International Journal of Production Research,
Vol. 44, No. 2, 15 January 2006, 215–236

Taylor & Francis
Taylor & Francis Group

The trade-off between stability and makespan in resource-constrained project scheduling

S. VAN DE VONDER, E. DEMEULEMEESTER,
W. HERROELEN* and R. LEUS

Research Centre for Operations Management, Katholieke Universiteit Leuven,
Naamsestraat 69, B-3000 Leuven, Belgium

(*Received September 2004; in final form April 2005*)

During the last decade, considerable research efforts in the project scheduling literature have concentrated on resource-constrained project scheduling under uncertainty. Most of this research focuses on protecting the project due date against disruptions during execution. Few efforts have been made to protect the starting times of intermediate activities. In this paper, we develop a heuristic algorithm for minimizing a stability cost function (weighted sum of deviations between planned and realized activity starting times). The algorithm basically proposes a clever way to scatter time buffers throughout the baseline schedule. We provide an extensive simulation experiment to investigate the trade-off between quality robustness (measured in terms of project duration) and solution robustness (stability). We address the issue whether to concentrate safety time in so-called project and feeding buffers in order to protect the planned project completion time or to scatter safety time throughout the baseline schedule in order to enhance stability.

Keywords: Project management; Scheduling/sequencing; Simulation methods

1. Problem description

The *resource-constrained project scheduling problem* (RCPSP) aims at minimizing the duration of a project, subject to the finish–start, zero-lag precedence constraints and the renewable resource constraints. Many exact and heuristic algorithms for solving the deterministic RCPSP have been described in the literature (for overviews, see Herroelen *et al*. 1998, Brucker *et al*. 1999, Kolisch and Hartmann 1999, Kolisch and Padman 1999, Demeulemeester and Herroelen 2002) for solving the deterministic RCPSP. When uncertainty comes into play, it has been advocated that project-planning practitioners rely on the well-known *critical chain buffer management* (CC/BM) methodology (Goldratt 1997), a heuristic approach that tries to deliver good makespan protection (i.e. *quality robust* schedules). Recently, however, *stability* or *solution robustness* has become a central point of attention in project scheduling (Herroelen and Leus 2005). This means that given the uncertainty during execution,

*Corresponding author. Email: willy.herroelen@econ.kuleuven.be one would like the realized schedule to resemble the projected schedule (as defined in section 3) as much as possible. This projected schedule is used to organize resources, negotiate contracts with subcontractors, etc. Deviations from this projected schedule will induce stability costs, which may include financial costs, inventory costs or various organizational costs.

For projects with ample resource availability, exact and suboptimal algorithms have been developed to produce schedules under the objective of maximizing solution robustness (Leus 2003). In a recent paper (Van de Vonder *et al.* 2005), the authors have provided an extensive analysis of the results of a simulation experiment set up to investigate the trade-off between stability and makespan. Research on the generation of stable baseline schedules in a project environment with constrained resources is still in a burn-in stage. Recently, Leus (2003) and Leus and Herroelen (2004) have proposed an exact resource-allocation procedure for a given baseline schedule.

The main contribution of this paper is twofold. We develop a heuristic procedure for generating baseline schedules that exhibit acceptable quality and solution robustness in the presence of multiple activity disruptions, and we provide a thorough analysis of the stability/makespan trade-off in a resource-constrained project scheduling environment. This analysis addresses the issue of whether it is beneficial to protect the project makespan by concentrating buffers at the end of the project or to spread buffers throughout the baseline schedule in order to enhance stability. For this purpose, stability and makespan performance measures of the newly developed heuristic procedure are compared with results of CC/BM for a wide range of project characteristics.

The paper is organized as follows. In the next section, we present our heuristic procedure for generating stable resource-constrained baseline schedules and provide an illustration by means of an example problem. In section 3, we exploit the same problem instance to describe our implementation of the CC/BM scheduling methodology for generating so-called buffered baseline schedules and unbuffered projected schedules. Section 4 is devoted to the description of the computational experiment set up to examine the stability/makespan trade-off. The experimental results are described in section 5. We conclude the paper with overall conclusions and suggestions for further research.

