DIMACS Technical Report 2002-14
April 2002

A Bi-objective Model for Maximizing the Quality in Project Scheduling

by

Mohamed Haouari
Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa. Tunisia

Mohammad A. Al-Fawzan
King Abdulaziz City for Science and Technology, P.O. Box 6086,
Riyadh 11442, Saudi Arabia

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS was founded as an NSF Science and Technology Center, and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

Traditionally, the Resource Constrained Project Scheduling Problem (RCPSP) is investigated in the operations research literature from the makespan minimization perspective. However, a recent survey conducted in the United States revealed the surprizing fact that the majority of project planners consider maximization of the quality of project schedules as the most important objective (Icmeli Tukel and Rom, 1998). In this paper, the integration of quality in project scheduling is investigated. For that purpose, the problem is modeled as a bi-objective resource-constrained project scheduling problem. A new objective defined as the schedule robustness is introduced as a quality measure. The maximization of this objective is considered along with the makespan minimization. A tabu search algorithm is developped in order to generate an approximate set of efficient solutions. Several variants of the algorithm are tested and compared on a large set of benchmark problems. The results are analyzed using statistical design of experiments techniques.

Keywords : resource-constrained project scheduling, multi-objective combinatorial optimization, tabu search, quality, design of experiments.

1 Introduction

Project scheduling deals with the allocation of scarce resources to a set of interrelated activities that are usually directed toward some major output and require a significant period of time to perform. Since the early days of operations research, project scheduling has attracted an ever growing attention. This interest is largely motivated by its great practical importance in diverse industries for designing new production processes, developing new products, setting up new facilities, and implementing new information systems, to cite just a few applications. The two well-known network techniques CPM and PERT, were both developed in the late 1950s to determine the minimum completion time of a project in such a way that the precedence relationships among activities are satisfied. However, in most practical situations the activities of a project require several additional resources. These resources may include manpower, machines, equipment, and capital budget (Blazewicz et al., 1986). Ignoring the resource constraints may result in an infeasible schedule with several resource conflicts. During the last twenty years, Resource Constrained Project Scheduling Problems (RCPSPs) have generated a great amount of research in the operations research literature. Solving a RCPSP means providing an optimal project schedule that not only satisfies the precedence constraints but also provide a feasible allocation of the required resources (i.e., resolve resource conflicts). Traditionally, two main categories of resource constraints are distinguished:

- *Renewable resources*: a specified amount of the resource is *continuously* available during the planning horizon (examples are machines and manpower).
- *Non-renewable resources*: a specified amount of the resource is available for the *entire* planning horizon (an example is the capital budget).

Moreover, in many practical situations, each activity may be performed according to several alternatives called *modes*. Each mode being characterized by a specific time duration and resource consumption.

In this paper, we address the single-mode RCPSP with renewable resources. Formally, this problem may be defined as follows: a project consists of a set I of n interrelated activities $1, .., n$. There are precedence constraints which specify that activity i can not be performed until all its predecessors P_i have been finished. There are K renewable resources. A constant amount of R_k units of resource k is continuously available from time zero onwards. The duration of activity i is p_i units of time. During this time period a constant amount of r_{ik} units of resource k is occupied. Preemption is not allowed. The objective is to determine starting (or finishing) times for the activities in such a way that a performance criterion is optimized, the precedence constraints are satisfied, and at each unit of time the total resource demand does not exceed the resource availability for each resource type.

We will not attempt to review the important RCPSP literature, since very recently, several exhaustive review papers have been published. Brucker et al. (1999) provide an extensive review of the RCPSP literature, quoting more than 200 relevant papers. They discuss both exact and heuristic recent approaches, and they present a new notation and classification scheme, similar to the one traditionally used in the machine scheduling literature. Herroelen et al. (1998) present an exhaustive survey of exact algorithms (branch and bound procedures) for several variants of the RCPSP. An exhaustive review of heuristic algorithms is presented in Kolisch and Hartmann (1998). Most of these heuristics are extensively evaluated in Hartmann and Kolisch (2000). Finally, Herroelen et al. (1997) present a survey paper exclusively focusing on project scheduling problems with discounted cash flows. Moreover, the reader can refer to the comprehensive monograph of Kolisch (1995).

However, it is worth noting that the considerable effort devoted hitherto to modeling and solving RCPSP has been almost exclusively focusing on three optimization criteria. The optimization criterion which has been the most extensively considered so far is the makespan (C_{max}) minimization, which is defined as the total time elapsed between the start and the end of the project. A second optimization criterion, which has also been widely used is the Net Present Value (NPV) maximization. This latter criterion is appropriate for capturing the monetary aspects of project management when important levels of cash flows (expenditures and/or payments) are present. Finally, the third optimization criterion which has also attracted attention is the Cost Minimization (CM). This objective includes the case where activities may be performed in several modes resulting in different costs and/or resource consumption. Surprisingly, a study conducted in the United States (Icmeli-Tukel and Rom, 1998) revealed that none of the aforementioned optimization criteria is entirely satisfactory from a practical perspective. Indeed, most project managers consider the objective of maximizing the quality of projects as the most important one.

In this paper, we investigate the meaning of "*maximizing the quality*" of a project schedule, and we show how the incorporation of quality can be achieved through a bi-objective modeling. It is worth noting that although project scheduling problems are often inherently multi-criteria decision problems, the literature on multi-objective project scheduling is surprisingly scant. Indeed, as far as we know, Slowinski (1989) is the first author who explicitly placed the RCPSP in a multi-objective framework. Recently, Hapke et al. (1998) presented a two-stage interactive search procedure aimed at handling a very general class of multiple-criteria project scheduling problem. In addition, Viana and de Sousa (2000) implemented the algorithm developed by Hansen (1997) to solve a multi-mode RCPSP with three objectives that are minimized: the makespan, the weighted lateness of activities, and the violation of resource constraints (thus, they considered "soft" resource constraints). However, contrary to the project scheduling area, an extensive literature dealing with multi-objective machine scheduling problems has been published so far. For a review of this work the reader is referred to Nagar et al. (1995).

The rest of the paper is organized as follows. In section 2, we analyze the previous work on ensuring quality in project scheduling, and we provide a detailed description of a bi-objective model for the RCPSP with an emphasis on quality. In section 3, we present a Tabu Search heuristic for our model. Several variants of the algorithm are compared on a series of benchmark problems using an experimental design. The computational results are reported in section 4. In the final section, we present our conclusions and discuss future research.

2 Incorporating quality in project scheduling

2.1 The model of Icmeli-Tukel and Rom

Quality is probably the most important paradigm introduced in organizations worldwide during the last two decades. The fierce global competition, compelled firms to adopt quality as a management philosophy, and as a strategic key element of success. However, very little work has been done in developing quantitative models incorporating quality in project scheduling. Hence, the pioneering work of Icmeli-Tukel and Rom (1997), could be considered as the only contribution in this field. In order to avoid any confusion, it is worth clarifying that we are dealing with the integration of quality in *project scheduling* and not in the *project management process*. This latter issue is extensively discussed in the project management literature (see for e.g., chapter 5 of Lockyer and Gordon, 1996, chapter 22 of Lewis, 1998, or chapter 7 of Turner, 1999).

