Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 55 (2008) 183–194
computers & industrial engineering
www.elsevier.com/locate/caie

A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling

Hédi Chtourou [a,*], Mohamed Haouari [b]

[a] Institut Préparatoire aux Études d'Ingénieurs de Sfax, Département de Technologie, Sfax, Tunisia
[b] Combinatorial Optimization Research Group–ROI, Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa, Tunisia

Received 28 November 2004; received in revised form 30 May 2007; accepted 30 November 2007
Available online 28 January 2008

Abstract

Traditionally, the resource-constrained project scheduling problem (RCPSP) is modeled as a static and deterministic problem and is solved with the objective of makespan minimization. However, many uncertainties, such as unpredictable increases in processing times caused by rework or supplier delays, random transportation and/or setup, may render the proposed solution obsolete. In this paper, we present a two-stage algorithm for robust resource-constrained project scheduling. The first stage of the algorithm solves the RCPSP for minimizing the makespan only using a priority-rule-based heuristic, namely an enhanced multi-pass random-biased serial schedule generation scheme. The problem is then similarly solved for maximizing the schedule robustness while considering the makespan obtained in the first stage as an acceptance threshold. Selection of the best schedule in this phase is based on one out of 12 alternative robustness predictive indicators formulated for the maximization purpose. Extensive simulation testing of the generated schedules provides strong evidence of the benefits of considering robustness of the schedules in addition to their makespans. For illustration purposes, for 10 problems from the well-known standard set J30, both robust and non-robust schedules are executed with a 10% duration increase that is applied to the same randomly picked 20% of the project activities. Over 1000 iterations per instance problem, the robust schedules display a shorter makespan in 55% of the times while the non-robust schedules are shown to be the best performing ones in only 6% of the times.

Keywords: Resource-constrained project scheduling; Robustness; Stability; Two-stage algorithm; Priority rules

1. Introduction

The resource-constrained project scheduling problem (RCPSP) deals with the allocation of scarce resources to a set of interrelated activities that are usually related by precedence constraints and directed toward some major goal. Due to its practical importance in various industrial fields (construction, product and process design, implementation of communication systems, etc.), the RCPSP continuously attracts the attention of

* Corresponding author. Tel.: +216 98 667 530; fax: +216 74 246 347.
E-mail address: hedi.chtourou@ipeis.rnu.tn (H. Chtourou).

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2007.11.017

researchers and project managers. A vast amount of literature addressed the RCPSP solution using either optimization or heuristic approaches (Demeulemeester and Herroelen, 2002; Kolisch and Hartmann, 1999). Despite the fact that the RCPSP is inherently multi-objective, it is more often solved deterministically for minimizing the project makespan (C_{max}).

However, Icmeli-Tukel and Rom (1998) revealed that, from a practical perspective, the project planners consider that maximizing the schedule quality should be considered as the main criterion. Such a quality is related to three aspects: performance, conformance and robustness of the schedule. The first two aspects correspond to minimizing the makespan while respecting the precedence and resource constraints. On the other hand, the deterministic scheduling outcome may be seriously jeopardized by the uncertainties characterizing the environment in which the project is actually executed. Such uncertainties include rework caused by quality deficit, machine breakdowns, employee absenteeism and delay in materials supply. All of these causes may result in one or more project activities requiring more processing time than that anticipated in baseline schedule. Hence, the schedule robustness may be defined as "its ability to cope with small increases in the time duration of some activities that may result from uncontrollable factors" (Al Fawzan and Haouari, 2005). When this robustness is measured in terms of project duration it is referred to as quality-robustness. However, if it is measured in terms of the deviation between the planned and realized start times of the projected schedule, it is referred to as solution-robustness or stability. An ideal schedule should combine both types (Van de Vonder et al., 2005).

Recently, Herroelen and Leus (2004b, 2005) provided a survey of the various approaches of scheduling under uncertainties. These approaches can be classified into four main scheduling categories: reactive, stochastic, fuzzy and proactive.

In the first category, the generation of a baseline schedule is accomplished first with no anticipation of variability. Second, various rules and heuristics are used in the project execution phase to correct the schedule in case of disruption occurrence (Artigues and Roubellat, 2000; Calhoun et al., 2002). The corrective rules may range from simple right shift of disruption-affected activities to complete rescheduling. Approaches belonging to the category of stochastic project scheduling basically consider the RCPSP as a multi-stage decision process requiring a priori knowledge about the distributions of activity time duration (Fernandez et al., 1996; Möhring et al., 1984; Pet-Edwards et al., 1998). Their major shortcoming is a failure to provide a baseline schedule.

On the other hand, the fuzzy project scheduling approaches are based on the notion of fuzzy activity duration and produce fuzzy schedules (i.e. fuzzy start and finish times). The application of these approaches requires the use of membership functions that define the activity duration distributions (Hapke and Slowinski, 2000; Wang, 2004). However, from a practical perspective, it may be difficult to estimate uncertainties that affect the project execution. Hence, it may not be easy to determine realistic fuzzy membership functions and stochastic activity duration distribution. This can seriously limit the domain of application of these two categories of approaches.

