev = \left[\left(C_{\max}^{comp} - C_{\max}^{TS}\right) / C_{\max}^{comp}\right] \times 100\% \quad (16)$$

Where $C_{\max}^{TS}$ the makespan is obtained by our algorithm, and $C_{\max}^{comp}$ is the makespan of algorithm we compare to. The values of LB within parenthesis are optimal and the makespan associated with asterisk is the best known upper bound by far. The results show that our algorithm outperforms the other two Genetic Algorithms and have comparable performance as M.G.

The results on Patterson and Alvarez instances are displayed in table 3. For each problem set, we display in the first two columns the average/maximal deviation from the optimum or from the best known upper bound, the average/maximal CPU time required and the number of improved solutions are indicated in the next two columns. We have solved to optimality all the PAT instances and found five new best solutions on ALV sets.

In table 4, we compare our algorithm with the MMSH (Multi-mode Multi-skill Heuristic) heuristic proposed by kadrou et al. [18]. The results reveal that our TS is capable to report consistently good results and outperform MMSH heuristic on all tested instances. The deviation of computational time seems quite large. This is because the size of neighborhood varies from instance to instance. In closing, we emphasize that our code is designed for the MRCPSP-MS. In fact, among the tested algorithms, only our code can be applied to different extension of RCPSP problem and achieve reasonable solution quality. We believe that the computational value of our TS procedure can be considerably improved by an extensive test which allows finding an adequate parameters setting of the TS.

Table 3: The computational results of RCPSP instances

Prob.	*av.(max) ΔLB*	*# best*	*av. (max) CPU*	*#new best*
PAT	0 (0)	110/110	1.24 (16)	-
ALV	-1.36 (4.13)	43/48	215.14	5

Table 2: The computational results of the FJSP instances

Prob.	*LB*	*TS*	*CPU(s)*	*M.G.*	*dev(%)*	*Pezzella*	*dev(%)*	*Chen*	*dev(%)*
Mk01	36	40*	2.34	40	0	40	0	40	0
Mk02	24	26*	1.27	26	0	26	0	29	+10.34
Mk03	(204)	204*	0.58	204	0	204	0	204	0
Mk04	48	60*	10.93	60	0	63	+4.76	63	+4.76
Mk05	168	173*	0.45	173	0	173	0	181	+4.41
Mk06	33	59	17.67	58*	-1.72	63	+6.34	60	+1.66
Mk07	133	141	15.4	144	+2.08	139*	-1.44	148	+4.72
Mk08	(523)	523*	0.02	523	0	523	0	523	0
Mk09	299	307*	8.85	307	0	311	+1.28	308	+0.32
Mk10	165	205	73.78	198*	-3.5	212	-7.07	212	+3.30

Table 4: Summary of computational results of the KN instances

Prob.	TS			MMSH		
	av.(max) ΔLB	*# best*	*av. (max) CPU(s)*	*av.(max) ΔLB*	*# best(%)*	*av. (max) CPU(s)*
n = 50	13.73 (17)	300/300	0.44 (17)	26.93 (33)	216/300	0.7
n = 100	19.64(25)	300/300	10 (80)	28.79 (39)	143/300	3.3
n = 150	27.42(46)	300/300	30(200)	28.79 (39)	137/300	8.33
n = 200	33.14 (43)	300/300	156(540)	28.79 (39)	83/300	16.33

6 CONCLUSION AND FUTURE RESEARCH

In this paper, a tabu search algorithm for solving the multi-mode resource constrained project scheduling problem with multi-skilled laborers is presented. It should be clear that this problem is a very difficult problem to solve (NP-hard). Moreover, to our knowledge, this problem has never been completely dealt with in literature, however, it can be encountered in many manufacturing area such as assembly systems. The neighborhood used in the proposed Tabu Search algorithm is based on reinserting critical activities on possibly different resource sets that minimize the makespan such that feasibility is maintained. It is shown that a neighbor can be computed quite efficiently and that a move can be evaluated without making it by computing a lower bound on the makespan after the move which speed up considerably the search process.

