Open access • Journal Article • DOI:10.1007/S10951-007-0011-2

A classification of predictive-reactive project scheduling procedures

Stijn Van de Vonder, Erik Demeulemeester, Willy Herroelen

Institutions: Katholieke Universiteit Leuven

Published on: 01 Jun 2007 - Journal of Scheduling (Kluwer Academic Publishers-Plenum Publishers)

Topics: Schedule (project management), Project planning, Dynamic priority scheduling, Two-level scheduling and Rate-monotonic scheduling

J Sched (2007) 10: 195–207
DOI 10.1007/s10951-007-0011-2

A classification of predictive-reactive project scheduling procedures

Stijn Van de Vonder · Erik Demeulemeester · Willy Herroelen

Published online: 15 May 2007

Abstract The vast majority of the project scheduling research efforts over the past several years have concentrated on the development of workable predictive baseline schedules, assuming complete information and a static and deterministic environment. During execution, however, a project may be subject to numerous schedule disruptions. Proactive-reactive project scheduling procedures try to cope with these disruptions through the combination of a proactive scheduling procedure for generating predictive baseline schedules that are hopefully robust in that they incorporate safety time to absorb anticipated disruptions with a reactive procedure that is invoked when a schedule breakage occurs during project execution.

In this paper we discuss the results obtained by a large experimental design set up to evaluate several predictive-reactive resource-constrained project scheduling procedures under the composite objective of maximizing both the schedule stability and the timely project completion probability.

Keywords Proactive-reactive project scheduling · Time uncertainty · Stability · Timely project completion

S. Van de Vonder · E. Demeulemeester (✉) · W. Herroelen
Research Center for Operations Management, K.U.Leuven, Leuven, Belgium
e-mail: erik.demeulemeester@econ.kuleuven.be

S. Van de Vonder
e-mail: stijn.vandevonder@econ.kuleuven.be

W. Herroelen
e-mail: willy.herroelen@econ.kuleuven.be

1 Introduction

The vast majority of the research efforts in project scheduling over the past several years have concentrated on the development of exact and heuristic procedures for the generation of a workable *baseline schedule* (*pre-schedule or predictive schedule*), assuming complete information and a static and deterministic environment. This baseline schedule serves a number of important functions (Aytug et al. 2005; Mehta and Uzsoy 1998), such as facilitating resource allocation, providing a basis for planning external activities (i.e., activities to be performed by subcontractors) and visualizing future work for employees. Pre-schedules are the starting point for communication and coordination with external entities in the company's inbound and outbound supply chain: they are the basis for agreements with suppliers and subcontractors, as well as for commitments to customers.

During execution, however, a project may be subject to considerable uncertainty which may lead to numerous schedule disruptions. Many types of disruptions have been identified in the literature (we refer to Zhu et al. 2005; Wang 2005). Activities can take longer than primarily expected, resource requirements or availabilities may vary, ready times and due dates may change, new activities might have to be inserted (Artigues and Roubellat 2000), etc. We limit ourselves to the treatment of *time uncertainties* caused by the fact that actually realized activity durations during project execution may deviate from the durations that were planned in the baseline schedule.

In general, there are two approaches to dealing with uncertainty in a scheduling environment (Davenport and Beck 2002; Herroelen and Leus 2005): proactive and reactive scheduling. *Proactive scheduling* constructs a predictive schedule that accounts for statistical knowledge of uncertainty. The consideration of uncertainty information is used

to make the predictive schedule more *robust*, i.e., insensitive to disruptions. *Reactive scheduling* involves revising or re-optimizing a schedule when an unexpected event occurs. At one extreme, reactive scheduling may not be based on a predictive schedule at all: allocation and scheduling decisions take place dynamically in order to account for disruptions as they occur. A less extreme approach is to reschedule when schedule breakage occurs, either by completely regenerating a new schedule or by repairing an existing predictive schedule to take into account the current state of the system. It should be observed that a proactive technique will always require a reactive component to deal with schedule disruptions that cannot be absorbed by the baseline schedule. The number of interventions of the reactive component is inversely proportional to the robustness of the predictive baseline schedule.

The basic scheduling problem in a deterministic project setting is the so-called *resource-constrained project scheduling problem* (RCPSP). This problem (problem $m,1|cpm|C_{\max}$ in the notation of Herroelen et al. 2000) involves the determination of a baseline schedule that satisfies both the finish-start, zero-lag precedence constraints between the activities and the renewable resource constraints under the objective of minimizing the project duration. Developing algorithms to solve this problem dominates the project scheduling literature (for overviews: Herroelen et al. 1998; Kolisch and Padman 1999; Kolisch and Hartmann 1999; Brucker et al. 1999; Demeulemeester and Herroelen 2002).

A recent research track focuses on the *stochastic resource-constrained project scheduling problem*, an extension of the RCPSP that involves the minimization of the expected makespan of a project with stochastic activity durations (problem $m,1|cpm,\mathbf{d}_j|E(C_{\max})$). The stochastic RCPSP aims at making a project *quality robust*, i.e., insensitive to disruptions that affect the performance metrics used to evaluate its quality. In this paper quality robustness refers to makespan performance. Most of the research efforts on the stochastic RCPSP rely on so-called *scheduling policies* (Möhring et al. 1984, 1985). These policies do not use a baseline schedule but view the scheduling problem as a multi-stage decision process where decisions on the set of activities to be started next have to be made at stochastic decision points that correspond to the completion time of activities, exploiting only knowledge about the observed past and the a priori knowledge about the processing time distributions. Stork (2001) examines the performance of different classes of policies. In the proactive-reactive terminology of this paper, these scheduling policies can be viewed as quality robust reactive procedures.

Besides minimizing the expected makespan $E(C_{\max})$, the *service level* of a project can be regarded as a practical quality robustness measure that maximizes $P(\mathbf{z} \leq z)$, the probability that the objective function value $\mathbf{z}$ of the realized schedule stays within a certain threshold z. For the makespan objective, we maximize the probability that the project completion time does not exceed the predefined project due date δ_n, i.e., $P(\mathbf{s}_n \leq \delta_n)$, where $\mathbf{s}_n$ denotes the start time of the dummy end activity. We will refer to this measure as the *timely project completion probability* or *TPCP*.

Constructing a baseline schedule with good makespan performance for a wide range of execution scenarios corresponds to building a *quality robust* schedule. Research efforts on proactive quality robust project scheduling are rather scarce. The well-known critical chain buffer management approach of Goldratt (1997) might be considered as a quality robust baseline scheduling method in that the inserted buffers aim at protecting the project due date.