In this section, we briefly review the work of Icmeli-Tukel and Rom, analyze the complexity of their model, and point out some important shortcomings. Reworks play a central role in the model developed by Icmeli-Tukel and Rom (1997). Since, activities may be performed at a poor quality and then require some additional rework time, thus, resulting in delaying the project completion time and increasing the cost. They propose a model, which they call the Resource Constrained Project Scheduling Problem with Reworks (RCPSPR). This model is similar to the standard RCPSP, but with two additional constraints to limit the total amount of rework time and the total amount of rework cost. For that purpose, they assume that both the rework time P_{it} and the rework cost RWC_{it} of activity i completing at time t, are known for all $i = 1, .., n$ and $t = 1, .., T$, where T is an upper bound of the completion time of the project. A compacted form of their MILP formulation, which is based on the classical formulation of Pritsker et al. (1969), is provided below. For the sake of simplicity, the precedence and the resource constraints are not reproduced in their mathematical form.

Decision variables :

x_{it} : binary variable which equals 1 if activity i is completed at time t, and 0 otherwise.
Q_1 : percentage of the total rework time used
Q_2 : percentage of the total rework cost used

Parameters :

D : Sum of the durations of all activities in the project

B : Total budget allocated to the normal functioning of a project.

$$Min(Q_1 + Q_2) \quad (1)$$
$$\sum_{i=1}^{n}\sum_{t=1}^{T} P_{it}x_{it} \leq DQ_1 \quad (2)$$
$$\sum_{i=1}^{n}\sum_{t=1}^{T} RWC_{it}x_{it} \leq BQ_2 \quad (3)$$
Precedence constraints between activities (4)
Resource constraints (5)
$$Q_1, Q_2 \in [0,1]\,,\ x_{it} \in \{0,1\} \quad \forall i,t \quad (6)$$

We provide a formal proof of the NP-hardness of the RCPSPR.

Lemma 1 *The RCPSPR is NP-hard in the strong sense*

Proof. It is easy to see that the RCPSPR is equivalent to a standard RCPSP with the objective of minimizing a weighted sum of completion times. Using notation of Brucker et al. (1999), this latter problem is denoted by $PS\,|prec|\sum w_jC_j$ (it is worth noting that the entry of the first field means that it is a "*project scheduling*" problem, the entry of the second field means that there are "*precedence*" constraints, and the entry of the third field means that the objective function is the minimization of a "*weighted sum of completion times*"). Indeed, at the optimum, the constraints (2) and (3) are necessarily binding. Therefore, if we replace in the objective function (1) Q_1 by $\sum_{i=1}^{n}\sum_{t=1}^{T} \frac{P_{it}}{D}x_{it}$ and Q_2 by $\sum_{i=1}^{n}\sum_{t=1}^{T} \frac{RWC_{it}}{B}x_{it}$, the objective function becomes $\sum_{i=1}^{n}\sum_{t=1}^{T} w_{it}x_{it}$, with $w_{it} = \frac{P_{it}}{D} + \frac{RWC_{it}}{B}$. Conversely, given an instance of a $PS \mid prec \mid \sum w_jC_j$ problem, an equivalent RCPSPR instance can be obtained by setting for any i and t, $P_{it} = \frac{Dw_{it}}{2}$, $RWC_{it} = \frac{Bw_{it}}{2}$ (where D and B are two arbitrary non-negative numbers). As a particular case of the $PS \mid prec \mid \sum w_jC_j$ is the $PS \mid prec \mid C_{\max}$ problem (i.e. makespan minimization). This can be seen, by assigning a weight equal to 1 to a dummy activity which represents the end of the project, and a weight equal to zero to all other activities. Since, the $PS \mid prec \mid C_{\max}$ is known to be strongly NP-hard (Blazewicz et al., 1983), the result follows. ■

Next, we briefly discuss several comments suggested by the RCPSPR model. Firstly, the practical applicability of the model is questionable. Indeed, the determination of the rework time and the rework cost functions, which play a central role in the model, is not clear. In particular, in our opinion, the rework cost of an activity i can not be determined *a priori*, since it depends not only on the completion time of this activity, but also on all the costs incurred after rescheduling the activities which are successors of activity i. Secondly, note that the transformation of the problem into an equivalent $PS \mid prec \mid \sum w_jC_j$ problem, highlights the fact that the weight function w_{it} is a dimensionless number which is the sum of two ratios of different nature (time and money). Hence, the model assumes a perfect substitutability between these two different measures. Finally, the model does not take into account the makespan. Though, as suggested by Pinnell and Busch (1993), the makespan is one of the most important measures of the quality of a project.

2.2 An alternative view of the quality of a schedule

The aforementioned shortcomings motivated us to consider different alternatives for integrating quality in project scheduling. But how to define the quality of a schedule? Recall that quality is defined as "*the total of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs*" (ISO 8402). This definition of quality is widely understood as the user's perception of the degree to which the product meets her/his expectations. This perception of the quality of a product is dependent, to a large extent, on three important factors: performance, conformance to specifications, and robustness of the design. In the context of project scheduling, the performance can be measured by the required duration to complete the project (Pinnell and Busch, 1993). Hence, the makespan minimization constitutes a valuable performance measure of a schedule, since competition urges project managers to complete all the activities of the project in a minimum time. For instance, the growing intensity of global competition forces companies to introduce new products at an accelerated pace. The speed at which a firm can succeed in introducing new products is dependent, in a large degree, on the schedule of the entire product development cycle. The conformance to specifications is obviously translated into the requirements of achieving all the activities while satisfying resource and precedence constraints. The third factor, which is the robustness of the design, has been first emphasized by Taguchi (1986) as a key factor of attaining good quality. In project scheduling, this factor is translated as the degree to which the planned schedule could be achieved as intended even if undesirable conditions occur. Indeed, several uncontrollable factors may cause the project to be late. Among these undesirable factors is the imperfect quality in the achievement of some activities, which may cause additional time for rework and correction of the detected defects. It is worth recalling that rework is considered by several authors as the primary cause of project schedule disturbance (Cooper, 1993, Lewis 1998, p. 265). Other factors which disrupt likewise a project schedule are the delay caused by unreliable deliveries from suppliers and unreliable subcontractors who are unable to meet the promised schedule.