As for the proactive scheduling category, robust schedules that account for variability are generated. This category is also referred to as robust scheduling. The literature related to this topic is much more abundant in the machine scheduling field than in that associated with the project scheduling. Goldratt (1997) applied the theory of constraints to the RCPSP to propose the critical chain/buffer management (CC/BM) methodology. This methodology is based on inserting time buffers into a good makespan schedule in order to improve its quality-robustness. Although this methodology has gained a great popularity in the scientific and in the managerial communities, Herroelen and Leus (2001) pointed a certain number of its pitfalls resulting from the oversimplification of the RCPSP reality.

More recently, Herroelen and Leus (2004a) developed mathematical programming models for generating stable (solution-robust) baseline schedules to face activity duration disruptions. Their approach, also based on inserting time buffers in a pre-schedule, aims at the minimization of a function representing the cost of overrunning or underrunning the start time of the activities. It makes abstraction of resource requirements. Leus and Herroelen (2004) have dropped the hypothesis of unrestricted resource availability by using a resource flow network for robust resource allocation to a feasible baseline schedule. A branch-and-bound algorithm was proposed by the authors for this purpose.

Van de Vonder et al. (2005) addressed the issue of achieving a trade-off between quality-robustness and solution-robustness for resource-unconstrained project scheduling. They used simulation in order to investi-

gate whether it is beneficial to concentrate safety time in project and feeding buffers as in the original CC/BM approach and a modified CC/BM approach developed in their paper, or whether it is preferable to spread time buffers throughout the project schedule, as done by the adapted float factor model (ADFF) developed by Leus (2003). A similar research, but with consideration of resource constraints, was presented by Van de Vonder et al. (2006).

In addition, Al Fawzan and Haouari (2005) developed a multi-objective tabu search heuristic for solving a bi-objective RCPSP. They considered the objectives of quality-robustness maximization along with makespan minimization and used several variants of the algorithm in order to find an approximate set of efficient solutions. Unlike the previously cited robust scheduling researches, Al Fawzan and Haouari (2005) do not advocate buffer insertion and therefore, avoid the associated inevitable increase of the initial planned makespan. Instead, they search the solution space for "naturally" robust schedules. Abbasi et al. (2006) proposed to solve the same bi-objective RCPSP using a simulated annealing meta heuristic. However, as it is the case for all multi-objective optimization problems, choosing a solution from the often-large approximate set of efficient solutions is neither an obvious nor a simple task.

Finally, some researches in the single machine scheduling context used the flexibility concept rather than robustness (Aloulou et al., 2002; Mauguiere et al., 2002). They propose schedules that may be decomposed into several interchangeable fragments in case of disruption occurrence.

This study aims at developing a robust scheduling approach generating schedules that are invulnerable to small increases in activity durations that may be caused by rework or supplier delays. It addresses the solution of the single-mode RCPSP with renewable resources using a simple two-stage-priority-rule-based approach. The RCPSP is first solved for minimum project duration then for maximum quality-robustness while avoiding the bi-objective dilemma. It also proposes some efficient "predictive" robustness indicators intended to assist the project managers in selecting the most robust schedule among a set of schedules sharing the same makespan. Finally, this paper aims at demonstrating the benefits of using the robustness concept through extensive simulation runs on a set of randomly generated benchmark problems.

The remainder of the paper is organized as follows: in Section 2, a formal definition of the RCPSP is given along with the definitions of the priority rules and the robustness indicators required by the approach. Section 3 presents a detailed description of the proposed two-stage algorithm while the main results of the proposed approach application on a set of benchmark problems are reported in Section 4. Finally, the general conclusions and the directions for future research are presented in Section 5.

2. Notation and definitions

Formally, the single-mode RCPSP could be defined as follows: a project consists of a set I of n interrelated activities $(1,\ldots,n)$ related by precedence constraints which specify that activity i cannot 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. Two performance criteria are considered in this approach:

- The makespan $C_{\max} = \underset{i=1,\ldots,n}{Max}\ (t_i + p_i)$ is minimized
- A predictive robustness measure (RM) is maximized

In the remainder of this paper, the term "robustness" refers to quality-robustness. It represents the schedule's ability to cope with small increases in the time duration of some activities. In addition, no buffer insertion is advocated for schedule robustness enhancement. The presented approach aims at selecting from a set of schedules sharing a "very good" makespan the one offering the best intrinsic robustness. Actually, in order to predict the robustness of a given schedule, 12 surrogate RMs are proposed and assessed through the simulation study presented in Section 4. They all are based on the notion of activity free slack s_i defined as the