2. A heuristic procedure for generating stable resource-constrained schedules

Leus (2003), Herroelen and Leus (2004) and Van de Vonder *et al.* (2005) describe a heuristic procedure for generating buffered baseline schedules for projects with ample renewable resource availability. Basically, their *adapted float factor heuristic* (ADFF) is an adaptation of the float factor model that was originally introduced by Tavares *et al.* (1998) to generate a schedule S in which the start time of activity i is obtained as $s_i(S) := s_i(\text{ESS}) + \alpha[s_i(\text{LSS}) - s_i(\text{ESS})]$, where $\alpha \in [0, 1]$ is the so-called *float factor*, $s_i(\text{ESS})$ denotes the earliest possible start time of activity i, and $s_i(\text{LSS})$ represents the latest allowable start time of activity i. Both start times are derived from critical path calculations for a given project due date. Instead of using a single float factor α for all the activities, ADFF adopts an *activity-dependent float factor* that is calculated as $\alpha_i = \beta_i/(\beta_i + \delta_i)$, where β_i is the sum of the weight of activity *i* and the weights of all its transitive predecessors, while δ_i is the sum of the weights of all transitive successors of activity *i*. In doing so, ADFF inserts longer time buffers in front of activities that would incur a high cost if started earlier or later than originally planned.

Obviously, when applied to a resource-constrained project, ADFF scatters intermediate time buffers throughout a baseline schedule but does not prohibit resource conflicts from occurring because neither the early start schedule nor the late start schedule is guaranteed to be resource-feasible. To ensure that the buffered baseline schedule is resource-feasible, the ADFF procedure is modified as follows.

The first step is to obtain a good precedence and resource-feasible starting schedule. A number of exact procedures for generating minimum duration schedules for the RCPSP have been described in the literature (see Demeulemeester and Herroelen 2002). For illustrative purposes, we use the branch-and-bound procedure of Demeulemeester and Herroelen (1992, 1997) for generating a minimum makespan resource-constrained schedule. The simple example network of figure 1 will be our vehicle of analysis. This network is a 10-activity, zero-lag, finish–start activity-on-the-node network with three single-item renewable resources, identified for each activity by the bracketed capital letters above the corresponding node. Activities 0 and 9 are dummies, respectively denoting the single start and end node of the project. The numbers below each node denote the corresponding expected activity duration to be used in generating a baseline schedule and a weight that denotes a relative cost of actually starting the activity one time unit earlier or later than originally planned in the baseline schedule. Activity 6, for example, has a planned duration of nine periods, must be performed by single unit resource A and has a stability cost of 1.

The activity-dependent weights may include unforeseen storage costs, extra organizational costs, costs related to agreements with subcontractors, costs to penalize the resulting system nervousness and shop co-ordination difficulties, or they may just represent a cost that expresses the dissatisfaction of employees with schedule changes. In many practical settings, the weights may represent the penalties, stipulated in the contractual agreements, that are to be paid by the project contractors when intermediate project milestones or the project due date are exceeded. In practice, these penalties may be considerable. For example, three delivery dates were specified for the renovated Berlaymont building, housing the European Commission in Brussels: 31 December 2003 for the basic building including all the

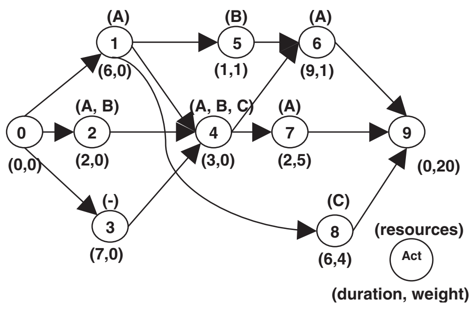

Figure 1. Example project network.

main office surfaces, 31 March 2004 for all other works required to prepare the physical infrastructure for enlargement, and 30 June 2004 for the multi-media equipment. If either of the agreed delivery dates was not met, a penalty was set to €221 000 per month of delay until the date of the actual delivery (European Commission 2004).

The *minimum duration schedule* obtained by the branch-and-bound procedure is shown in figure 2. The critical sequence, i.e. the precedence and resource-constrained chain of activities that determines the 22-period makespan, is the chain $\langle 0, 1, 2, 4, 7, 6, 9 \rangle$. The project due date is set to 33, 50% higher than the critical sequence length. Note that alternative optimal schedules are possible. Figure 3 gives the corresponding *right-justified schedule*. For every activity i, the float value 'float[i]' is calculated as the difference between its latest allowable starting time (its starting time in the right-justified schedule) and its scheduled starting time in the minimum duration schedule of figure 2 {float[i] $= s_i(\text{LSS}) - s_i(\text{ESS})$}. Given the project due date of 33, the latest allowable starting time of activity 5 is 23 (as can be seen in figure 3). Hence, because $s_5(\text{ESS}) = 11$, we find that float[5] $= 23 - 11 = 12$.