2.3 A model for maximizing the quality of a project schedule

Owing to the determinants of the quality of a project schedule, we propose the development of a RCPSP model with two objectives: makespan minimization and robustness maximization. We define the robustness of a schedule, as its ability to cope with "small" increases in the time duration of some activities that may result from uncontrollable factors (i.e., with a limited effect on the completion time of the project). More precisely, our proposed model aims to construct schedules that are not only short but also that can be effectively implemented (as planned) without assuming that the activities will be carried out in ideal conditions. Thus, the problem is modeled as a bi-objective combinatorial optimization problem. It is worth noting that our model is in accordance with the expectations of the project planners as surveyed in Icmeli Tukel and Rom (1998), who wrote that "*...project managers perceive project scheduling as a problem with multiple objectives, the most important objective being the maximization of project quality*".

We propose to measure the robustness of a given schedule as follows: define the slack s_i as the difference between its earliest and latest precedence and resource feasible finish time. It corresponds to the amount of time that an activity i $(i = 1, .., n)$ can slip without causing a delay in the completion of the project. In practice, the activities are not equally liable to be subject to an undesirable delay. So, non-negative weight ρ_i are assigned to each activity i $(i = 1, .., n)$. Activities which are the most likely to be late have the largest weights. Thus, the robustness of a schedule is the weighted sum $\Omega = \sum_{i=1}^{n} \rho_i s_i$. Formally, the objective of our model, which we call the Bi-objective Resource Constrained Project Scheduling Problem (BRCPSP), is to determine starting times t_i for the activities $i = 1, .., n$ in such a way that:

- at each time t, the total resource demand does not exceed the resource availability for each resource type,
- the given precedence constraints are satisfied,
- the makespan $C_{max} = \underset{i=1,..,n}{Max} (t_i + p_i)$ is minimized, and
- the robustness $\Omega = \sum_{i=1}^{n} \rho_i s_i$ is maximized.

Extending the notation of Brucker et al. (1999), we denote our model by $PS \mid prec \mid \sum_{i=1}^{n} \rho_i s_i, C_{max}$. In the following section, we present a solution strategy for the BRCPSP.

3 Solution methodology

3.1 An overview of multi-objective combinatorial optimization

Multi-Objective Combinatorial Optimization (MOCO) problems in which H objectives $f_h(x)$ $(h = 1, .., H)$ are optimized while satisfying some constraints can be formulated as follows:

$$Max\ \{f_1(x), \ldots, f_H(x)\} \tag{7}$$
$$x \in D \subset \{0, 1\}^n \tag{8}$$

where D is a discrete set, and $x = (x_1, \ldots, x_n)$ is an n dimensional vector with binary variables. For the MOCO framework, the ideal case consists in finding the solution that maximizes the H objectives simultaneously. But, with respect to the feasible space, this solution does not usually exist. Therefore, solving (7-8) results in a set of *efficient solutions* rather than a single solution. A feasible solution x^* is said to be efficient if it is not possible to improve x^* simultaneously over all objectives.

The last decade witnessed an increasing interest in research dealing with MOCO problems. A non-exhaustive list of these problems includes the Minimum Spanning Tree problem (Hamacher, 1994), the knapsack problem (Ben Abdelaziz et al., 1999), the assignment problem (Tuyttens et al., 2000), and machine scheduling (Loukil et al., 2000). A survey of the related research is discussed in Ulungu and Teghem (1994), and a recent annotated bibliography is presented in Ehrgott and Gandibleux (2000).

Three types of strategies have been proposed so far to solve MOCO problems:

i) Enumerate the entire set of efficient solutions through exact methods like Branch and Bound (B&B) or Dynamic Programming (DP). These methods are very scarce in the literature and most of them are designed for specific problems. For instance, the B&B algorithm of Visée et al. (1998) provides the whole set of efficient solutions of the bi-objective knapsack problem, and the DP algorithm of Klamroth and Wiecek (2000) generates all the non-dominated solutions of different complex knapsack models.

ii) It is also possible to approximate the set of efficient solutions by means of constructive heuristics (see for instance Chattopadhyay and Momoh, 1999), or metaheuristics such as Genetic Algorithms (Murat and Ishibuchi, 1997), Tabu Search (Ben Abdelaziz et al., 1999), and Simulated Annealing (Ulungu et al., 1999).

iii) The third strategy is to cooperate with the Decision Maker (DM) in order to obtain his *preferred solutions*. This process is called an *interactive method*. Interactive methods include either exact methods or heuristic methods since at each step, the procedure has to search for some efficient solutions in order to present them to the DM. This approach is particularly designed for the case where the set of efficient solutions is large. Teghem et al. (2000), for example, proposed a simulated annealing based interactive method in order to help a DM to make a choice between the approximate set of efficient solutions. This method was experimented on the knapsack and the assignment multi-objective problems.

As it is widely known for single objective (NP-hard) combinatorial optimization problems, exact methods can only solve (in a reasonable amount of time) small sized MOCO problems. In addition, the cardinality of the efficient set of solutions increases not only with the problem size but also with the number of criteria. For this reason, approximate methods are largely preferred when dealing with practical sized MOCO problems.

In the sequel, we present an algorithm for the BRCPSP based on the second strategy, and aimed at providing an approximate set of nondominated schedules.

3.2 A Multi-Objective Tabu Search algorithm for the BRCPSP

Tabu Search (TS) is a local search metaheuristic designed by Glover (1989, 1990). Roughly speaking, TS is an iterative algorithm, which takes as an input a feasible solution S and replaces it with a new one obtained after performing a single transformation of S. The set of feasible solutions that can be obtained by performing a single transformation of S is called the *neighborhood* of S. The main characteristic of TS is that the search technique forbids moving to solutions already visited in order to avoid cycling by the use of a flexible memory that contains a subset of the history of the search. It accepts a deterioration of the solution for an eventual improvement, and it applies other improved operators such as diversification, intensification and aspiration criteria in order to avoid local optima. A detailed presentation of this technique can be found in Glover and Laguna (1997). During the last decade, TS has been widely used to solve a large variety of combinatorial optimization problems. In particular, TS has been applied for the solution of the classical single objective RCPSP by Pinson et al. (1994), Lee and Kim (1996), and Baar et al. (1999).

In this section, we propose the adaptation of the principles of TS to the BRCPSP. Other implementations of Multi-Objective Tabu Search ($MOTS$) algorithms were proposed by Gandibleux et al. (1996), Hansen (1997), and Ben Abdelaziz et al. (1999). Since we are concerned with a multi-objective framework, the problem may have more than one solution forming the efficient set. The adaptation of TS to this context generates an approximation of this set. Moreover, because of the combinatorial aspect of the BRCPSP, the search must generate both supported and unsupported efficient solutions. We organize the presentation of our $MOTS$ algorithm by describing successively the schedule representation, the neighborhood structure, the selection scheme, the tabu list attributes and the aspiration criterion.