Table 1
Robustness measures

$RM_1 = \sum_{i=1}^{n} s_i$	$RM_5 = \sum_{i=1}^{n} \alpha_i$	$RM_9 = \sum_{i=1}^{n} \min(s_i, frac \cdot p_i)$
$RM_2 = \sum_{i=1}^{n} s_i \cdot NSucc_i$	$RM_6 = \sum_{i=1}^{n} \alpha_i \cdot NSucc_i$	$RM_{10} = \sum_{i=1}^{n} \min(s_i, frac \cdot p_i) \cdot NSucc_i$
$RM_3 = \sum_{i=1}^{n} \left(s_i \cdot \sum_{k=1}^{K} r_{ik}\right)$	$RM_7 = \sum_{i=1}^{n} \left(\alpha_i \cdot \sum_{k=1}^{K} r_{ik}\right)$	$RM_{11} = \sum_{i=1}^{n} \left(\min(s_i, frac \cdot p_i)_i \cdot \sum_{k=1}^{K} r_{ik}\right)$
$RM_4 = \sum_{i=1}^{n} \left(s_i \cdot NSucc_i \cdot \sum_{k=1}^{K} r_{ik}\right)$	$RM_8 = \sum_{i=1}^{n} \left(\alpha_i \cdot NSucc_i \cdot \sum_{k=1}^{K} r_{ik}\right)$	$RM_{12} = \sum_{i=1}^{n} \left(\min(s_i, frac \cdot p_i) \cdot NSucc_i \cdot \sum_{k=1}^{K} r_{ik}\right)$
With	$NSucc_i$: Number of immediate successors of activity i; $i = 1 \ldots, n$ $\alpha_i = 1$ if $s_i > 0$ and $\alpha_i = 0$ if $s_i = 0$ i; $i = 1, \ldots, n$ $0 < \text{frac} < 1$	

amount of time that an activity i ($i = 1, \ldots, n$) can slip without delaying the start of any of its immediate successors while upholding resource feasibility. Free slack is calculated by $s_i = LS_i - ES_i$ where $ES_i(LS_i)$ is the earliest (latest) start time of activity i as determined by the standard forward (backward) recursion procedure (Hartmann and Kolisch, 2000). The latest start time of every activity is defined as the latest time at which the activity could start without delaying any of its successors earliest start time. These measures are given in Table 1.

RM_1 is the sum of free slacks over all the project activities. It corresponds to the measure adopted by Al Fawzan and Haouari (2005). This measure gives the same weight to all activities with no regard to their number of successors or their resource requirements. However, an increase in the duration of an activity having a large number of successors is more likely to affect the project makespan than an increase in the duration of a job with very few successors. Therefore, in the proposed measure RM_2, the free slacks of all activities are weighed by the numbers of their respective successors before being summed up. Similarly, when a duration increase occurs, the resources required by the affected activity will be seized and made unavailable for a longer period. The more the delayed activity requires resources the more the project makespan is likely to increase because of unplanned resource unavailability. This is expressed by the RM_3 measure in which the free slacks of all activities are weighed by their respective aggregate resource requirements before being summed up. Measure RM_4 combines the RM_2 and RM_3 *measures. It simultaneously accounts for the* successor number and resource requirement weights.

In addition, measures RM_1 to RM_4 could be biased by very large s_i values. In fact, these measures will be increased in a disproportionate manner compared to the real ability of the schedule to cope with small duration increases. In other words, just a small free slack is sufficient to absorb a small duration increase. Summing up large free slacks in the robustness measures may wrongfully inflate them. Consequently, eight other measures are proposed. First, measures RM_5 to RM_8 are the equivalent of measures RM_1 to RM_4 with the free slack s_i being replaced by a binary variable α_i. This variable is set to unity if the activity free slack exists and nullified otherwise. Hence, this group of measures consider only the possible existence of a slack and grants no importance to its actual length. Second, measures RM_9 to RM_{12} are the equivalent of measures RM_1 to RM_4 with the free slack s_i being replaced by the minimum among s_i and a fraction *frac* of the activity duration p_i. This fraction equals the average percentage increase in activity duration. Hence, if the free slack is small, it is included as it is; if it is very large, only its potentially useful part is considered by this last group of measures.

3. Two-stage robust schedule generation approach

3.1. General description

The two_stage robust schedule generation approach is based on the so-called *biased-sampling-multi-pass method* featuring an enhanced version of the serial schedule generation schemes (SGS) coupled with the use of priority rules (Hartmann and Kolisch, 2000). The robust schedule is determined following a two phase methodology:

- Phase I: The heuristic is run a large number of times with the sole objective of minimizing the project duration C_{max}. The smallest value generated thus far is taken as a threshold value for the next phase.
- Phase II: The heuristic is re-run a large number of times with the objective of maximizing the *RM* while keeping the C_{max} at a level equal to or smaller than the threshold value.

Since the objectives pursued in both phases are different, the best performing priority rule in phase I will not necessarily be the most appropriate one in phase II. Therefore, it is not viable to combine the two phases in a unique phase in which the robustness objective would be targeted while the C_{max} threshold would continuously be updated.