In a second step, the starting time of each activity i is calculated as $s_i(S) := s_i(\text{B\&B}) + \alpha_i(\text{float}[i])$, where $s_i(\text{B\&B})$ denotes the starting time of activity i in the minimum duration schedule (the schedule of figure 2). However, using the ADFF float factors $\alpha_i = \beta_i/(\beta_i + \delta_i)$ does not ensure that the resulting $s_i(S)$ are resource-feasible. Indeed, although both the activity starting times in the minimum duration schedule and in the right-justified schedule are resource feasible, this might not be the case for the computed $s_i(S)$. Table 1 gives the weights $w[i]$, floats $f[i]$, durations $d[i]$ and value α_i for all activities of the example network when the project

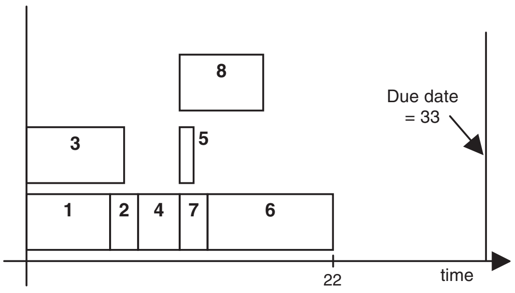

Figure 2. Minimum duration schedule.

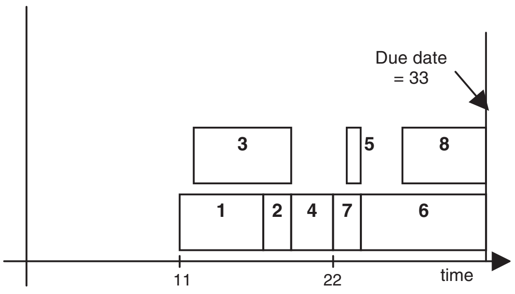

Figure 3. Right-justified schedule.

Table 1. Calculation of starting times for figure 4.

	s_i(B&B)	$f[i]$	$w[i]$	$d[i]$	β_i	δ_i	α_i	$s_i(S)$
0	0	0	0	0	0	31	0	0
1	0	11	0	6	0	31	0	0
2	6	11	0	2	0	26	0	6
3	0	12	0	7	0	26	0	0
4	8	11	0	3	0	26	0	8
5	11	12	1	1	1	21	0.045	11.545
6	13	11	1	9	2	20	0.09	14
7	11	11	5	2	5	20	0.2	13.2
8	11	16	4	6	4	20	0.167	13.667
9	22	11	20	0	31	0	1	33

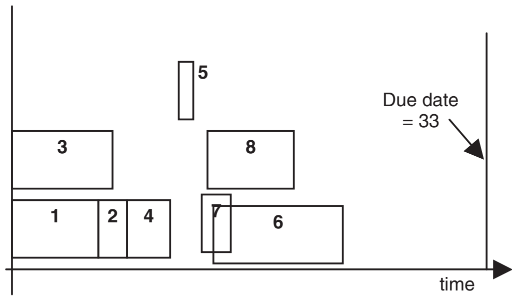

Figure 4. Schedule generated by standard ADFF.

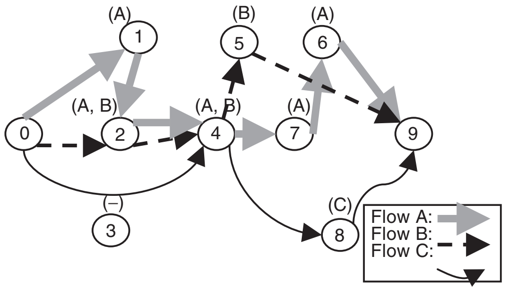

Figure 5. Resource flows for the example network.

due date is fixed at 33. Figure 4 shows us the resulting schedule, in which we observe the existence of a resource conflict: activities 6 and 7 are concurrent users of the single resource item A between time 14 and time 15.2, which obviously violates the resource constraint.