However, when uncertainties come into play, optimizing the makespan performance is no longer the whole story. Project plans should also include some *stability* or *solution robustness*, i.e., the insensitivity of planned activity start times to schedule disruptions. The a priori predictive schedule should not differ too much from the realized schedule obtained after project execution. Constant rescheduling in order to improve the expected makespan might strongly decrease the predictive value of the baseline schedule. For this reason, we aim at minimizing a stability cost function $\sum w_j E|\mathbf{s}_j - s_j|$, defined as the weighted sum of the expected absolute deviation (E is the expectation operator) between the actually realized activity start times $\mathbf{s}_j$ and the planned activity start times s_j. Leus (2003) suggests this objective function to solve the *NP*-hard robust resource allocation problem in exact and approximate formulations, while Van de Vonder et al. (2005) use it to introduce a heuristic buffer allocation procedure aiming at solution robustness to investigate the trade-off between makespan performance and stability. The activity-dependent weights w_j that are used in the stability cost function represent the marginal cost of starting the activity later or earlier than planned in the baseline schedule. They may include unforeseen storage costs, extra organizational costs, costs related to agreements with subcontractors or just a cost that expresses the dissatisfaction of employees with schedule changes.

The problem used as vehicle of analysis in this paper is a variant of the stochastic RCPSP. The project has a single zero-duration dummy start node 0 and a single zero-duration dummy end node n. Project activities j ($j = 1, 2, \ldots, n-1$) have stochastic activity durations $\mathbf{d}_j$, are subject to finish-start zero-lag precedence constraints and require an integer per period amount r_{jk} of one or more renewable resource types k ($k = 1, 2, \ldots, K$) during their execution. The renewable resources have a constant per period availability a_k. During project execution the actually realized start time $\mathbf{s}_j$ of activity j ($j = 1, 2, \ldots, n-1$) may be smaller than, equal to or larger than its scheduled activity start time s_j. We assume that $\mathbf{s}_0 = s_0$ and that $s_n = \delta_n$. The scheduled project

completion time s_n is thus set equal to a predefined deterministic project due date δ_n. We assume that the project is not penalized when finished early. This boils down to assuming that the realized project completion time $\mathbf{s}_n$ is set equal to δ_n if the originally obtained $\mathbf{s}_n$ is smaller than δ_n. The project has to be scheduled under the *composite objective* of maximizing the timely project completion probability (TPCP) and minimizing the stability cost function $\sum w_j E|\mathbf{s}_j - s_j|$. Relaxing the assumption that the earliness costs equal the tardiness costs would incur minor changes to the procedures proposed in this paper and would not affect the validity of our research.

It is to be expected that it will be very difficult to find a schedule that achieves the optimal value for both the timely project completion probability and stability performance criteria simultaneously. As no criterion is dominant from the outset, we opt for *a posteriori optimization* (Hoogeveen 2005). Following the notation of Herroelen et al. (2000), the problem can be classified as $m, 1|cpm, \mathbf{d}_j, \delta_n|F(P(\mathbf{s}_n \leq \delta_n), \sum w_j E|\mathbf{s}_j - s_j|)$. The function $F(\cdot,\cdot)$ is a composite objective function that is not known a priori and where the relative importance of the two criteria is not specified from the outset and no clear linear combination is known that would reflect the preference of the decision maker.

We tackle the problem in a roundabout way. Solutions to the problem will be obtained through the application of different predictive-reactive scheduling procedures. The analytic evaluation of the composite objective function is very cumbersome (the PERT problem is ♯P complete (Hagstrom 1988) and the scheduling problem for stability has been shown to be *NP*-hard in the ordinary sense by Leus and Herroelen 2005). Therefore, the composite objective function values will be evaluated through simulation. In the simulation we make the assumption that at any given time instant we not only know the real activity duration of the activities finished by that time but also the remaining duration of the activities that did not finish but should have finished by that time, given their realized start times and expected duration.

The simulation runs provide insight into the overall performance of the tested predictive-reactive scheduling procedures with respect to the composite objective function used. The main objective of this paper is to identify the conditions under which proactive scheduling pays off and the conditions that favor reactive scheduling when the uncertainty of a project resides in the activity durations. The simulation results also permit to examine the potential trade-off between quality and solution robustness. They will also enable an analysis of the impact of (a) the level of variability in the activity durations, (b) the relative weights of the project activities, and (c) the tightness of the project due date.

The remainder of the paper is organized as follows. Section 2 introduces a number of baseline scheduling algorithms that differ in the complexity of the scheduling methodology used and in their degree of proactiveness, i.e., the amount of safety included. The reactive scheduling procedures are the subject of Sect. 3. They can be classified as either quality or solution robust. Section 4 is devoted to a description of the experimental set-up used in the computational experiment. The computational results are described in Sect. 5. A last section provides some overall conclusions.

2 The baseline scheduling methods

Three procedures for generating a baseline schedule will be introduced in this section: an exact procedure for generating quality robust schedules (Sect. 2.1), a suboptimal quality robust scheduling procedure (Sect. 2.2), and a suboptimal procedure for generating solution robust schedules (Sect. 2.3). To the best of our knowledge, the literature on exact solution robust project scheduling procedures is still void.

The procedures will be illustrated on the simple project instance of Fig. 1. This project network is a 6-activity, zero-lag, finish-start activity-on-the-node network with one renewable resource with a constant per period availability of 10 units and resource requirements identified for each activity by the number shown to the right of the corresponding node. The number shown to the left of each node denotes the expected duration of the corresponding activity. The project due date is arbitrarily set to $\delta_5 = 10$.

2.1 An exact procedure for generating quality robust baseline schedules

Any solution procedure for the deterministic resource-constrained project scheduling problem (problem $m, 1|cpm|C_{\max}$) may be used to generate a predictive schedule. Using an exact procedure and mean activity durations, we opt for minimizing the project duration in the hope that this provides a quality robust baseline schedule, i.e., a schedule that maximizes $P(\mathbf{s}_n \leq \delta_n)$, the probability that the project ends within a given project due date. Many exact procedures have been proposed in the open literature. In this paper we

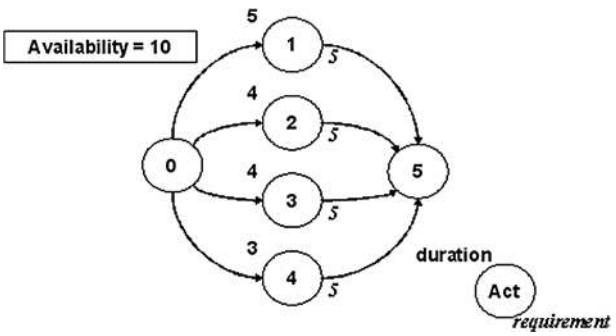

Fig. 1 An example network

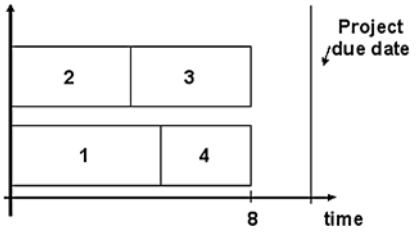

Fig. 2 A minimal duration quality robust baseline schedule for the example network of Fig. 1

use the branch & bound procedure of Demeulemeester and Herroelen (1992, 1997). Application of this procedure to the problem example of Fig. 1 yields the minimum duration baseline schedule of Fig. 2. The procedure yields a deterministic makespan of 8 periods, 20% shorter than the project due date $\delta_5 = 10$. The two-period time interval between the minimum project completion time and the due date acts as a protective time cushion during project execution, inducing quality robustness into the schedule.