3.2. Phase I description

In this method, a large number of iterations, each resulting in a different schedule, is executed. The current best schedule generated is retained. Thus, each iteration consists of three main steps. In the first step, the activity priority values, issued from the selected priority rule (*PR*), are used to obtain the selection probabilities. The second step is a random-biased selection of eligible activities according to their selection probabilities. In the third step, selected activities are scheduled to their precedence and resource feasible earliest start times (Kolisch & Hartmann, 1999). A total of 16 *PRs* reported from the RCPSP literature are used in this study and are listed in Table 2 (Hartmann & Kolisch, 2000; Yang, 1998).

Moreover, two important additional features are incorporated in order to enhance the performance of the heuristic.

3.2.1. Random partial destruction and reconstruction of the incumbent solution

The incumbent solution is partially destroyed and then reconstructed, in the sense that, a random number of its last selected activities are discarded before being rescheduled again. Hence, to generate a new schedule on the basis of the incumbent solution, a random number of the first activities of the latter are considered as already scheduled and then, the RCPSP is solved for the still unscheduled activities using the same serial SGS. The reader should notice that this procedure is different from the so-called ''lower bound destructive improvement'' proposed by Klein and Scholl (1999).

3.2.2. Solution of the symmetric problem

For each activity, successors become predecessors and vice versa. In fact, it is of common knowledge that the regular scheduling problem and its symmetric counterpart share the same optimal solution (Cavalcante

Table 2
Priority rules used for activity selection

Abbreviation	Priority rule
MaxDur	Maximum duration
MinDur	Minimum duration
MaxRR	Maximum resource requirement
MinRR	Minimum resource requirement
MaxSuc	Maximum number of direct successors
MinSuc	Minimum number of direct successors
LST	Latest starting time
LFT	Latest finishing time
MinSlk	Minimum activity free slack
MaxSlk	Maximum activity free slack
MaxRPW	Maximum rank positional weight
MinRPW	Minimum rank positional weight
MaxCRR	Maximum cumulated resource requirement
MinCRR	Minimum cumulated resource requirement
MaxCSuc	Maximum cumulated number of successors
MinCSuc	Minimum cumulated number of successors

et al., 2000; Khemekhem and Chtourou, 2006; Möhring et al., 2003). Hence, trying to solve the symmetric problem is intended to enlarge the search domain for the best solution. Moreover, the incumbent solution of the symmetric problem was partially destroyed and reconstructed. Therefore, the total number of iterations (*ITER*) of the first phase was split on four quarters as described in Box 1, where r is the iteration counter and C is the makespan of the just obtained solution

Box 1 Phase I general scheme

- Initialization
  - $C_1 = LargeNumber$
  - $C_1^s = LargeNumber$

- For $r = 1$ to $0.25 * ITER$
  - Solve the regular problem with a forward recursion pass
  - If $(C < C_1)$ update the best solution yet S_1 $(C_1 = C)$

- For $r = (0.25 * ITER + 1)$ to $0.50 * ITER$
  - Partially and randomly destruct schedule S_1
  - Reconstruct schedule S_1
  - If $(C < C_1)$ update the best solution yet S_1 $(C_1 = C)$

- For $r = (0.50 * ITER + 1)$ to $0.75 * ITER$
  - Solve the symmetric problem for minimum $C_{\max}$
  - If $(C < C_1^s)$ update the best solution yet $S_1^s (C_1^s = C)$

- For $r = (0.75 * ITER + 1)$ to $ITER$
  - Partially and randomly destruct schedule S_1^s
  - Reconstruct schedule S_1^s
  - If $(C < C_1^s)$ update the best solution yet $S_1^s (C_1^s = C)$

- If $(C_1 \leqslant C_1^s)$
  S_1 is the best schedule and C_1 is the threshold makespan value
  Else
  S_1^s is the best schedule and C_1^s is the threshold makespan value

These enhancements have shown significant improvement of the solution and the additional computational burden they caused is negligible (computational requirements are provided in Section 4). Hence, for the best found *PR*, we observed that the mean deviation (MD) with regard to the optimal solution was 25% lesser than the solution obtained with the classical algorithm with the same number of iterations (5000). Table 3 presents the performance of the first phase of the algorithm with the three best *PRs* among the sixteen investigated with $ITER = 5000$. It is worth noting that in sets J60 and J90 (Kolisch et al., 1998, chap. 9), the performance of the schedule generation heuristic is assessed using the MD with regard to the upper bound (best known solution) and also, with regard to the critical path-based lower bound (Kolisch and Hartmann, 2006). In fact, for these two standard sets, the large problem instance size does not permit solving for the optimal solution through exact methods.

Table 3
Performance of the schedule generation heuristic

Rule	J30		J60				J90			
	Deviation/opt. sol. (%)		Deviation/ub. sol. (%)		Deviation/lb. sol. (%)		Deviation/ub. sol. (%)		Deviation/lb. sol. (%)	
	Av.	Sd.	Av.	Sd.	Av.	Sd.	Av.	Sd.	Av.	Sd.
MaxRPW	0.45	1.15	1.71	3.10	3.72	6.81	2.47	4.36	4.57	8.28
MaxCRR	0.51	1.24	2.01	3.40	4.04	7.08	2.67	3.45	4.76	8.24
MaxCSuc	0.58	1.32	1.98	3.46	4.02	7.15	2.62	4.33	4.71	8.15

Av., average of the 480 deviation values; Sd., standard deviation of the 480 deviation values; Opt. sol., optimal solution; ub. sol., upper bound solution; lb. sol., lower bound solution.