In order to obtain a precedence and resource-feasible schedule, a set of different float factors α_i has to be used. For this purpose, a resource flow network is constructed for each resource type. A *resource flow network* (Artigues and Roubellat 2000) identifies how each single item of a resource is passed on through a schedule. When a certain item of resource is transferred from activity i to activity j upon completion of activity i, the flow between those two activities will be positive. Figure 5 represents the flow network based on the schedule of figure 2 for all

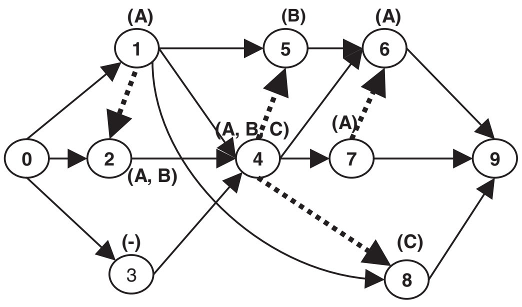

Figure 6. Adapted project network with extra flow-based precedence relations.

Table 2. Calculation of starting times for figure 7.

	s_i(B&B)	$f[i]$	$w[i]$	$d[i]$	β_i	δ_i	α_i	$s_i(S)$
0	0	0	0	0	0	31	0	0
1	0	11	0	6	0	31	0	0
2	6	11	0	2	0	31	0	6
3	0	12	0	7	0	31	0	0
4	8	11	0	3	0	31	0	8
5	11	12	1	1	1	21	0.045	11.545
6	13	11	1	9	7	20	0.259	15.852
7	11	11	5	2	5	21	0.192	13.115
8	11	16	4	6	4	20	0.167	13.667
9	22	11	20	0	31	0	1	33

three single-unit resource types A, B and C. In general, of course, renewable resources can have availabilities that are larger than one. If that is the case, multiple possible flow networks can be constructed. We have opted for the procedure described by Artigues and Roubellat (2000) to generate a feasible resource flow network for each resource type.

The original project network will now be modified as follows. Every pair of activities i and j that are not (directly or transitively) ordered in the original project network and for which there is a positive flow going from i to j in the resource flow network will be linked by an extra precedence constraint. In this way, we obtain an adapted network as depicted in figure 6. The computation of the α_i taking into account the extra precedence constraints is shown in table 2. The corresponding final schedule is shown in figure 7. We will later refer to this procedure as the *resource flow-dependent float factor* (RFDFF) heuristic.

3. Critical chain buffer management (CC/BM)

In the experimental set-up of the next section the above-described RFDFF-heuristic will be compared with CC/BM. CC/BM—the direct application of the Theory of Constraints (TOC) to project management (Goldratt 1997)—has received considerable attention in the project management literature. The fundamental

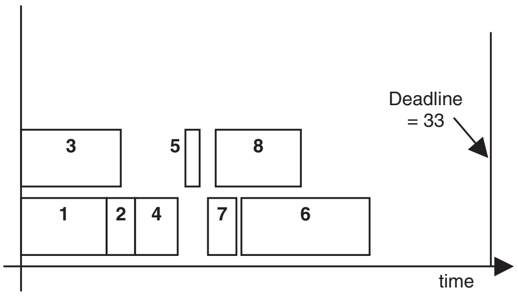

Figure 7. Baseline schedule generated by the RFDFF heuristic.

working principles of CC/BM have been reviewed by Goldratt (1997), Newbold (1998) and Herroelen and Leus (2001). CC/BM builds a baseline schedule using aggressive median or average activity duration estimates rather than using activity durations that are based on the 80–90% confidence levels which, according to CC/BM, are in common use in project-management practice. The safety in the durations of activities that was cut away by selecting aggressive duration estimates is concentrated at the end of the schedule in the form of a *project buffer* (PB) that is positioned at the end of the so-called critical chain. The *critical chain* is defined as the longest chain of precedence and resource-dependent activities that determines the overall duration of a project. If there is more than one critical chain, an arbitrary choice is made. The project buffer should protect the project due date from variability in the critical chain activities. *Feeding buffers* (FB) are inserted whenever a non-critical chain activity joins the critical chain. This basically means that non-critical chains will start earlier in time. By doing this, new resource conflicts can be provoked. The literature is not very clear on how those conflicts should be solved. For executing a project, the CC/BM approach does not rely on the buffered schedule but relies on a so-called *projected schedule*. This schedule is precedence- and resource-feasible, contains no buffers and is to be executed according to the road-runner mentality, i.e. the so-called gating tasks (activities with no non-dummy predecessors) are started at their scheduled start time in the buffered schedule while the other activities are started as soon as possible. The projected schedule is recomputed when disruptions occur. Neither the buffered schedule nor the projected schedule is constructed with a view to stability (*solution robustness*, i.e. the insensitivity of planned activity start times to schedule disruptions). At this juncture, we can explain the implementation of CC/BM that is used in the remainder of this paper.