3.2.1 Schedule representation

A schedule π is represented as a permutation $L = (j_{(1)}, ..., j_{(n)})$ of the n activities. Given a list L, a corresponding precedence and resource feasible schedule can be built by using a *Schedule Generation Scheme* (SGS). As pointed out by Kolisch and Hartmann (1998), SGS is the core of most heuristics for the RCPSP. This algorithm starts from scratch and constructs in n steps a *feasible* schedule, so that at step i ($i = 1, .., n$) an activity j is selected and scheduled at the earliest precedence and resource feasible start time. At each step, an activity is selected in the following way:

- Firstly, we compute the set A of eligible activities. An activity is said to be eligible if all its predecessors are already scheduled
- Secondly, we choose among the eligible activities the one which has the smallest index in the list L.

For that purpose, we use a Forward Recursion Procedure.

Notation

$\check{R}_k(t)$ = amount of resource k available at time t.
$Pred_j$ = set of immediate predecessors of activity j.
S_j^e = earliest start time of activity j.
C_j^e = earliest completion time of activity j.
U = set of unscheduled activities
$\{0\}$ = dummy activity with zero processing time, no resource consumption, and which precedes all the activities.
$T^e = \left\{C_j^e : j \notin U\right\}$
$\left[C_{(q)}\right]_{q=0,n}$ = list of the elements of T^e ranked in a non-decreasing order .

SGS - Forward Recursion Procedure

Initialization : $U = J$, $T^e = \{0\}\,, C_{(0)} = 0$
For $i = 1$ to n do
  Compute $A = \{j \in U : Pred_j \cap U = \emptyset\}$
  Select the activity $j = j_{(i)}$ such that $i = \min\limits_{j_{(s)} \in A} \{s\}$
  Set $U = U \backslash \{j\}$,
  For all $t \in T^e$ do
    For $k = 1, ..., K$ do
      Compute $\check{R}_k(t)$
  $S_j^e := \underset{i \in Pred_j}{Max} \{C_i^e\} = C_{(q)}$
  While ($\forall\ t \in \left[S_j^e, S_j^e + p_j\right], \forall\ k = 1, ..., K,\ \check{R}_k(t) \geq r_{jk}$) is False do
    $q := q + 1$
    $S_j^e := C_{(q)}$
  End (while)
  $C_j^e := S_j^e + p_j$
  $T^e = T^e \cup \left\{C_j^e\right\}$,Update C
End (for i)

This algorithm provides the earliest feasible completion times. The makespan of the project is $C_{max} = \underset{j=1,..,n}{Max} \left\{C_j^e\right\}$. However, in order to compute the slacks of the activities, we need to compute the latest feasible completion times. Here, we use also a SGS-type algorithm. For that purpose, we construct an ordered list of the n activities $L' = (u_{(1)}, u_{(2)}, ..., u_{(n)})$, where activities are ranked in a non-decreasing order of their earliest completion time. In order to compute the latest completion times, we use a backward procedure so that at step i ($i = n, .., 1$) the activity $u_{(i)}$ is included in the partial schedule and scheduled at the latest precedence and resource feasible completion time. Before providing a formal description of the procedure, we extend the previous notation :

S_j^l = latest start time of activity j.
C_j^l = latest completion time of activity j.
$Succ_j$ = set of immediate successors of activity j.
$\{n+1\}$ = dummy activity with zero processing time, no resource consumption, and which is preceded by all the activities ($u_{(n+1)} = \{n+1\}$).
$T^l = \left\{S_j^l : j \notin U\right\}$
$[S_{(p)}]_{p=n+1,1}$ = list of the elements of T^l ranked in a non-increasing order .

SGS - Backward Recursion Procedure

Initialization : $U = J$, $T^l = \{n+1\}$, $S_{(n+1)} = C_{\max}$.
For $i := n$ down to 1 do
  $j = \ u_{(i)}$
  For all $t \in T_i^l$ do
    For $k = 1, ..., K$ do
      Compute $\check{R}_k(t)$
  $C_j^l = \underset{i \in Succ_j}{Min} \ \{S_i^l\} = S_{(q)}$
  While ($\forall t \in [S_j^e, C_j^l]$, $\forall\ k = 1, ..., K$, $\check{R}_k(t) \geq r_{jk}$) is False do
    $q := q + 1$
    $C_j^l := S_{(q)}$
  End (while)
  $S_j^l := C_j^l - p_j$
  $T^l = T^l \cup \left\{S_j^l\right\}$, Update S
End (for i)

Note that at step i the activity $j = u_{(i)}$ is included in the partial schedule and scheduled at the latest precedence and resource feasible start time, so that the K resources are *continuously* available during the time interval $[S_j^e, C_j^l]$. The slack of an activity j ($j = 1, ..., n$) is $s_j = C_j^l - C_j^e$.

3.2.2 Neighborhood structure

At a given iteration, let π^0 be the current feasible schedule with a corresponding priority list $L(\pi^0)$. We generate N neighbors by randomly selecting N feasible moves (N is a problem parameter which denotes the neighborhood size). A move (i, j) consists of swapping in the list $L(\pi^0)$ the activities i and j, thus resulting in a new list $L(\pi^{ij})$ with a corresponding neighbor schedule π^{ij}. A move (i,j) is said to be feasible if activity i is neither a predecessor nor a successor of activity j.

In order to reduce the computational burden, and to avoid feasibility tests, we keep a static set F containing all the feasible moves. This set is constructed as follows. Define P_j as the set of all predecessors (not necessarily immediate) of activity j $(j = 1, ..., n)$. Obviously, a move (i,j) is feasible if and only if $i \notin P_j$ and $j \notin P_i$. In order to obtain the sets P_j, we first build a precedence-feasible list of the n activities $L_0 = (\ q_{(1)},\ \ q_{(2)}, ...,\ \ q_{(n)})$. The list L_0 as well as P_j are obtained by using the following simple procedures:

Procedure generation of L_0

For $k = 1$ to n do
  Select an activity i such that $Pred_i = \emptyset$
  $q_{(k)} = i$
  For $j \in Succ_i$ do $Pred_j = Pred_j \setminus \{i\}$
End (For k)

Procedure construction of P_j

For $k = 1$ to n do
  $j = q_{(k)}$
  $P_j = Pred_j$
  For $i \in Pred_j$ do $P_j = P_j \cup P_i$
End (For k)

Therefore, the set of feasible moves is:

$$F = \{(i,j) : i = 1, ..., n-1; j = i+1, ..., n \text{ and } (i \notin P_j \text{ and } j \notin P_i)\}$$

3.2.3 Selection strategy

The list L_0 is used to generate an initial schedule. Assume that π^0 is the current feasible schedule. For each neighbor schedule π^u $(u = 1, ..., N)$, compute the makespan (C_{max}) and the robustness $(\sum_{i=1}^{n} \rho_i s_i)$. Let $z_M(u)$ and $z_R(u)$ denote the corresponding values, respectively. Consider the aggregation function defined as $z_\lambda(u) = \lambda z_M(u) - (1-\lambda) z_R(u)$ (where $0 \leq \lambda \leq 1$). At a given iteration, we select as a new current solution the neighbor schedule π^{u^*} such that $z_\lambda(u^*) = \min_{u=1,N} z_\lambda(u)$. The weight λ plays an important role since it aims to reach a particular supported efficient solution and then generate its efficient neighborhood, so that the interior of the feasible set is explored in order to get unsupported efficient solutions. Since non-supported, nondominated points are typically not far from the nondominated frontier, the generated set is expected to be a good sample of the whole set of efficient solutions. In our implementation, we used the set of weights $W = \{\lambda_v = \frac{v}{s} : v = 0, 1, ..., s\}$, where s is a non-negative integer parameter denoted hereafter by the *number of steps*. Therefore, our $MOTS$ algorithm consists in running $s+1$ times a single objective TS, each with a particular value of the weight λ. Note that, the particular values of λ_s and λ_0 tend to provide the approximate solutions of the minimum makespan and the maximum robustness, respectively.