2.2 A suboptimal procedure for generating quality robust baseline schedules

Because the RCPSP is known to be ordinary NP-hard (Blazewicz et al. 1983), solving it to optimality can induce a large computational effort. Most commercial project scheduling software packages do not aim at schedule stability and rely on simple priority-based scheduling heuristics to solve the RCPSP. The *Late Start Time* (LST) priority rule (Davis and Patterson 1975) has been shown to obtain a good makespan performance among several examined priority rules in an experimental study by Kolisch (1996). We implement the LST rule using a simple serial scheduling generation scheme. The project activities are entered in a precedence feasible list in ascending order of their critical path based latest allowable start times. The list is scanned and the activities are scheduled at their earliest precedence and resource feasible start time. For the example project of Fig. 1, the serial LST uses the list $L = (0, 1, 2, 3, 4, 5)$ and, a lucky coincidence, generates the minimum duration schedule of Fig. 2.

2.3 A suboptimal solution robust baseline schedule

The last baseline scheduling procedure that we consider does not try to minimize the project completion time but aims at minimizing the stability cost function $\sum w_j E|s_j - s_j|$, the sum of the weighted expected absolute deviations between realized and planned activity start times. The literature on this ordinary NP-hard scheduling problem is virtually void. We use the suboptimal *resource flow dependent float factor* (RFDFF) heuristic that has been developed by Van de Vonder et al. (2006) as an extension to the *activity dependent float factor* (ADFF) heuristic proposed in Leus (2003) and Van de Vonder et al. (2005). RFDFF starts from a feasible RCPSP solution (in this paper this is the minimum duration schedule obtained by the branch-and-bound procedure as described in Sect. 2.1) and changes it by adding safety buffers in front of activities in order to make the schedule solution robust. The start time of activity j in the RFDFF schedule is calculated as $s_j(S) := s_j(B\&B) + \alpha_j(float(j))$, where $s_j(B\&B)$ denotes the start time of activity j in the optimal minimum duration baseline schedule of Sect. 2.1. This start time is augmented by some safety time $(\alpha_j(float(j)))$ as a time buffer to enhance stability. The total float "$float(j)$" in this expression is the difference between the latest allowable start time of activity j given the project due date (i.e., its start time in the right-justified version of the minimum duration baseline schedule) and its scheduled start time in the minimum duration schedule.

To calculate the float factors α_j we first need to construct a resource flow network (Artigues and Roubellat 2000) for the minimum duration schedule. The flow network is a network with the same nodes as the original project network, but with arcs connecting two activities if there is a resource flow between these activities. It thus identifies how each single item of a resource is passed on between the activities in the schedule. We use the single-pass algorithm of Artigues and Roubellat (2000) to construct a feasible resource flow network. In our simple example network of Fig. 1 the non-dummy activities were not precedence related. The minimum duration schedule of Fig. 2 specifies that 5 resource units have to be transferred from activity 1 to activity 4 and 5 resource units have to be transferred from activity 2 to activity 3. This resource allocation induces a resource flow network that is identical to the network of Fig. 1, except for the two extra precedence arcs from activity 2 to activity 3 and from activity 1 to activity 4. Note that in this simple example the resource flow network is unique because the schedule only allows for a unique way in which the resources may be passed among the activities. In general, the same schedule may allow for different ways of allocating the resources so that the same schedule may be represented by different resource flow networks.

The float factors α_j are now calculated as follows: $\alpha_j = \beta_j/(\beta_j + \lambda_j)$, where β_j is the sum of the weight of activity j and the weights of all transitive predecessors of activity j in as well the original network as the resource flow network, while λ_j is the sum of the weights of all transitive successors of activity j in both networks. The weights of activities that start at time 0 are not included in these summations because these activities can always start at their planned start time and thus do not need any buffering to cope with possible disruptions of their predecessors. The RFDFF heuristic

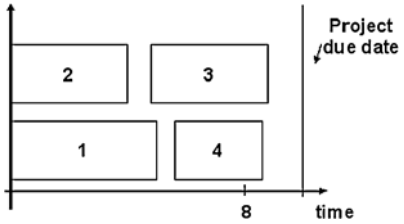

Fig. 3 RFDFF schedule for the example network of Fig. 1

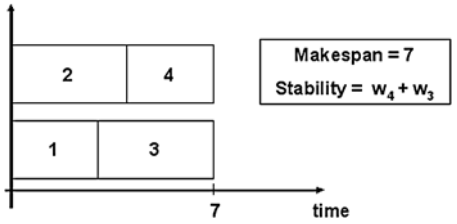

Fig. 4 Full rescheduling by using the original baseline scheduling method

consequently inserts longer time buffers in front of activities that would incur a high cost if started earlier or later than originally planned and resource constraints will always remain satisfied in the resulting schedule.

For our small example network, the RFDFF schedule could be (dependent on the activity weights) as proposed in Fig. 3 which reveals the time buffers in front of activities 3 and 4 that can absorb small disruptions of the activities' predecessors in the resource flow network.

3 The reactive scheduling methods

In the previous section we discussed baseline scheduling methods with a different ability to absorb disruptions. However, it is to be expected that none of them will ever be stable enough to anticipate all possible disruptions that may occur during project execution. A proactive scheduling procedure must, therefore, be combined with a reactive scheduling procedure that allows to react during schedule execution on schedule disturbances that cannot be absorbed by the proactive schedule. In this section several possible reactive procedures are defined. In fact, all procedures can again be classified as optimal quality robust (Sect. 3.1), heuristic quality robust (Sects. 3.2 and 3.3) or solution robust (Sect. 3.4). The solution robust reactive procedure of Sect. 3.4 can be considered as optimal because the schedule is reoptimized at any decision point by fully using all information available at that time. The development and investigation of heuristic solution robust reactive procedures is subject to future research.

In order to illustrate the reactive procedures, we start from the quality robust schedule of Fig. 2 with a project due date δ_5 equal to 10 and we explain how each procedure would react on a schedule disruption caused by the fact that upon schedule execution the actual duration of activity 1 drops from the expected 5 time units to 3 time units.

3.1 Complete rescheduling by solving a deterministic RCPSP

The first reactive procedure studied in this paper is to completely regenerate a new up-to-date schedule when schedule breakage occurs. Rescheduling is done by applying the scheduling algorithm that was used to generate the baseline schedule but now on a modified project network. Activities that are already finished at the schedule breakage point are omitted from the original project network. Activities that have already started but did not yet finish by the schedule breakage point are kept in the network with the projected remainder of the activity duration as their planned duration. As stated above, we indeed make the assumption in this paper that when an activity was planned to have finished by a particular time instant, but has not, the remaining duration is known.