First, we solve the deterministic RCPSP by running the branch-and-bound code of Demeulemeester and Herroelen (1992, 1997). Because CC/BM starts with a baseline schedule that is as late as possible, we run the procedure on the inverse network and reverse the resulting schedule again to obtain a right-justified resource feasible unbuffered schedule. For our example network of figure 1, this results in the schedule of figure 8, where we identify the critical chain as the sequence $\langle 0, 2, 1, 4, 7, 6, 9\rangle$. Note that this critical chain differs slightly from that obtained in the previous section. The order of the unrelated activities 1 and 2 is reversed because the code was executed on the reversed network. This will obviously not affect the project makespan.

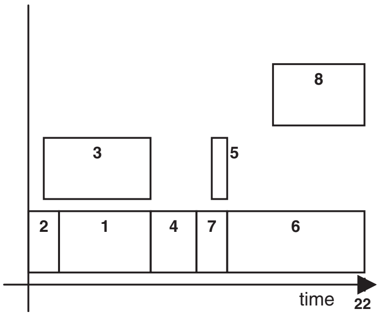

Figure 8. Right-shifted minimal duration schedule.

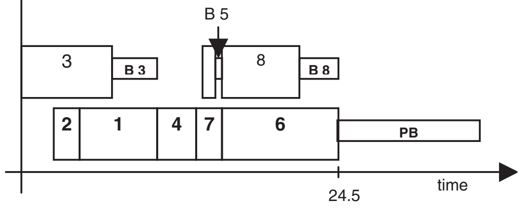

Figure 9. Buffered CC/BM baseline schedule.

Besides identifying the critical chain, we now also have to compute the feeding buffers. For the example network in figure 1, clearly three non-critical chains can be discovered. CC/BM adds feeding buffers between the last activity of the non-critical chains and the activity of the critical chain where this feeding non-critical chain joins the critical chain. In the example, we will add three feeding buffers, namely between the activities 3 and 4, 5 and 6 and 8 and 9. For the time being, the size of a feeding buffer is set to 50% of the length of its feeding chain. The buffer sizing decision will be further examined in section 5.6.

As was already mentioned earlier and as has been demonstrated by Herroelen and Leus (2001), simply starting the feeding chains earlier in time to make room for the feeding buffers may introduce new resource conflicts. Instead of using some heuristic to resolve these resource conflicts, we opt for a complete rescheduling procedure in which the buffers are properly sized and considered as extra dummy activities with positive duration and no resource requirements, while ensuring that the sequential order of the critical chain activities is kept unchanged. For the example network, this results in the buffered baseline schedule of figure 9 where three feeding buffers and a 50% project buffer have been inserted.

For executing a project, however, CC/BM does not rely on this *buffered baseline schedule* but relies on the so-called *projected schedule*, which has been introduced earlier in this section. The alert reader will observe that the construction of such a projected schedule requires some additional information, which can for example be obtained by fixing the flows of a resource flow network. The derivation of the earliest possible activity starting times in the projected schedule depends not only on the original precedence constraints but also on the resource flows between activities. All activities that pass on resources to other activities should be completed by the time these other activities start. Clearly, when disruptions occur during project

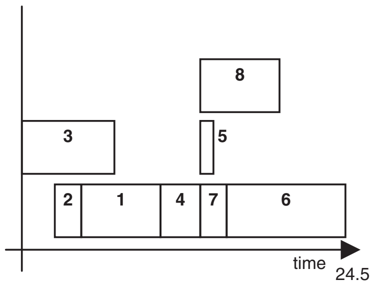

Figure 10. Initial CC/BM projected schedule.

execution, the projected schedule has to be recomputed. Figure 10 shows the initial projected schedule for our example network of figure 1. Activities 5 and 8 are started earlier, i.e. at the completion time of activity 4. Note that there is obviously no resource flow between activities 5 and 8 so that they can be scheduled concurrently.