3.2.4 Tabu list attributes and aspiration criterion

Since a move consists of selecting a feasible pair $(i, j) \in F$, then we store in the tabu list the last performed moves. In order to override the tabu list when there is a good move, we implement the following aspiration criterion: a tabu move is accepted if it produces a solution which outperforms the best solution obtained so far. This comparison is performed with respect to the aggregation function.

3.2.5 Approximation of the efficient set

An approximate set AE_v of potentially efficient solutions of the single objective problem defined by the weight $\lambda_v\,(v = 0, 1, .., s)$, is obtained in the following way: the set AE_v is initialized with the initial schedule. At each step of the TS, the generated neighborhood is scanned and the set of potentially efficient solutions AE_v is updated. This operation consists in removing dominated schedules (a schedule π^x dominates a schedule π^y if $z_M(x) < z_M(y)$ and $z_R(x) \geq z_R(y)$, or $z_M(x) \leq z_M(y)$ and $z_R(x) > z_R(y)$). If after a fixed number of iterations ($Iter_{max}$) there is no improvement of AE_v then the algorithm stops. An approximation AE of the efficient set E is obtained after dropping the dominated schedules from the union set $\bigcup\limits_{v=0,..,s} AE_v$.

3.2.6 Synthesis of the $MOTS$ algorithm

Initialization

Construct the set F of feasible moves

Construct the list L_0 and generate an initial schedule π^0

$AE = \emptyset$, $TL = \emptyset$ (TL denotes the tabu list)

Iterative step

For $v = 0$ to s do

$AE_v = \{\pi^0\}, \lambda_v = \frac{v}{s}$

Repeat

Select randomly N moves from F (non-tabu moves and/or satisfying the aspiration criterion)

Construct the set of neighbors $N(\pi^0) = \{\pi^u : u = 1,...,N\}$

Update the tabu list TL

Compute for each π^u the makespan $z_M(u)$ and the robustness $z_R(u)$

$AE_v = AE_v \cup N(\pi^0)$

Remove from AE_v all dominated schedules

Select π^{u^*} such that $z_{\lambda_v}(u^*) = \min_{u=1,..,N} (\lambda_v z_M(u) - (1-\lambda_v) z_R(u))$
$\pi^0 = \pi^{u^*}$
Until ($Iter_{\max}$ iterations are performed with no improvement of AE_v)
$AE = AE \cup AE_v$
End (for)
Remove from AE all dominated schedules

4 Computational experiments

4.1 Objectives of the experiments

The experiments are designed with the goal of providing specific answers to the following questions:

- What factors affect significantly the performance of the algorithm?
- What is the variation of the CPU time?
- How robust is the algorithm?
- What is the empirical distribution of the cardinality of AE?
- What is the performance of the algorithm on the makespan minimization?

4.2 Test problems

We tested our approach on the benchmark problem set $j30$ generated with PROGEN, which is a problem generator developed by Kolisch and Sprecher (1997). This problem set is specifically designed for the RCPSP, and it consists of 480 instances with $n = 30$ activities and $K = 4$ resources. The weights ρ_i $(i = 1, .., n)$ were randomly drawn from the discrete uniform distribution $U[1, 5]$.

4.3 Parameters selection

Our $MOTS$ algorithm allows for different choices of five parameters: the tabu list size TL, the number of iterations before stopping $Iter_{max}$, the number of steps s, the neighborhood size N, and the aggregation function. We performed some preliminary experiments and we found no evidence that the variation of the two first factors (namely, TL and $Iter_{max}$) plays any significant role. Hence, based on these findings, we fixed the tabu list size to $TL = 7$, and the number of iterations before stopping to $Iter_{max} = 0.5n$ (where n represents the number of activities). Moreover, we assigned two levels for each of the three remaining factors in the following way:

- Number of steps : $s = 5$ and 10
- Neighborhood size : $N = \lceil \theta \, |F| \rceil$, with $\theta = 0.05$ and 0.1 ($|F|$ is the cardinality of the set F)
- Aggregation function AF : $z_\lambda(u)$ and $w_\lambda(u) = \lambda w_M(u) + (1-\lambda) w_R(u)$ where:

  $w_M(u)$: relative improvement of the makespan $= \frac{z_M(0)-z_M(u)}{z_M(0)}$ ($z_M(0)$ is the makespan of the current solution).

  $w_R(u)$: relative improvement of the robustness $= \frac{z_R(u)-z_R(0)}{z_R(0)}$ ($z_R(0)$ is the robustness of the current solution).

Note that when $w_\lambda(.)$ is used as the aggregation function, then at each step, the best neighbor is the schedule π^{u^*}such that $w^\lambda(u^*) = \max\limits_{u=1,N} w^\lambda(u)$. The use of $w_\lambda(.)$ is motivated by the observation that very often the robustness is much larger than the makespan, and, as a consequence, it strongly dominates it in the aggregation function $z_\lambda(.)$. This drawback is avoided by considering only relative improvements.

After combining the three factors, eight different variants V_h $(h = 1, .., 8)$ are obtained. The characteristics of these variants are displayed in Table 1.

Variant	Number of steps	Neighborhood size	Aggregation function
V_1	5	0.05	$z_\lambda(.)$
V_2	10	0.05	$z_\lambda(.)$
V_3	5	0.1	$z_\lambda(.)$
V_4	10	0.1	$z_\lambda(.)$
V_5	5	0.05	$w_\lambda(.)$
V_6	10	0.05	$w_\lambda(.)$
V_7	5	0.1	$w_\lambda(.)$
V_8	10	0.1	$w_\lambda(.)$

Table 1 - Characteristics of the 8 variants

4.4 Performance measure

The eight variants were coded in Basic and compiled with the Quick Basic 4.5 compiler. All the computational experiments were carried out on a Pentium II 350 MHz Personal Computer. In order to analyze the relative performance of the variants, all of the 480 instances of the set $j30$ were successively solved by the eight variants. For each instance i, we computed the approximate nondominated set AE_{ih} and the CPU time t_{ih}, obtained after running the variant V_h $(h = 1, .., 8)$.