In order to illustrate the procedure, let us assume that the baseline schedule of Fig. 2 is disrupted due to the fact that activity 1 finishes two time units earlier than planned, i.e., activity 1 finishes at time instant 3 instead of time instant 5. At time instant 3, activity 2 is still in progress but has not yet finished. As activity 2 has been in execution for 3 time periods and was planned to finish only at time instant 4, its remaining duration is set to 1 period. Activity 1 is removed from the original project network and activity 2 is kept with its remaining duration of 1 period. The RCPSP is set up for this new project network. Solving the RCPSP for this new project using the branch-and-bound procedure yields the updated projected schedule of Fig. 4. Observe that the original resource flows are broken. Activity 3 does not have to wait any longer for the resources to be released by activity 2, as was the case in Fig. 2, but may now receive its required resource units from activity 1 upon the completion of this activity at time instant 3. When complete rescheduling is used, it may be necessary to calculate a completely new resource flow network.

In principle, as is the case here, the application of full rescheduling may be capable of maintaining solutions with minimum makespan. Alongside the high computational effort required, the main disadvantage of this procedure is that the stability cost might be extremely high, because the resulting schedule can differ completely from the original baseline schedule. In the example of Fig. 4, activities 3 and 4 both start one time unit earlier than planned, leaving the total stability cost at $w_3 + w_4$.

It should be noted that when the baseline schedule is constructed using a solution robust procedure such as RFDFF, complete rescheduling using the same solution robust base-

line scheduling procedure does not make any sense and this reactive option will not be investigated.

3.2 Early start policy after fixing the resource flows

The second reactive scheduling procedure studied in this paper applies an early start policy at the schedule breakage point, while maintaining the resource allocation decisions made in the baseline schedule, as reflected in its resource flow network. As activity duration disruptions do not change the resource requirements of any activity, maintaining the resource flow network that was constructed for the baseline schedule and applying an early start policy upon schedule disruption will yield a precedence and resource feasible schedule. This schedule allows every activity to start as soon as all its precedence and resource-based predecessors in the resource flow network have finished. Fixing the resource allocation reduces the rescheduling flexibility during project execution. We will investigate the impact of this decreased flexibility on makespan performance and stability cost in Sect. 5.

When the baseline schedule is constructed using the procedure described in Sect. 2.3, we apply *railway scheduling*, i.e., activities will never start earlier than their planned start time in the baseline schedule. A solution robust baseline schedule requires railway scheduling during execution in order to preserve the stability advantage of the idle times in the schedule. The computational requirements for this reactive procedure are very low. Disruptions do not require complete rescheduling nor building a new resource flow network.

For the disruption scenario used in our problem example, the reactive policy must be applied at time instant 3 when activity 1 finishes. Keeping the resource flow network fixed means that we maintain the resource-based precedence relations between activity 2 and 3 and between activity 1 and 4, respectively. Applying the early start policy now allows to start activity 4 at time 3. The resulting *projected schedule*[1] is shown in Fig. 5. We note that the projected makespan at this time is not changed by the disruption (it remains at the originally planned 8 periods) and that the total stability cost is projected as $2 \times w_4$ because activity 4 starts two time units earlier than planned and all other activities can start at their planned start time.

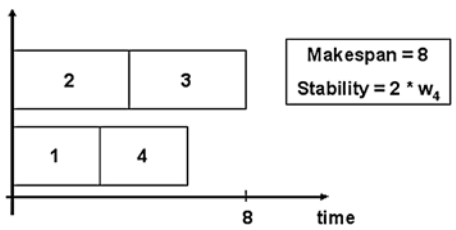

Fig. 5 Fixing the flow network and calculating the early start schedule when $d_1 = 3$

3.3 Activity-based priority rules

We stated in the introduction that a large part of the stochastic RCPSP literature has concentrated on the development of scheduling policies to decide which activity to schedule next at stochastic decision points corresponding to the completion of activities. Many of these policies depend upon the (minimal) forbidden set[2] concept introduced by Igelmund and Radermacher (1983a, 1983b) and might become computationally extensive. The activity-based priority policies, however, are the most efficient among the examined policies and have shown to yield a good heuristic solution without requiring the enumeration of the forbidden sets. Solutions to the RCPSP are not represented by a schedule but by an activity list. This activity list is an ordering of all activities that can be transformed into a schedule at all times by applying a schedule generation scheme on the activity list.

In our research, we deduce the activity list from the baseline schedule by ordering the activities in increasing order of their scheduled start time. The example schedule of Fig. 2 can be represented by the activity lists $(1, 2, 3, 4)$ and $(2, 1, 3, 4)$. This directly shows that a schedule might be represented by multiple activity lists. We randomly choose one activity list, for example $(1, 2, 3, 4)$. The scheduling generation scheme (SGS) used to build a schedule from his activity list is the one used by Stork (2001). Upon schedule breakage, the activity list is scanned and we try to schedule all activities that have not yet been started as soon as possible after the current decision point (we must make sure that we don't allow an activity to be scheduled in the past).

For the solution robust baseline schedule of Sect. 2.3, again railway scheduling will be applied to preserve good stability. When starting from a quality robust baseline schedule, adding railway scheduling to a reactive procedure might be considered as an easy adaptation to make a procedure more solution robust. We will use the priority rule based procedure of this section extended with railway scheduling (this procedure will be named *Railway*) as sub-optimal solution robust reactive procedure.

The reactive procedure introduced in this section (without railway scheduling) would react as follows on the example disruption scenario used in our problem example. The completion of activity 1 at time 3 introduces the first decision point at which the reactive policy has to decide which

[1] A *projected schedule* is a schedule that at every point in time describes how the project will execute if no further disruptions occur, given all information available at that time. It must be constantly updated.

[2] A *forbidden set* is a set of activities that are not precedence related, but concurrently scheduling all activities of this set would result in a resource constraint violation.

activity should be scheduled next. The next activity in the list $(1, 2, 3, 4)$, activity 2, has already started and should stay in progress because no preemption is allowed. Activity 3 is the first unscheduled activity in the activity list. After checking the precedence and resource constraints, we may conclude that this activity can start at time 3. The last activity from the list, activity 4, then starts at time 4, resulting in the projected schedule shown in Fig. 4 with a projected makespan of 7 time units.

3.4 Minimizing earliness-tardiness costs

The reactive scheduling procedures described in the previous sections do not rely on stability issues to take corrective action. A reactive scheduling procedure that focuses on solution robustness can be based on exact procedures for the *resource-constrained earliness-tardiness project scheduling problem* (RCPSPWET). This problem, classified as $m, 1|cpm|early/tardy$ in the classification scheme of Herroelen et al. (2000), allows for the specification of activity due dates δ_j and associated unit earliness and unit tardiness penalty costs. The objective then is to schedule the project activities subject to the precedence and renewable resource constraints in order to minimize the weighted earliness/tardiness penalty cost of the project.

Vanhoucke et al. (2001) have developed an effective and efficient exact solution procedure for the RCPSPWET that may be turned into a reactive scheduling procedure in the following way. The due date for an activity is set to its projected finish time in the baseline schedule. For the dummy end activity, the due date thus becomes the project due date δ_n. The unit earliness and tardiness costs of a non-dummy activity j are assumed to be identical and are set equal to the activity weight w_j. The earliness cost of the dummy end activity is set to zero, because we do not punish the project for finishing earlier than planned, while the tardiness cost again equals the weight w_n. By invoking this exact RCPSPWET algorithm upon schedule breakage, we make sure that at each rescheduling point a new projected schedule is constructed with the lowest stability cost (in terms of deviation from the original baseline schedule).