The obtained results generally compare well to the results of similar biased random sampling approaches especially for the small sized problems (Kolisch and Hartmann, 1999, 2006). In particular, the *MaxRPW* rule proved to be the best performing rule for both minimizing makespan and also for finding the largest number of optimal solutions over the 480 instance problems of each set. This rule attributes to each activity the value of its own duration plus those of all its direct and indirect successors. Hence, *MaxRPW* is considered as the reference *PR* for phase I. In other words, it will be the only rule used in phase I of the approach in all the upcoming applications of this work.

3.3. Phase II description

Phase II of the proposed approach is intended to find the most robust schedule with a makespan not larger than the threshold value found in phase I. The same number of iterations as in phase I are executed. Each iteration starts by the execution of a forward recursion using one particular *PR* among the 16 investigated. This allows one to determine the project makespan as well as every activity i earliest completion time EC_i ($i = 1,\ldots,n$). A backward recursion pass is then carried out in order to obtain for each activity i, its latest completion time LC_i and consequently, its free slack $s_i = LC_i - EC_i$ ($i = 1\ldots,n$). This step is performed only in case the makespan is not larger than the phase I threshold value. Finally, the algorithm can compute a *RM* selected among those of Table 2. This measure is used for the selection of the most robust schedule respecting the makespan constraint issued from phase I. The general scheme of phase II is depicted in Box 2.

Box 2 Phase II general scheme

- Initialization
  - –Threshold makespan value (phase I, *MaxRPW*) $C_1^r = min(C_1, C_1^s)$
  - $-RM_{\max} = 0$
- For $r = 1$ to *ITER*
  - –Execute forward recursion to determine a schedule S with a makespan C;
  - –If ($C \leqslant C_1^r$)
    - * Find robustness measure *RM* by a backward recursion procedure;
    - * If ($C < C_1^r$ OR $RM > RM_{\max}$)
      - ·Update best schedule of phase II: $S_2 = S$;
      - ·Update best schedule *RM*: $RM_{\max} = RM$;
      - ·Update best schedule makespan: $C_2 = C$;

4. A simulation study

4.1. Scope

The simulation study presented in this section is intended to demonstrate the benefits of considering not only deterministic makespan but also schedule robustness for projects facing possible disruptions of their activity durations. Thus, two series of simulations are performed:

- Series 1: 10 problem instances from the benchmark problem set J30 (Kolisch et al., 1998, chap. 9)
- Series 2: 10 problem instances from the benchmark problem set J60 (Kolisch et al., 1998, chap. 9)

For each series, the goal was to determine if the robust schedules perform better than the minimum-makespan-non-robust schedules, when a random activity duration disruption scheme is applied to each project. In this work, a schedule S_a with a corresponding makespan C_a is said to outperform a schedule S_b with a corresponding makespan C_b whenever we have $C_a < C_b$. In addition, since the disruption scheme is stochastic, a large number of iterations is to be performed for each problem instance in order to guarantee statistical viability. Furthermore, many *PRs* could be used to generate schedules and many *RMs* are available to estimate their robustness. Therefore, the study also aims at assessing the performances of the various *PRs* and *RMs* in terms of generating schedules that are sufficiently robust to cope with small disruptions.

4.2. Description of the simulation procedure

In order to better illustrate the effects of the robustness on the final performance of the schedules, the simulation runs are planned for the comparison of schedules with the same makespans but with very different *RMs*. Therefore, for every problem instance of each series, two schedules are obtained: a "non-robust" schedule S_1 with a makespan C_1 issued from phase I and a "robust" schedule S_2 with a makespan C_2 ($C_2 \leqslant C_1$) issued from phase II. However, as described in Section 3.2, phase I classically solves the RCPSP for makespan minimization with no regard to robustness. Therefore, it was necessary to slightly adjust this first phase in order to obtain the "non-robust" schedule S_1. This is done in the following manner. First, every forward recursion pass of phase I is followed by a backward recursion pass. This permits one to obtain the free slacks required for calculating *RM*. Second, the test for updating the incumbent solution is now twofold. If a schedule with a better makespan ($C < C_1$) is found, it is automatically saved and its *RM* is assigned to the variable holding the minimum robustness value $RM_{\min}$. If a schedule with the already attained makespan is found, it is saved only if its *RM* is worst than $RM_{\min}$ ($C = C_1$ AND $RM < RM_{\min}$). Both C_1 and $RM_{\min}$ are updated whenever a new solution is saved. Without this adjustment, schedule S_1 would have an arbitrary value of *RM* and the comparison with S_2 may be ambiguous.