It is worth noting, that the problem of selecting the best variant has no simple answer. Indeed, in MOCO, and contrary to single objective combinatorial optimization, there is no straightforward basis of comparison of heuristics, and addressing the issue of designing a reliable measure of performance is by itself a challenging problem. In this context, an interesting approach developed by Czyzac and Jaszkiewicz (1998) provides an estimate of the distance of the approximate nondominated set to the exact one (denoted by R). In the case of the BRCPSP, this latter measure is based on the distance measure $d(\sigma, R)$ between a feasible schedule σ and the nondominated set R and defined as:

$$d(\sigma, R) = \min_{\mu \in R} \left\{ \max\left(\frac{|C^{\sigma}_{\max} - C^{\mu}_{\max}|}{\Delta\left(C_{\max}\right)}, \frac{|\Omega^{\sigma} - \Omega^{\mu}|}{\Delta\left(\Omega\right)}\right) \right\} \tag{9}$$

where $C^{x}_{\max}$ and Ω^{x} represent the makespan and the robustness of a given schedule x, respectively, and $\Delta\left(C_{\max}\right)$ and $\Delta\left(\Omega\right)$ represent the range of the makespan and the robustness in the nondominated set R, respectively (the reader may have noticed that it is implicitly assumed that the nondominated set is *not* a singleton). Hence, a measure of the quality of a given approximate set AE_{ih} may be defined as:

$$d(AE_{ih}, R) = \frac{1}{|AE_{ih}|} \sum_{\sigma \in AE_{ih}} d(\sigma, R) \tag{10}$$

This measure offers the advantage of providing a reliable estimate of the proximity of a heuristic set to the nondominated frontier. However, since in our experiments the exact nondominated set is unknown, we have to replace in (9-10) the exact set R by the approximate set AE_i obtained after removing the dominated schedules from the union set $\bigcup_{h=1,8} AE_{ih}$. Unfortunately, the drawback of this measure is that it is not appropriate for the case where the approximate set AE_i is a singleton which contains just one approximate ideal point (and in our experiments we found relatively many such cases).

An alternative measure of performance, that we used in our experiments, and which was introduced by Ulungo (1993), is based on the following ratio:

$$\varepsilon_{ih} = \frac{|AE_{ih} \cap AE_i|}{|AE_i|} \tag{11}$$

Although this latter ratio is insensitive to the "closeness" to the approximate nondominated set, it provides a measure of the "*contribution*" of variant h to the approximate nondominated set AE_i. Consequently, if variant h dominates (is dominated) by the other variants, then ε_{ih} will be close to 1 (0). Moreover, it can be used in all cases, including the case where AE_i is a singleton.

Table 2 presents the results obtained with each of the eight variants. For each variant h $(h = 1, ..., 8)$ the sum $\varepsilon_h = \sum_{i=1,480} \varepsilon_{ih}$, as well as the mean CPU times t_h in seconds.

Variant	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
ε_h	237.54	260.03	255.98	273.98	250.20	280.07	271.89	310.87
t_h	14.72	23.92	28.24	45.90	14.67	23.56	28.65	45.57

Table 2 - Performance results

4.5 Statistical design of experiments

In order to ensure that the results of our experiments can be rigorously analyzed by statistical methods, we used the well-known statistical design of experiments (Box et al., 1978, Montgomery, 1997). Specifically, a 2^3 factorial experiment was used to identify which factors among s, θ, and AF have a significant effect on the quality of variants. The main results of the analysis of variance are presented in Table 3.

Source of variation	Factor effect	Sum of squares	Mean square	Computed f
Main effect				
s	0.0569	3.114	3.114	21.186
θ	0.0442	1.876	1.876	12.601
AF	0.0445	1.903	1.903	12.950
Two-factor interaction				
$\{s, \theta\}$	0.0024	0.005	0.005	0.0371
$\{s, AF\}$	0.0147	0.209	0.209	1.425
$\{\theta, AF\}$	0.0104	0.105	0.105	0.716
Three-factor interaction				
$\{s, \theta, AF\}$	0.0071	0.048	0.048	0.328
Error		563.234	0.146	
Total		570.496		

Table 3 - Analysis of Variance

The analysis reveals that at the 1% level of significance, all the three factors are significant ($F_{0.01,1,\infty} = 6.63$). However, there is no evidence that there is any interaction between the factors, except a very mild interaction effect between the number of steps and the aggregation function ($P - value \simeq 20\%$). Therefore, the analysis confirms that the best performance is obtained with a large number of steps, a large neighborhood size, and the aggregation function $w_\lambda(.)$.

As expected, we observe that the CPU time increases (almost linearly) as s and/or θ increase. However, statistical testing (namely, the so-called *paired comparison design*) strongly supports the argument that the aggregation function has no significant impact on the running time.

4.6 Robustness analysis

A very desirable algorithmic characteristic of a heuristic is its *robustness*. This robustness may be defined as the ability of the heuristic to generate an approximate solution which is not sensitive to the variation of the seeds. In order to analyze the robustness of our best variant (namely, V_8), we selected randomly 50 instances from the set $j30$. Each instance i was run 10 times with different seeds, while keeping all the parameters fixed. For each instance i ($i = 1, ..., 50$), we denote AE_{ik} the approximate nondominated set generated after run k ($k = 1, ..., 10$), and AE_i the approximate nondominated set which results from the union of the sets AE_{ik} ($k = 1, ..., 10$). Then, we computed the ratio ε_{ik} using (11). Now consider the mean value :

$$\varepsilon_i = \frac{1}{10} \sum_{k=1,10} \varepsilon_{ik}$$

Clearly, the distribution of the series (ε_i) provides a good idea on the robustness of the approach. Hence, a large value of ε_i (i.e. close to 1) means that, *on average*, a large percentage of the set of nondominated schedules (AE_i) is generated in each run of instance i, regardless of the used seed. On the contrary, a small value of ε_i (i.e. close to 0) indicates that there is a significant difference between the approximate sets obtained with different seeds.

In Figure 1, we report the results obtained with the fifty instances (the instances are numbered by increasing ε_i).