For our previously mentioned example, rescheduling by solving the RCPSPWET results in the schedule of Fig. 6.

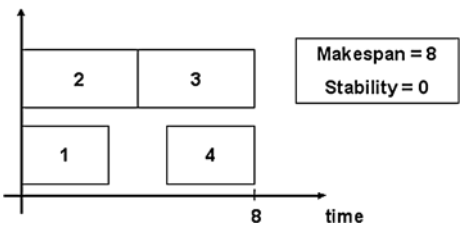

Fig. 6 Rescheduling by solving the RCPSPWET

We note that the stability cost drops to zero because all activities keep their originally scheduled start time. The projected makespan equals the baseline schedule duration in this example, but it should be understood that in a larger project this reactive scheduling procedure might deteriorate makespan performance because some activities are started beyond their earliest possible start time, holding the risk that the project may have to be delayed by disruptions that may occur later during project execution.

4 Experimental set-up

The predictive and reactive scheduling procedures were coded in Microsoft Visual C++. A problem test set was constructed using the RanGen project scheduling network instances generator developed by Demeulemeester et al. (2003). Eighty 30-activity networks were generated using different settings for the order strength, resource factor and resource constrainedness. *Order strength* (*OS*) (Mastor 1970) defines the density of the network by dividing the number of precedence relations in the network, including the transitive ones, by the theoretical maximum number. The *resource factor* (*RF*) and the *resource constrainedness* (*RC*) describe the resource usage of all activities. *RF* (Pascoe 1966) reflects the average number of resource types used by an activity, while *RC* (Patterson 1976) defines the average portion of the resource availability that is used by an activity that uses a certain resource. For all three project characteristics, two values are used to build the data set, namely 0.5 and 0.75. For each of the 2^3 parameter combinations, 10 network instances were generated, yielding a total of 80 test instances. For a detailed study of the impact of these network parameters on makespan performance and stability, we refer to Van de Vonder et al. (2006).

In our computational experiment we will study the impact of three parameters (see Table 1): the level of uncertainty in activity durations, the weighting parameter and the project due date. Activity duration uncertainty is set to two possible values. *High duration variability* means that the real activity duration is a discretized value drawn from a right-skewed beta-distribution with parameters 2 and 5 that is transformed in such a way that the minimum duration equals half the expected duration, the mean duration equals the expected duration and the maximum duration equals

Table 1 Experimental parameter settings

Parameter	Setting 1	Setting 2
Variability in d_j	low	high
$w_n = wp \times w_{\text{avg}}$	$wp = 5$	$wp = 10$
Project due date δ_n	$115\% \times C_{\max}$	$130\% \times C_{\max}$

2.25 times the expected duration. *Low duration variability* means that the real activity durations are also discretizations of values drawn from a beta-distribution with parameters 2 and 5, with the mean equal to the expected activity duration, but with minimum and maximum values equal to, respectively, 0.75 times and 1.625 times the expected activity duration. This setting allows for fewer activities to be disrupted (due to the discretization) with a smaller average disruption size.

The *weighting parameter* (*wp*) is defined as the ratio between the weight of the dummy end activity and the average of the distribution of the weights of the other activities ($wp = w_n/w_{\text{avg}}$). Van de Vonder et al. (2005, 2006) use the *wp* to study the impact of the weight (importance) attributed to the dummy end activity on the trade-off between stability and makespan performance. The activity weights w_j of the non-dummy activities $j \in \{1, 2, \ldots, n-1\}$ are drawn from a discrete triangular distribution with $P(w_j = q) = (21-2q)\%$ for $q \in \{1, 2 \ldots 10\}$. This distribution results in a higher probability for low weights and in an average weight $w_{\text{avg}} = 3.85$ that is used to calculate $w_n = wp \times w_{\text{avg}}$. In this paper two values for *wp* ($wp = 5$ and $wp = 10$) are examined.

We use two possible settings for the *project due date*: a 15% and a 30% increase above the minimum baseline schedule duration ($C_{\max}$) obtained using an exact solution procedure for solving the RCPSP with mean activity durations.

5 Computational results

In this section we discuss the computational results obtained by combining the predictive and reactive scheduling procedures of Sects. 2 and 3 into a total of twelve predictive-reactive scheduling procedures. For each of the 80 project network instances and every possible combination of the activity duration uncertainty, weighting parameter and due date settings, 30 project executions are simulated by drawing beta-distributed activity durations as discussed above. For each algorithm, this results in a total of 2400 project executions per parameter setting.

Each row of Table 2 shows the average results obtained by a particular predictive-reactive procedure for the objective functions $\sum w_j E|\mathbf{s}_j - s_j|$ and $P(\mathbf{s}_n \leq \delta_n)$ over the 19 200 network executions ($2^3 \times 2400$) carried out in this experiment. The column with heading *Baseline* identifies the baseline scheduling methods discussed in Sects. 2.1, 2.2 and 2.3 as RCPSP, LST and RFDFF, respectively. The *Reactive* column identifies the complete rescheduling procedures discussed in Sect. 3.1 as RCPSP, while the reactive procedures of Sects. 3.2, 3.3 and 3.4 are identified as Fix Flow, ABP and WET, respectively. The adaptation of ABP that includes railway scheduling to improve solution robustness is identified as Railway.

Table 2 Overall performance values

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	334.46	98.72	0.79	9.65
2	RCPSP	Fix flow	297.03	98.48	1.18	0.08
3	RCPSP	ABP	297.19	98.61	1.03	0.16
4	RCPSP	Railway	257.26	98.21	3.70	0.16
5	RCPSP	WET	235.00	96.90	4.63	353.41
6	LST	Fix flow	335.11	84.71	8.51	0.09
7	LST	ABP	342.52	86.25	8.09	0.16
8	LST	Railway	290.78	80.74	11.50	0.16
9	LST	WET	231.38	82.20	11.86	312.38
10	RFDFF	Fix flow	48.14	88.56	19.39	0.09
11	RFDFF	ABP	47.06	88.69	19.38	0.20
12	RFDFF	WET	45.38	88.68	19.38	9.56

The column with heading *Stability* shows the stability cost $\sum w_j E|\mathbf{s}_j - s_j|$ averaged over all 19 200 network executions. The column with heading *TPCP* shows the average $P(\mathbf{s}_n \leq \delta_n)$. The next column, headed by *dev.*, represents the average deviation of the realized makespan during schedule execution from the project makespan that would be obtained by solving the RCPSP using the actually realized activity durations. It should be observed that, even when complete rescheduling with an exact quality robust reactive policy is applied, this deviation value will not be zero for every schedule execution. This is due to the fact that during schedule execution activities may already have been started when a schedule disruption occurs but cannot be interrupted due to the non-preemption assumption. The last column denotes the average computational time in seconds required per network (for 30 simulated executions).