For each of the *NProb* problem instances, the two previously obtained schedules are first retrieved. A number *ITER* of iterations, each comprising three main steps, are then run. In the first step, the durations of randomly picked $\gamma\%$ of the project activities *are* increased by $\eta\%$ of their original values. Subsequently, the new post-disruption makespans of schedules S_1 and S_2 are computed in the second step. This is done by executing an activity-list-based serial SGS on S_1 then on S_2, respectively (Kolisch and Hartmann, 1999). This procedure differs from the already used *PR*-based serial SGS in the fact that two important steps are no longer required. These are: finding the eligible activity set and selecting one activity among this set. In fact, an activity list is, by definition, a precedence feasible one. The list used in this task is simply the list of activities following their selection order when scheduled in the first time. This guarantees obtaining the post-disruption $C_{\max}$ of the same schedule. In fact, the order of the activities in the schedule will remain the same since the random selection step is removed. Finally, the third and final step consists in updating the proportions P_1 (P_2), each time schedule $S_1(S_2)$ outperforms schedule $S_2(S_1)$. The proportion difference ($P = P_2 - P_1$) is considered as the most revealing performance measure since it shows the net advantage of the robust schedule over the non-robust one. The simulation procedure general scheme is summarized in Box 3.

Box 3 General simulation scheme

- Initialization of performance measures
  - $P_1 = 0$ (P_1: Proportion of iterations where S_1 outperforms S_2);
  - $P_2 = 0$ (P_2: Proportion of iterations where S_2 outperforms S_1);
- For *prob = 1* to *Nprob*
  - Schedule retrieving
    - Retrieve the "non-robust" schedule S_1 with a makespan C_1 obtained from phase I;
    - Retrieve the "robust" schedule S_2 with a makespan C_2 obtained from phase II;
  - For $r = 1$ to *ITER*
    - Activity duration alteration
      - Randomly pick a percentage $\alpha\%$ of activities to have their duration altered;
      - Increase the duration of the picked activities by $\beta\%$ of their original values;
    - Simulation of the execution of schedules S_1 and S_2 with altered durations
      - Execute an activity list forward recursion on $S_1$to obtain new makespan C'_1;
      - Execute a forward recursion pass with activity list S_2 to obtain new makespan C'_2;
    - Updating proportions
      - If $(C'_1 < C'_2)$

        $P_1 = P_1 + 1/(ITER * Nprob)$;
      - If $(C'_2 < C'_1)$

        $P_2 = P_2 + 1/(\text{ITER} * Nprob)$;
- $P = P_2 - P_1$

4.3. Results

For each of the two simulation series, the 16 *PRs* were investigated together with the 12 *RMs* in a full factorial design. In addition, the number of runs has an influence on the quality of the solution random biased sampling methods (Kolisch and Hartmann, 2006). The iteration numbers reported in the literature vary between 1000 and 50,000. In this study, each of the two phases of the robust schedule generation algorithm was carried out 20,000 times for each instance. It is worth mentioning here that, since many solutions may share the same makespan and the same *RM*, the re-execution of phases I and II on an instance problem does not necessarily generate the same schedule. Besides, the activity duration alteration scheme was as follows: $\gamma = 20\%$ of activities to have their duration increased by $\eta = 10\%$ of their original values. Also, 1000 iterations per run were sufficient to achieve results repeatability in the simulation phase. Hence, for each couple (*RM*, *PR*), the sample sizes necessary to compute the standard deviations of proportions P_1 and P_2 are, respectively: $N_1 = N_2 = ITER * NProb = 1000 * 10 = 10{,}000$.

In order to demonstrate that the proposed approach provides a significantly higher proportion of best performing schedules we test the one sided following hypotheses with a significance threshold of 0.001

Table 4
Advantage of using of robust schedules (problems from J30)

RM	*PR*	P_2	σ_{P2}	P_1	σ_{P1}	P	σ_{P2-P1}	z
RM_7	*MinRR*	0.56	0.0050	0.09	0.0029	0.47	0.0066	70.956
	MaxSlk	0.52	0.0050	0.08	0.0027	0.44	0.0065	67.893
	MinCRR	0.51	0.0050	0.09	0.0029	0.42	0.0065	64.807
	MaxCSuc	0.52	0.0050	0.13	0.0034	0.39	0.0066	58.878
RM_1	*MinRR*	0.40	0.0049	0.14	0.0035	0.26	0.0063	41.411
	MaxSlk	0.52	0.0050	0.10	0.0030	0.42	0.0065	64.214
	MinCRR	0.54	0.0050	0.07	0.0026	0.47	0.0065	72.184
	MaxCSuc	0.55	0.0050	0.06	0.0024	0.49	0.0065	75.256

P_2: Proportion of iterations in which schedule S_2 outperforms schedule S_1 (over a 10,000 iterations sample).
σ_{P2}: Standard deviation corresponding to P_2.
P_1: Proportion of iterations in which schedule S_1 outperforms schedule S_2 (over a 10,000 iterations sample).
σ_{P1}: Standard deviation corresponding to P_1.
P: Net advantage of S_2 over S_1 ($P = P_2 - P_1$).
σ_{P2-P1}: Standard deviation of the proportion difference.
z: Standard normal decision variable.