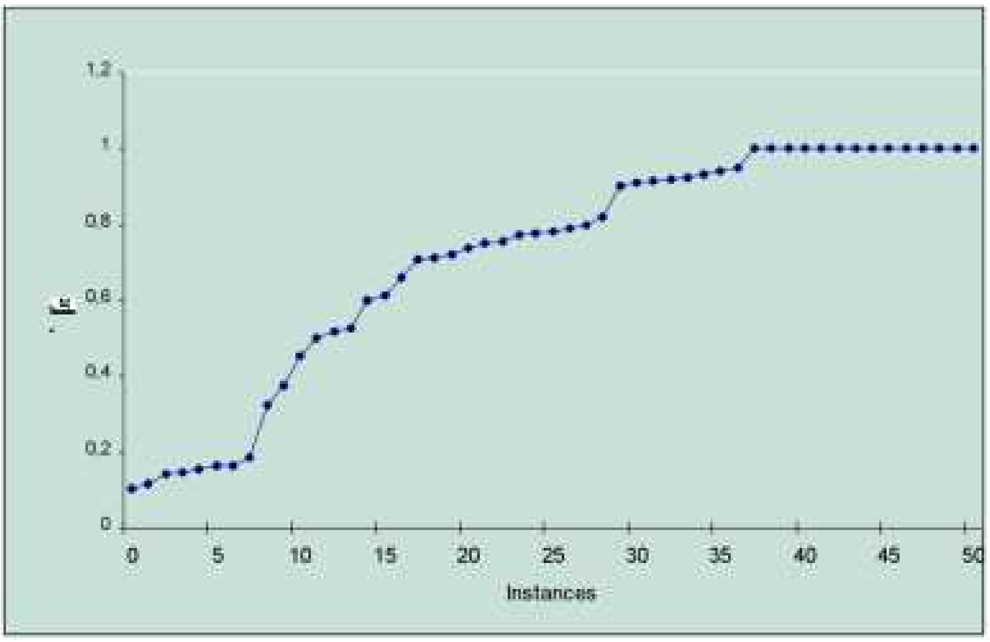

Figure 1 - Analysis of the robustness

Figure 1 indicates that the approach is fairly robust. Indeed, we found that the mean value of the (ε_i) is 0.711 and the median is 0.785.

4.7 Analysis of the cardinality of AE

It is of primary importance in multi-objective modeling to have an estimate of the cardinality of the approximate set of efficient solutions. Indeed, the approach may not be appropriate if the number of suggested efficient solutions is extremely large. In this case, an interactive method must be used in order to remove some efficient solutions according to the decision maker preferences. However, in our computational experiments, we observe that in most cases $|AE|$ is of moderate size. Indeed, we found that (for the set $j30$) 120 instances have only one single approximate nondominated solution. Table 4 provides the empirical distribution of the cardinality of the approximate efficient set obtained over the set $j30$. The mean (maximum) cardinality is 12.47 (44).

$\lvert AE \rvert$	1	2-10	11-18	19-26	27-44
$Frequency$ (%)	25.00	19.37	23.95	25.62	6.04

Table 4 - Empirical distribution of $|AE|$ for the set $j30$

4.8 Performance on makespan minimization

The approximate nondominated set AE obtained after running the $MOTS$ algorithm can be used to derive a good approximate solution to the (single objective) RCPSP with minimum makespan objective function. For that purpose, it suffices to scan the set AE and to select the schedule π^{u^*} with minimum makespan. Hence, the approach consists in running $s+1$ times a tabu search algorithm, each time considering as an optimization criterion the single objective $w_v(.)$ obtained through the aggregation of the relative improvement of makespan *and* the robustness:

$$w_v(u) = \tfrac{v}{s} w_M(u) + (1 - \tfrac{v}{s}) w_R(u) \quad v = 0, ..., s$$

We used this strategy and compared the minimal makespan schedule provided by the $MOTS$ algorithm with the optimal makespan. For that purpose, we ran the variant V_8 on the entire set $j30$ (480 instances). The optimal values are taken from Demeulemeester and Herroelen (1997). We found that 384 instances out of 480 (80%) were solved exactly, and that the average percentage deviation from optimal solution is 0.54%. The maximal deviation being 7.89%. We compared our results with those obtained by three specialized algorithms, namely:

- Two tabu search algorithms ($TS1$ and $TS2$) both developed by Baar et al. (1999), and which are currently considered as the best state-of-the-art implementations of tabu search for the RCPSP.
- A genetic algorithm (GA) devised by Hartmann (1998), and which is currently considered as best published heuristic for the RCPSP (as observed in Hartmann and Kolisch, 2000).

The results of the comparison are presented in Table 5.

	$TS1$	$TS2$	GA	$MOTS$
Problems solved to optimality (out of 480)	347	389	not available	384
Average deviation (%)	0.7	0.4	$0.54^{(1)}/0.25^{(2)}$	0.54
Maximal deviation (%)	7.5	6.9	not available	7.89
Average CPU time (in sec.)	$2^{(3)}$	$15^{(3)}$	not available	48.3

Table 5 - Performance of the $MOTS$ algorithm on the makespan minimization

(1) This deviation was obtained after 1000 evaluations

(2) This deviation was obtained after 5000 evaluations

(3) This time was obtained on a Sun Ultra 2 workstation

The results indicate that even though our algorithm is not specifically designed for makespan minimization, it compares rather favorably with the best existing special purpose algorithms. However, as expected, the mean computing time is comparatively very long for $MOTS$. This is due to two main reasons. Firstly, running $MOTS$ is equivalent to solving $(s+1)$ times a single objective RCPSP, and secondly, at each iteration the robustness of each neighbor schedule is evaluated, which is time consuming.

5 Conclusion

In this paper, we considered the RCPSP from the perspective of maximizing the quality. This objective is considered by project planners as the most important one. We analyzed the previous works, pointed out some of their shortcomings, and modeled the problem as a bi-objective resource-constrained project scheduling problem. Two objectives were considered : makespan minimization and robustness maximization. The importance of the first objective is obvious in the present context of global competition which compels companies to complete the project in a minimum time. The second objective, which is equivalent to the maximization of a weighted sum of slacks, is a measure of the ability of a schedule to be completed within the promised date. A multi-objective tabu search algorithm has been devised for solving the bi-objective resource constrained project scheduling problem. Several variants of the algorithm were compared using statistical design of experiments. The best variant was thoroughly tested on a large sample of 480 benchmark problems and it proved to be robust and to generate in most cases a moderately sized set of efficient solutions. Moreover, its performance on the single objective makespan minimization is comparable to the best existing tabu search algorithms.

For future research, at least two issues are worth investigating. Firstly, it would be interesting to generalize the present model to include non-renewable resources as well as multiple execution modes for each activity. Secondly, it appears that the number of efficient solutions increases with the number of activities. Thus, an extension which deserves to be investigated is to embed the present work in an interactive strategy which aims to reduce the number of proposed schedules.

References

[1] Baar T., P. Brucker, S. Knust (1999), Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem. *Metaheuristics : Advances and Trends in Local Search Paradigms for Optimization*, S. Voβ, S. Martello, I. H. Osman, C. Roucairol (Eds.), 1-18, Kluwer Academic Publishers.

[2] Ben Abdelaziz F., S. Krichen, J. Chaouachi (1999), A hybrid heuristic for multiobjective knapsack problems. *Metaheuristics : Advances and Trends in Local Search Paradigms for Optimization*, S. Voβ, S. Martello, I. H. Osman, C. Roucairol (Eds.), 205-212, Kluwer Academic Publishers.

[3] Blazewicz J., J. K. Lenstra, A. H. G. Rinnooy Kan (1983), Scheduling subject to resource constraints. *Discrete Applied Mathematics* 5, 11-24.