The results shown in row 1 for the RCPSP-RCPSP procedure demonstrate that the use of an exact procedure for deriving the baseline schedule and performing a complete rescheduling upon schedule breakage excels in makespan performance but is clearly outperformed in terms of stability costs. RCPSP-Fix Flow and RCPSP-ABP are computationally far less demanding and obtain comparable makespan performance at smaller stability cost. The extra computational effort for RCPSP-RCPSP is due to the fact that a new RCPSP must be solved to optimality at every schedule disruption, while fixing the flow network and building an early start schedule (for Fix Flow) or generating a new schedule from an activity list (for ABP) is very easily done. The higher stability cost is due to the fact that complete rescheduling does not look at the previous projected schedule at all upon rescheduling, while combinations RCPSP-Fix Flow and RCPSP-ABP do (partially) maintain the or-

dering of the activities which slightly improves stability. Combining an exact baseline scheduling procedure with a solution robust reactive procedure (combinations RCPSP-Railway and RCPSP-WET) yields acceptable makespan performance at much smaller stability cost. Using WET as a reactive procedure requires a very heavy computational effort due to the need to solve a weighted earliness-tardiness problem to optimality at each schedule breakage point.

Using exact procedures for constructing the baseline schedule pays off. Comparing the results shown in rows 2–4 and rows 6–8 demonstrate that, when a quality robust reactive procedure is used, the makespan and stability performance for the predictive-reactive procedures that rely on an exact procedure for constructing the baseline schedule (rows 2–4) are much better than for the predictive-reactive procedures that generate the baseline schedule using a simple priority-based procedure such as LST (rows 6–8). The combination LST-WET yields a slightly better average stability than RCPSP-WET, but performs much worse on makespan. The improvement in stability is due to the fact that a less tight schedule such as LST includes more flexibility to absorb disruptions during execution. Obviously, this advantage does not compensate for the loss in makespan performance, but it indicates that generating initial schedules with (near) minimum duration, but with more slack included than the current minimum duration schedule, holds promise.

The real advantage of using a heuristic procedure for generating baseline schedules, especially for large real-life projects, is the smaller computational effort in the project scheduling phase. However, for large projects we would advise a more advanced (meta-)heuristic than the simple LST-heuristic.

We note the huge difference in stability costs between the strategies that rely on a quality robust baseline schedule (rows 1–9) and those that start from a solution robust baseline schedule (rows 10–12). The type of reactive scheduling procedure to be used in combination with RFDFF has limited impact on makespan or stability. This is due to the fact that the proactive schedule generated using RFDFF already anticipates many disruptions and because all policies try to preserve the buffers included in the baseline schedule. WET remains, however, the best reactive procedure to maximize the stability in the project.

Using a solution robust proactive procedure clearly outperforms the simple use of a solution robust reactive procedure on stability. However, the obtained TPCP of solution robust scheduling methods (10–12) might be deemed insufficient by project management when no information is available about the project settings, it might be advisable to combine an exact procedure for generating quality robust baseline schedules (RCPSP) with solution robust reactive procedures. This yields reasonable results in terms of both makespan and stability.

Table 3 Performance values for high variability

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	440.69	97.48	1.08	10.03
2	RCPSP	Fix flow	396.32	97.02	1.66	0.08
3	RCPSP	ABP	395.14	97.29	1.43	0.16
4	RCPSP	Railway	347.95	96.48	4.90	0.17
5	RCPSP	WET	312.69	94.31	5.92	502.94
6	LST	Fix flow	445.09	81.83	8.76	0.09
7	LST	ABP	451.78	83.83	8.16	0.16
8	LST	Railway	390.43	76.65	12.61	0.16
9	LST	WET	300.81	77.95	12.83	454.89
10	RFDFF	Fix flow	83.34	80.54	19.56	0.09
11	RFDFF	ABP	81.33	80.77	19.54	0.20
12	RFDFF	WET	78.10	80.75	19.55	18.71

In the remainder of this section, we examine the performance of the predictive-reactive procedures in more detail and study the impact of the different settings for activity duration uncertainty, weighting parameter and project due date.

5.1 High activity duration variability

In this section we analyze the results obtained when the activity duration variability is set to high. The results presented in Table 3 are the averages of four setting combinations (9600 executions) of the three examined parameters (variation, *wp*, project due date), namely (high, 5, +15%), (high, 5, +30%), (high, 10, +15%) and (high, 10, +30%).

We note that high variability increases stability cost and decreases TPCP compared to the overall results of Table 2. The 80% TPCP obtained by the solution robust baseline scheduling method RFDFF (rows 10–12) will most likely be rejected by the planner. Generating minimum duration predictive schedules (rows 1–5) rises the TPCP above a reasonable 94%. The combination RCPSP-Railway (row 4) and RCPSP-WET (row 5) yield acceptable makespan and stability performance. The high computational requirements of WET could drive the planner towards heuristic solution robust procedures. Note that the TPCP-values should be interpreted with care. Remember that TPCP represents $P(\mathbf{s}_n \leq \delta_n)$. However, it might be the case that the disruptions are so heavy that whatever proactive-reactive procedure was used, $\mathbf{s}_n$ will exceed δ_n, i.e., the makespan of the ex-post minimum duration schedule exceeds δ_n. Punishing a procedure for violating this due date would thus be improper. For the high variability case, the simulation results revealed that an unavoidable due date violation occurred in 1.50% of all executions, leaving the maximum obtainable TPCP at 98.50%. Taking this into account, RCPSP-RCPSP only fails in 1% of all executions to complete within time when possible.

Table 4 Performance values for low variability

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	228.23	99.96	0.50	9.27
2	RCPSP	Fix flow	197.74	99.94	0.70	0.09
3	RCPSP	ABP	199.23	99.94	0.63	0.16
4	RCPSP	Railway	166.57	99.94	2.50	0.16
5	RCPSP	WET	157.31	99.49	3.34	203.89
6	LST	Fix flow	225.13	87.59	8.25	0.09
7	LST	ABP	233.27	88.67	8.02	0.16
8	LST	Railway	191.13	84.83	10.38	0.16
9	LST	WET	161.95	86.46	10.89	169.87
10	RFDFF	Fix flow	12.94	96.58	19.21	0.09
11	RFDFF	ABP	12.78	96.62	19.21	0.19
12	RFDFF	WET	12.66	96.62	19.21	0.41

Table 5 Performance values for loose due date settings

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	333.44	99.96	0.79	9.66
2	RCPSP	Fix flow	295.78	99.94	1.18	0.08
3	RCPSP	ABP	296.04	99.94	1.03	0.16
4	RCPSP	Railway	255.83	99.94	3.70	0.16
5	RCPSP	WET	231.68	99.75	4.70	333.42
6	LST	Fix flow	318.90	97.27	8.51	0.09
7	LST	ABP	328.06	97.86	11.50	0.16
8	LST	Railway	268.99	96.63	8.09	0.16
9	LST	WET	208.10	96.58	12.16	195.65
10	RFDFF	Fix flow	23.19	96.77	25.49	0.09
11	RFDFF	ABP	22.74	96.78	25.49	0.19
12	RFDFF	WET	22.21	96.74	25.49	0.36

5.2 Low activity duration variability

In this section we analyze the results obtained for the low activity duration settings. Activity durations are still stochastic, but the variance of the distribution is reduced. We observe that the predictive-reactive scheduling combinations have a lower stability cost and a higher TPCP than in the high variability case. The results are shown in Table 4.