- H_0: P_2 is not larger than P_1
- H_1: P_2 is larger than P_1

This is done by comparing the standard normal decision variable $z = (P_2 - P_1)/\sigma_{P2-P1}$ to the critical minimal value $z_{cr} = 3.08$ where, $\sigma_{P2-P1} = \sqrt{p \cdot (1-p) \cdot (1/N_1 + 1/N_2)}$ and $p = (N_1 \cdot P_1 + N_2 \cdot P_2)/(N_1 + N_2)$ (Tabachnik & Fidell, 1989). For simulation series 1 and with all the (*RM*, *PR*) couples, the results clearly confirm that the one sided null hypothesis is rejected since the smallest P value was 0.12, associated with $z = 19.6$ much higher than the critical value z_{cr}. This result corresponds to $P_2 = 0.33$, $P_1 = 0.19$ obtained for the couple (RM_1, *MaxRPW*). For all remaining (*RM*, *PR*) couples, the P and z values are larger, demonstrating a significant superiority of "robust" schedules S_2 over the "non-robust" ones S_1. The average and the maximum values of P were, respectively, 0.33 and 0.49. Table 4 depicts the results of the best performing *PRs* coupled with the best *RMs*.

As for series 2, results showed that, on the opposite of the first series, robust scheduling of larger projects could be effectively performed through some (*RM*, *PR*) couples only. In fact some *PRs* (*MinDur*, *MaxSuc*) exhibited negative values of P for all the *RMs* whereas some others (*MaxDur*, *MaxRR*, *MinRR*, *MinCRR*, *MaxCSuc*) displayed both positive and negative values depending on the chosen *RM*. Hence, the null hypothesis could not be disproved for all combinations. The remaining rules gave positive values of P with all *RMs*. With these rules the minimum value of P was 0% whereas the average and the maximum values were, respectively, 18% and 36%. Table 5 depicts the results of the best performing rules coupled with the best *RMs*. For all combinations of Table 5, the null hypothesis H_0 is rejected with a significance threshold of 0.001. It appears that the benefits of considering robustness are obvious, but less significant than in the case of smaller size projects (series 1). Finally, the average CPU required for phase I, phases I and II combined as well as for the simulation phase are given in Table 6. These times are obtained using a personal computer with an Intel Core 2 processor, a clock speed of 2×1.6 GHz and 1 GB of RAM.

5. Conclusion

This paper addressed the development of a simple priority-rule-based two-stage approach to solve RCPSP for minimum project duration then for maximum quality-robustness. It also proposed some efficient robustness "predictive" indicators intended to assist the project managers in selecting the most robust schedule among a set of schedules sharing the same makespan. Such a schedule is the more likely to be the best in facing small disruptions. Finally, this work demonstrated, through extensive simulation runs on a set of randomly

Table 5
Advantage of using robust schedules (problems from J60)

RM	*PR*	P_2	σ_2	P_1	σ_1	P	σ_{P1-P2}	z
RM_1	*MinSucc*	0.33	0.0047	0.13	0.0034	0.20	0.0060	33.605
	LFT	0.39	0.0049	0.10	0.0030	0.29	0.0061	47.679
	MinRPW	0.35	0.0048	0.12	0.0032	0.23	0.0060	38.357
RM_2	*MinSucc*	0.33	0.0047	0.14	0.0035	0.19	0.0060	31.686
	LFT	0.41	0.0049	0.09	0.0029	0.32	0.0061	52.256
	MinRPW	0.35	0.0048	0.13	0.0034	0.22	0.0060	36.425
RM_5	*MinSucc*	0.35	0.0048	0.10	0.0030	0.25	0.0059	42.333
	LFT	0.38	0.0049	0.09	0.0029	0.29	0.0060	48.364
	MinRPW	0.39	0.0049	0.11	0.0031	0.28	0.0061	45.724
RM_7	*MinSucc*	0.32	0.0047	0.08	0.0027	0.24	0.0057	42.426
	LFT	0.42	0.0049	0.06	0.0024	0.36	0.0060	59.604
	MinRPW	0.37	0.0048	0.13	0.0034	0.24	0.0061	39.192

P_2: Proportion of iterations in which schedule S_2 outperforms schedule S_1 (over a 10,000 iterations sample).
σ_{P2}: Standard deviation corresponding to P_2.
P_1: Proportion of iterations in which schedule S_1 outperforms schedule S_2 (over a 10,000 iterations sample).
σ_{P1}: Standard deviation corresponding to P_1.
P: Net advantage of S_2 over S_1 ($P = P_2 - P_1$).
σ_{P2-P1}: Standard deviation of the proportion difference.
z: Standard normal decision variable.

Table 6
Average CPU time per instance (ms)

	J30		J60	
	Average	Sd.	Average	Sd.
Phase I	0.20	0.05	0.66	0.22
Phase I + phase II	0.39	0.33	1.24	0.11
Simulation (with 12 *RMs*)	0.77	0.13	2.19	0.36

benchmark problems, the benefits of picking a robust schedule over a non-robust one having the same makespan. As for 10 problems from the standard set J30, both robust and non-robust schedules were executed with a 10% duration increase applied to the same randomly picked 20% of the project activities. Over 1000 iterations per instance problem, the robust schedules displayed a shorter makespan in 55% of the times whereas the non-robust schedules were the best performing ones in only 6% of the times.