[4] Blazewicz J., W. Cellary, R. Slowinski, J. Weglarz (1986), Scheduling under resource constraints-Deterministic models. *Annals of Operations Research* 7.

[5] Box G., W. Hunter, S. Hunter (1978), *Statistics for experimenters*, John Wiley & Sons.

[6] Brucker P., A. Drexl, R. Mohring, K. Neumann, E. Pesch (1999), Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research* 112, 3-41.

[7] Chattopadhyay D., J. Momoh (1999), A multiobjective operations planning model with unit commitment and transmission constraints, *IEEE Transactions on Power Systems* 14, 1078-1084.

[8] Cooper K. G. (1993), The rework cycle: benchmarks for the project manager. *Project Management Journal* 24, 17-21.

[9] Demeulemeester E., W. Herroelen (1997), New benchmark results for the resource-constrained project scheduling problem. *Managment Science* 43, 1485-1492.

[10] M. Ehrgott, X. Gandibleux (2000), A survey and annotated bibliography of multiobjective combinatorial optimization, *OR Spektrum* 22, 425-460.

[11] Glover F. (1989), Tabu Search, Part I. *ORSA Journal on Computing* 1, 190-206.

[12] Glover F. (1990), Tabu Search, Part II. *ORSA Journal on Computing* 2, 4-32.

[13] Glover F., M. Laguna (1997), *Tabu search*, Kluwer Academic Publishers.

[14] Gandibleux X., N. Mezdaoui, A. Fréville (1996), A multiobjective tabu search procedure to solve combinatorial optimisation problems. Working paper, LAMIH-LIMAV, Université de Valenciennes, 1996.

[15] Hamacher H. (1994), On spanning tree problems with multiple objectives, *Annals of Operations Research* 52, 209-230.

[16] Hansen P. (1997), Tabu search for multiobjective optimization: MOTS. Technical Report, Institute of mathematical Modelling, Technical University of Denmark.

[17] Hartmann S. (1998), A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics* 45, 733-750.

[18] Hartmann S., R. Kolisch (2000), Experimental state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research* 127, 394-407.

[19] Hapke M., A. Jaszkiewicz A., R. Slowinski (1998), Interactive analysis of multiple-criteria project scheduling problems. *European Journal of Operational Research* 107, 315-324.

[20] Herroelen W. S., P. Van Dommelen, E. L. Demeulemeester (1997), Project network models with discounted cash flows a guided tour through recent developments. *European Journal of Operational Research* 100, 97-121.

[21] Herroelen W., B. De Reyck, E. L. Demeulemeester (1998), Resource-constrained project scheduling : a survey of recent developments. *Computers and Operations Research* 25, 279-302.

[22] Icmeli-Tukel O., W. O. Rom (1997), Ensuring quality in resource constrained project scheduling. *European Journal of Operational Research* 103, 483-496.

[23] Icmeli-Tukel O., W. O. Rom (1998), Analysis of the characteristics of projects in diverse industries. *Journal of Operations Management* 16, 43-61.

[24] Klamroth K., M. M. Wiecek (2000), Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics* 47, 57-76.

[25] Kolisch R. (1995), *Project scheduling under resource constraints, efficient heuristics for several problem cases*, Physica-Verlag.

[26] Kolisch R., A. Sprecher (1997), PSPLIB - A project scheduling library. *European Journal of Operational Research* 103, 483-496.

[27] Kolisch R., S. Hartmann (1998), Heuristic algorithms for solving the resource-constrained project scheduling problem : classification and computational analysis. To appear in *Handbook on Recent Advances in Project Scheduling*, J. Weglarz (ed.). Kluwer.

[28] Lee J. K., Y. D. Kim (1996) Search heuristics for resource-constrained project scheduling. *Journal of Operational Research Society* 47, 678-689.

[29] Lewis J. P. (1998), *Mastering Project Management*, McGraw Hill.

[30] Lockyer K., Gordon J. (1996), *Project Management and Project Network Techniques.* Pitman Publishing.

[31] Loukil T., J. Teghem, D. Tuyttens (2000), Solving scheduling problems with the MOSA method. *Proceeding of the ACICDA'2000 IEEE Conference*, *Methods and heuristics for decision making,* 88-93. Monastir, Tunisia.

[32] Murat T., H. Ishibuchi (1997), Performance of MOGA for flow shop scheduling problems in Kuriynis (ed.), *Proceeding of the 14th Conference on Production Research*, 498-501. Osaka Inst. Tech., Japan.

[33] Montgomery D. (1997), *Design and analysis of experiments*, Wiley.

[34] Nagar A., J. Haddock, S. Heragu (1995), Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research* 81, 88-104.

[35] Pinnell S., J. S. Busch (1993), How do you measure the quality of your project management?, *PM Network*, December, 35-36.

[36] Pinson E., C. Prins, F. Rullier (1994), Using tabu search for solving the resource-constrained project scheduling problem. Working paper, Université Catholique de l'Ouest, Angers.

[37] Pritsker A., L. Watters, P. Wolfe (1969), Multiproject scheduling techniques with limited resources: A zero-one programming approach. *Management Science* 16, 93-107.

[38] Slowinski R. (1989), Multiobjective project scheduling under multiple-category resource constraints, in: Advances in Project Scheduling, R. Slowinski and J. Weglarz (eds.), Elsevier, Amsterdam.

[39] Taguchi G. (1986), *Introduction to quality engineering*, Asian Productivity Organization, Tokyo.

[40] Teghem J., D. Tuyttens, E. L. Ulungu (2000), An iteractive method for multi-objective combinatorial optimization. *Computers and Operations Research* 27, 621-634.

[41] Turner R. (1999), *The Handbook of Project-Based Mangement*. McGraw Hill.

[42] Tuyttens D., J. Teghem, P. Fortemps, K. Van Nieuwenhuyse (2000), Performance of the MOSA method for the bicriteria assignement problem. To appear in *Journal of Heuristics*.

[43] Ulungo E. L. (1993), Optimisation combinatoire multicritère: détermination de l'ensemble des solutions efficaces et méthodes interactives. Unpublished Ph.D. dissertation. Faculté Polytechnique de Mons, Belgium.

[44] Ulungu E. L., J. Teghem (1994), Multi-objective combinatorial optimization problems: a survey. *Journal of Multiple Criteria Decision Analysis* 3, 83-104.

[45] Ulungu E. L., J. Teghem, P. Fortemps, D. Tuyttens (1999), MOSA method: a tool for solving MOCO problem. *Journal of Multi-Criteria Decision Analysis* 8, 221-236.

[46] Viana A., J. P. de Sousa (2000), Using metaheuristics in multiobjective resource constraines project scheduling. *European Journal of Operational Research* 120, 359-374.

[47] Visée M., J. Teghem, M. Pirlot, E. L. Ulungu, An interactive heuristic method for multi-objective knapsack problem. *Journal of Global Optimization* 12, 139-155.