All predictive-reactive procedures yield much smaller stability costs than for the high duration variability settings due to the fact that the disruptions are smaller in number and size. The beneficiary impact of a solution robust baseline schedule on the stability cost is now more clearly pronounced. While the high variability setting results of Table 3 reveal that RCPSP-Fix Flow had a stability cost that was almost five times higher than for RFDFF-Fix Flow, this stability cost ratio is now as high as 15. Solution robust reactive scheduling (WET) results in an outstanding makespan performance and requires much less computational time than in the high variance case.

We can conclude that when activity duration variability is low, it pays to invest in predictive and reactive scheduling procedures that aim at stability. Observing the high TPCP values obtained by the solution robust scheduling combinations (rows 4, 5, 10, 11 and 12 in Table 4), we can even state that pure quality robust scheduling is inadequate when activity duration variability is low. Most likely, a project manager would opt for RFDFF-Fix Flow (row 10) resulting in a sufficiently high TPCP and a very low stability cost at almost no computational time. The fact that this proactive-reactive scheduling combination has a very high average makespan deviation of 19.21 (shown in column *dev.*) does not cause any problems, because the high TPCP of 96.58% indicates that this delay in the realized project completion hardly leads to due date violations. If the TPCP values slightly above 96.5% (rows 10–12) obtained by the solution robust procedures would be deemed unacceptable, management should opt for the RCPSP-WET or RCPSP-Railway procedure.

The strong impact of the activity duration variability on the obtained results is striking. RFDFF-Fix Flow had an insufficient TPCP value for high variability settings, while it produces very attractive results when activity duration variability is low. A project manager who anticipates minor schedule disruptions and therefore decides to go for a deterministic quality robust baseline schedule, runs into a serious misconception of the impact of activity duration variability. The lower the variability in the duration of the project activities, the more attractive solution robust baseline scheduling procedures become.

5.3 Loose project due date settings

In this section the deterministic project due date δ_n is set 30% above the minimum duration schedule obtained using average activity durations. This due date seems to offer a rather wide protection for the disturbances studied in this paper.

The aggregated results are shown in Table 5. They are correlated with the results obtained for the low-variability case (Table 4). We observe consistently higher stability costs, but the basic conclusions drawn from this parameter setting do not differ from the conclusions of the previous section. This means that a project planner should rely on solution robust baseline or reactive scheduling, depending on the importance attributed to obtaining a high TPCP.

5.4 Tight project due date settings

Table 6 shows the results obtained when the due date δ_n is set 15% above the minimum project duration obtained

Table 6 Performance values for tight due date settings

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	335.48	97.48	0.79	9.63
2	RCPSP	Fix flow	298.28	97.02	1.18	0.09
3	RCPSP	ABP	298.34	97.29	1.03	0.16
4	RCPSP	Railway	258.70	96.48	3.70	0.17
5	RCPSP	WET	238.32	94.05	4.56	373.41
6	LST	Fix flow	351.32	72.15	8.51	0.09
7	LST	ABP	356.99	74.65	8.09	0.16
8	LST	Railway	312.56	64.85	11.50	0.16
9	LST	WET	254.66	67.83	11.56	429.11
10	RFDFF	Fix flow	73.09	80.35	7.08	0.09
11	RFDFF	ABP	71.37	80.60	13.05	0.20
12	RFDFF	WET	68.54	80.63	12.82	18.76

Table 7 Performance values for large *wp*

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	334.81	98.72	0.79	9.63
2	RCPSP	Fix flow	297.45	98.48	1.18	0.08
3	RCPSP	ABP	297.58	98.61	1.03	0.16
4	RCPSP	Railway	257.75	98.21	3.70	0.16
5	RCPSP	WET	236.18	96.92	4.66	356.84
6	LST	Fix flow	341.27	84.71	8.51	0.09
7	LST	ABP	347.91	86.25	8.09	0.16
8	LST	Railway	298.95	80.74	11.50	0.16
9	LST	WET	238.51	82.22	11.85	328.03
10	RFDFF	Fix flow	52.55	92.23	18.19	0.09
11	RFDFF	ABP	51.31	92.36	18.19	0.20
12	RFDFF	WET	49.44	92.33	18.19	12.34

using average activity durations. In this case we again encounter the problem that the ex-post solution of the problem using the realized activity durations does not always yield a makespan within the predefined due date δ_n, whatever the proactive-reactive scheduling procedure used. The maximum obtainable TPCP is 98.5%, exactly the same value as in Sect. 5.1. This is due to the fact that the unavoidable due date violation problem typically occurs when high activity duration variability is combined with tight due dates. Only procedures 1–4 generate service levels that exceed the 95% threshold.

It appears that tight due dates in combination with high activity duration variability reveal a stability-makespan trade-off in that the price we have to pay for obtaining an acceptable TPCP is a high stability cost (e.g., RCPSP-RCPSP). The average results obtained over the parameter setting combinations (high, 5, +15%) and (high, 10, +15%) (not shown in Table 6) divulge a maximal obtainable TPCP of only 97.04%. But more importantly, the procedures that aim for a solution robust baseline schedule (rows 10–12) yield an inferior TPCP, while RCPSP-WET for which only the reactive module aims at solution robustness still yields an inadequate 89% TPCP. Remark that this is the first case where the TPCP obtained by this procedure is really considered unsatisfactory. From all procedures that can improve stability, only RCPSP-Railway obtains a reasonable 93%. But as stated before, the Railway reactive procedure is an adaptation of the quality robust ABP procedure and can hardly be typified as a real solution robust procedure because no statistical information about uncertainty is taken into account. The search for solution robust reactive procedures with acceptable makespan performance will be a major point of interest for future research.

5.5 Large *wp* settings

The overall performance values given in Table 2 and the results obtained for large *wp* settings given in Table 7 are very similar for the predictive-reactive procedures shown in rows 1–9 of both tables. The limited impact of the *wp* on the performance measures does not come as a surprise because all the procedures used to generate quality robust baseline schedules do not rely on the activity weights and thus do not depend on the weighting parameter *wp*. WET is the only reactive procedure that relies on the activity weights in order to make its decisions. The result is that among the predictive-reactive procedures shown in rows 1–9, only the procedures RCPSP-WET (row 5) and LST-WET (row 9) may yield different realized schedules and thus a different TPCP for different *wp*-values. The last term $w_n(\mathbf{s}_n - \delta_n)$ in the stability cost function $\sum w_j E|\mathbf{s}_j - s_j|$ depends on *wp*. However, because all the predictive-reactive procedures have a rather high TPCP (and thus for many executions $\mathbf{s}_n \leq \delta_n$) the impact of this last term on the total stability cost function is minor.