Although our procedure was implemented to resolve a very general scheduling problem, the results obtained seem to indicate that the performance of the proposed TS remains correct when applied to the standard RCPSP and outperforms the algorithms dedicated to FJSP. We believe that the computational value of our TS procedure can be considerably improved by an extensive test which allows finding an adequate parameters setting of the TS. Such experiments, however, are beyond the intended scope of this paper. Finally, the development of tight lower bounds for the MRCPSP-MS problem will undoubtedly constitute a promising area of future research.

7 REFERENCES

[1] Herroelen W., De Reyck B. and Demeulemeester E., 1998, Resource-Constrained Project Scheduling: A survey of recent developments, Computer Ops Res., 25, 279–302.

[2] Buddhakulsomsiri J., Kim D. S., 2006, Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting, EJOR.

[3] Lorenzoni L.L, Ahonen H., De Alvarenga A.G., 2006, A multi-mode resource-constrained scheduling problem in the context of port operations. Computers and Industrial Engineering, 50, 55 - 65.

[4] Mori M., Tseng C.C, 1996, A genetic algorithm for multi-mode resource constrained project scheduling problem, European Journal of Operational Research, 100, 134–141.

[5] Mastrolilli M., Gambardella L.M., 2000, Effective Neighborhood Functions for the Flexible Job Shop Problem, Journal of Scheduling, 3, 3–20.

[6] Blazewicz, J., Lenstra, J.K. and Rinnooy Kan, A. H. G., 1959, Scheduling projects to resource constraints: Classification and complexity, Discrete Applied Mathematics, 5, 11–24.

[7] Dauzere-Peres S., Roux J., Lasserre J.B., 1998, Multi-resource shop scheduling with resource flexibility, EJOR, 107, 289-305.

[8] Dauzere-Peres S., Pavageau C., 2003, Extensions of an integrated approach for multi-resource shop scheduling, IEEE Transactions, 33, 207–213.

[9] Tereso A. P., Mota J.R. and Lameiro R.J., 2005, Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in Java, Dynamic Programming Special Issue of the Journal of Control and Cybernetics, June 2005.

[10] Tereso A. P., Araújo M. M. and Elmaghraby S. E.,2004, The Optimal Allocation in Stochastic Activity Networks via the Electromagnetism Approach, Proceedings of the Project Management and Scheduling '04, Nancy - France.

[11] Glover, F., and Laguna, M., 1993, ''Tabu Search,'' in C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific, Oxford,.

[12] Hertz A., de Werra D., 1990, The Tabu Search Metaheuristic: how we used it, Annals of Mathematics and Artificial Intelligence 1, 111-121.

[13] Eikelder T. HMM., Aarts, B.J.M., Verhoeven, M.G.A., Aarts, EHL., 1997, Sequential and Parallel Local Search Algorithms for Job Shop Scheduling, Proceedings of the 2nd International Conference on Meta-heuristics, 75-80.

[14] Mausser, H.E., Lawrence, S.R., 1997, Exploiting block structure to improve resource-constrained project schedules. *In: Glover, F., Osman, I. and Kelley, J. (Eds.), Metaheuristics 1995: State of the art*, Kluwer, Maryland.

[15] Brandimarte, P., 1993, Routing and scheduling in flexible job shop by tabu search, Annals of Operation Research, 41, 157-183.

[16] Patterson, J., 1984, A Comparison of Exact Approaches for Solving the Multiple Constrained Resource Project Scheduling Problem, Management Science 30(7), 854–867.

[17] Alvarez-Valdés, R. and Tamarit, J.M., 1989, Heuristic Algorithms for Resource-Constrained Project Scheduling: A Review and an Empirical Analysis, In R. Słowiński and J. Weglarz (eds.), Advances in Projet Scheduling. Amsterdam: Elsevier, pp. 113–134.

[18] Kadrou Y., Najid M.N, 2006, A new heuristic to solve RCPSP with multiple execution modes and multi-Skilled Labor, IMACS Multiconference on Conference on Computational Engineering in Systems Applications.

[19] F. Pezzella, G. Morganti, G. Ciaschetti, 2007, A genetic algorithm for flexible job-shop scheduling problem, Computer & Operations Research, To appear.

[20] Chen, H., Ihlow, J. and Lehmann, C., 1999, A Genetic Algorithm for Flexible Job-Shop Scheduling, IEEE International Conference on Robotics and Automation, 1120-1125, Detroit.