As directions for future work, it is proposed to explore other performance measures for assessing the benefits of considering robustness. This is actually in progress. It would also be interesting to search for better performing *PRs* and *RMs*. In addition, it is planned to explore some typical disruption schemes characterizing particular application fields. Finally, the approach could be extended for disruptions caused by resource unavailability.

References

Abbasi, B., Shadrokh, S., & Arkat, J. (2006). Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation, 180*(1), 146–152.

Al Fawzan, M., & Haouari, M. (2005). A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics, 96*, 175–187.

Aloulou, M. A., Portmann, M. -C, & Vignier, A. (2002). Predictive reactive scheduling for the single machine problem. In *Proceedings of the 8th Workshop on Project Management and Scheduling, Valencia, Spain*.

Artigues, C., & Roubellat, F. (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research, 127*, 297–316.

Calhoun, K. M., Deckro, R. F., Moore, J. T., Chrissis, J. W., & Van Hove, J. C. (2002). Planning and re-planning in project and production planning. *Omega, 30*, 155–170.

Cavalcante, C. B. C., Cavalcante, V. C., Ribeiro, C. C., & De Souza, C. C. (2000). Parallel cooperative approaches for the labor constrained scheduling problem. Preprint, Instituto de Computaçao, UNICAMP, Campinas, Brazil, 2000.

Demeulemeester, E., & Herroelen, W. (2002). *Project scheduling: A research handbook*. Boston: Kluwer Academic Publishers.

Fernandez, A. A., Armacost, R. L., & Pet-Edwards, J. (1996). The role of the non-anticipativity constraint in commercial software for stochastic project scheduling. *Computers and Industrial Engineering, 31*, 233–236.

Goldratt, E. M. (1997). *Critical chain*. Barrington: The North River Press Publishing corporation.

Hapke, M., & Slowinski, R. (2000). Fuzzy set approach to multi objective and multi-mode project scheduling under uncertainty. In R. Slowinski & M. Hapke (Eds.), *Scheduling under fuzziness* (pp. 197–221). Heidelberg: Physica-Verlag.

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

Herroelen, W., & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management, 19*, 559–577.

Herroelen, W., & Leus, R. (2004a). The construction of stable project baseline schedules. *European Journal of Operational Research, 56*, 550–565.

Herroelen, W., & Leus, R. (2004b). Robust and reactive project scheduling: A review and classification of procedures. *International Journal of Production Research, 42*(8), 1599–1620.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty, survey and research potentials. *European Journal of Operational Research, 165*(2), 289–306.

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

Khemekhem, M. A., & Chtourou, H. (2006). Assessing the effects of several parameters of an enhanced multi-pass algorithm for the RCPSP. In *Proceedings of the International Conference on Service Systems and Service Management (IEEE ICSSSM'06), 25–27 October, France* (pp. 1223–1227).

Klein, R., & Scholl, A. (1999). Computing lower bounds by destructive improvement: An application to Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research, 112*, 322–346.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz (Ed.). *Project scheduling recent models, algorithm sand applications* (pp. 147–178). Kluwer Academic.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*, 23–37.

Kolisch, R., Schwindt, C., & Sprecher, A. (1998). Benchmark instances for project scheduling problems. In J. Weglarz (Ed.), *Handbook on recent advances in project scheduling*. Dordrecht: Kluwer.

Leus, R. (2003). The generation of stable project plans. Ph.D. Thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Leus, R., & Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE Transactions, 36*, 667–682.

Mauguiere, P., Billaut, J. -C., & Artigues, C. (2002). Grouping jobs on a single machine with heads and tails to represent a family of dominant schedules. In *Proceedings of the 8th Workshop on Project Management and Scheduling, Valencia, Spain*.

Möhring, R. H., Radermacher, F. J., & Weiss, G. (1984). Stochastic scheduling problems. I. General strategies. *ZOR—Zeitschrift fur Operations Research, 28*, 193–260.

Möhring, R. H., Schulz, A. S., Stork, F., & Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science, 49*, 330–350.

Pet-Edwards, J., Selim, B., Armacost, R. L., & Fernandez, A. (1998). Minimizing risk in stochastic resource-constrained project scheduling. In *Proceedings of INFORMS Fall Meeting, Seattle, USA*.

Tabachnik, B., & Fidell, L. (1989). *Using multivariate statistics* (2nd ed.). Harper Collins Publishers.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics, 97*, 227–240.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research, 44*(2), 215–236.

Wang, J. (2004). A fuzzy robust scheduling approach for product development projects. *European Journal of Operational Research, 152*, 180–194.

Yang, K. (1998). A comparison of dispatching rules for executing a resource-constraint project with estimated activity durations. *Omega, 26*(6), 729–738.