For the procedures that rely on RFDFF to generate the baseline schedule, *wp* has a strong impact. This had already been observed by Van de Vonder et al. (2006) who reaches the paradoxical conclusion that procedures such as RFDFF, that aim at generating solution robust (stable) baseline schedules, are at their best when quality robustness (i.e., makespan performance) is deemed important for the project, i.e., when *wp* values are large. For large values of *wp*, RFDFF starts to act as a quality robust scheduling procedure that still keeps an eye on stability. The results of Table 7 show that for $wp = 10$ procedures RFDFF-Fix Flow, RFDFF-ABP and RFDFF-WET (rows 10–12) yield a 92% TPCP, a very promising result bearing in mind that these results include the high variability and tight due date cases and

Table 8 Performance values for small *wp*

	Baseline	Reactive	Stability	TPCP	Dev.	Time
1	RCPSP	RCPSP	334.11	98.72	0.79	9.66
2	RCPSP	Fix flow	296.61	98.48	1.18	0.09
3	RCPSP	ABP	296.79	98.61	1.03	0.16
4	RCPSP	Railway	256.78	98.21	3.70	0.17
5	RCPSP	WET	233.82	96.89	4.60	349.98
6	LST	Fix flow	328.96	84.71	8.51	0.09
7	LST	ABP	337.13	86.25	11.50	0.16
8	LST	Railway	282.61	80.74	8.09	0.16
9	LST	WET	224.13	82.19	11.87	296.61
10	RFDFF	Fix flow	43.74	84.90	20.59	0.09
11	RFDFF	ABP	42.81	85.02	20.58	0.19
12	RFDFF	WET	41.31	85.03	20.58	6.78

that RFDFF is only a simple heuristic procedure. The procedures even outperform the LST heuristic that aims at quality robust schedules. These excellent results certainly encourage further research efforts on the development of effective and efficient scheduling procedures that aim at the generation of solution robust schedules.

5.6 Small *wp* settings

As mentioned above, the *wp* has an almost negligible impact on the results obtained by the predictive-reactive procedures that aim at a quality robust baseline schedule (rows 1–9 in Table 8). For the procedures that rely on RFDFF for generating the baseline schedule (rows 10, 11 and 12 in Table 8), the TPCP drops to 85%. In line with the overall results of Table 2, the procedures RCPSP-Fix Flow, RCPSP-ABP, RCPSP-Railway and RCPSP-WET (rows 2–5) yield the best combined results for small *wp* values.

6 Conclusions

The overall objective of this paper was to evaluate the performance of various predictive-reactive project scheduling procedures under the combined stability-timely project completion objective. The very promising results obtained by the proactive RFDFF heuristic that aims at generating solution robust (stable) baseline schedules, holds an invitation to continue the research on the development of stable baseline schedules. We advise project managers to add as much solution robustness to a proactive-reactive project environment as the characteristics of the project allow to. Certainly when timely project completion is deemed important (high values for the weighting parameter *wp*), when the activity duration variability is not too high and when the predefined project completion due date leaves some room for buffer insertion, the use of proactive scheduling procedures that aim at generating solution robust (stable) schedules pays off.

The computational experiment revealed that stable baseline scheduling procedures such as RFDFF do not perform that well on timely project completion when due dates are tight, duration variability is high, and *wp* values are small. In such project environments smart predictive-reactive scheduling procedures that combine solution and quality robust procedures are available that deliver good results. For example, combining a procedure that generates minimum duration baseline schedules with a stability-improving reactive policy, such as WET, clearly performs very well overall. Only if variability is very high and the due date is tight, this combination might still result in a unsatisfying makespan performance. The search for solution robust reactive procedures that maintain a good makespan performance and have a lower computational requirement than WET is an interesting issue for future research.

Acknowledgements This research has been supported by Project OT/03/14 of the Research Fund K.U.Leuven.

References

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*, 294–316.

Aytug, H., Lawley, M., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, *161*(1), 86–110.

Blazewicz, J., Lenstra, J., & Kan, A.R. (1983). Scheduling subject to resource constraints—classification and complexity. *Discrete Applied Mathematics*, *5*, 11–24.

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

Davenport, A., & Beck, J. (2002). A survey of techniques for scheduling with uncertainty. Unpublished manuscript.

Davis, E., & Patterson, J. (1975). A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, *21*, 944–955.

Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, *38*, 1803–1818.

Demeulemeester, E., & Herroelen, W. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, *43*, 1485–1492.

Demeulemeester, E., & Herroelen, W. (2002). *International series in operations research & management science*: *Vol. 49*. *Project scheduling—A research handbook*. Boston: Kluwer Academic.

Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, *6*, 17–38.

Goldratt, E. (1997). *Critical chain*. North River: Great Barrington.

Hagstrom, J. (1988). Computational complexity of PERT problems. *Computers and Operations Research*, *18*, 139–147.

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

Herroelen, W., De Reyck, B., & Demeulemeester, E. (2000). On the paper "Resource-constrained project scheduling: notation, classification, models and methods" by Brucker et al., *European Journal of Operational Research*, *128*(3), 221–230.

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

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research*, *167*(3), 592–623.

Igelmund, G., & Radermacher, F. (1983a). Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, *13*, 29–48.

Igelmund, G., & Radermacher, F. (1983b). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, *13*, 1–28.

Kolisch, R. (1996). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, *14*, 179–192.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In: J. Weglarz (Ed.), *Project scheduling: Recent models, algorithms and applications*, Dordrecht: Kluwer Academic.

Kolisch, R., & Padman, R. (1999). An integrated survey of deterministic project scheduling. *Omega*, *49*, 249–272.

Leus, R. (2003). *The generation of stable project plans*. PhD thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Leus, R., & Herroelen, W. (2005). The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters*, *33*, 151–156.

Mastor, A. (1970). An experimental and comparative evaluation of production line balancing techniques. *Management Science*, *16*, 728–746.

Mehta, S., & Uzsoy, R. (1998). Predictive scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, *14*, 365–378.

Möhring, R., Radermacher, F., & Weiss, G. (1984). Stochastic scheduling problems, I: Set strategies. *Zeitschrift für Operations Research*, *28*, 193–260.

Möhring, R., Radermacher, F., & Weiss, G. (1985). Stochastic scheduling problems, II: General strategies. *Zeitschrift für Operations Research*, *29*, 65–104.

Pascoe, T. (1966). Allocations of resources c.p.m. *Revue Française de Recherche Opérationelle*, *38*, 31–38.

Patterson, J. (1976). Project scheduling: the effects of problem structure on heuristic scheduling. *Naval Research Logistics*, *23*, 95–123.

Stork, F. (2001). *Stochastic resource-constrained project scheduling*. PhD thesis, School of Mathematics and Natural Sciences, Technical University of Berlin.

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.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2001). An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Annals of Operations Research*, *102*, 179–196.

Wang, J. (2005). Constraint-based schedule repair for product development projects with time-limited constraints. *International Journal of Production Economics*, *95*, 399–414.

Zhu, G., Bard, J., & Yu, G. (2005). Disruption management for resource-constrained project scheduling. *Journal of the Operational Research Society*, *56*, 365–381.