4. Set-up of the computational experiment

In this section, we describe our experimental set-up that is used to investigate the trade-off between quality robustness and solution robustness. Several network generators for project scheduling problems have been developed (Demeulemeester *et al.* 1993, 2003, Kolisch *et al.* 1995, Schwindt 1995, Agrawal *et al.* 1996). The often-used instances in the project scheduling problem library PSPLIB have been generated using ProGen (Kolisch *et al.* 1995), which takes into account network topology and resource-related characteristics.

We use the RanGen software developed by Demeulemeester *et al.* (2003) to generate network instances. The reasons for selecting RanGen can be summarized as follows (see also Demeulemeester *et al.* 2003). The generator aims at generating strongly random problem instances that span the full range of problem complexity and uses a non-superfluous reliable set of complexity measures which have been shown in former studies to stand in clear and strong relation to the hardness of resource-constrained project scheduling problems. It guarantees networks with pre-specified values of the order strength (OS), resource factor (RF) and resource constrainedness (RC), being precisely the three complexity measures used in our factorial design.

The settings for the parameters used in our factorial design are shown in table 3. Every network is characterized by the number of activities n and by OS (Mastor 1970). OS describes the density of the network and is defined as the number of precedence relations, including the transitive relations, divided by the theoretical maximum number of such precedence relations, $[n(n-1)]/2$. Previous studies (Herroelen and De Reyck 1999) have demonstrated that the order strength is an excellent measure of the impact of the topology of practical project networks on the complexity of the associated resource-constrained project scheduling problem: the higher the value of OS, i.e. the higher the density of the network, the easier

Table 3. Parameter settings for the factorial experiment.

	Low	Medium	High
n	10	20	30
OS	0.3	0.5	0.7
RF	0.5	0.75	1
RC	0.3	0.5	0.7

resource-constrained project scheduling becomes. In our experiments, we will use representative order strength settings of 0.3, 0.5 and 0.7.

In the experiment, four different resource types are considered. The resource usage is defined by two parameters: the *resource factor* (RF) and the *resource constrainedness* (RC). The resource factor reflects the average number of resource types used by an activity (Pascoe 1966). RF = 1 thus means that all activities require all resource types in a certain non-zero quantity. RF = 0 indicates that no activity requests any resource. According to Kolisch *et al.* (1995), there is a positive correlation between the RF and the required CPU time to solve the well-known resource-constrained project scheduling problem, while Alvarez-Valdés and Tamarit (1989) have observed that problems with RF = 1 were easier to solve than problems with RF = 0.5. We include representative RF-settings of 0.5, 0.75 and 1 in our factorial design.

The resource constrainedness (Patterson 1976) defines the average portion of the resource availability that is used by an activity. RC = 0.5 means that the average usage of an activity that needs a certain resource type equals half the availability of that type. The three settings for RC (0.3, 0.5 and 0.7) used in the factorial design represent the critical values found by Herroelen and De Reyck (1999) to characterize the easy–hard–easy phase transition in the complexity of the resource-constrained project scheduling problem.

We set up a factorial experiment to investigate the impact of the above-described parameters on the makespan/stability trade-off. We will only examine the main effects of the parameters by varying the corresponding parameter one by one. For every examined parameter combination, 100 network instances are generated.

The activity weights (apart from the weight of the dummy end activity) are drawn from a uniform distribution between 0 and 4. As mentioned earlier, these weights represent the cost incurred if the corresponding activity would start one time unit earlier or later than planned. The weight of the dummy end activity, which identifies the cost of completing the project later than planned, is defined by the *weighting parameter*, *wp*. This weighting parameter is the ratio between the weight of the dummy end activity and the average of the distribution of all other activity weights (2 in this case). We allow *wp* to fluctuate from 1 to 15, and we redo all calculations for each discrete intermediate value. Such a range makes it possible to represent the transition from a situation where project management deems project completion as equally important as meeting the intermediate milestones, to the situation often occurring in practice, where project management deems timely project completion as of utmost importance.

Given a certain network and specific choices for all parameters, we compute the CC/BM schedule according to the procedure explained in section 3. Then, 100 project executions are simulated using a right-skewed beta distribution for the actual activity durations (mean duration value equal to the deterministic duration used in the baseline schedule, minimum value equal to half the baseline duration and maximum value equal to 2.25 times the baseline duration). For every execution, the makespan performance and the stability cost are computed. The evaluation measure for makespan performance is the *timely project completion probability* (TPCP) that expresses the probability that the project will end by the project due date for certain parameter settings. Stability is measured as the weighted sum of deviations between planned and actual activity start times. Both measures obviously depend highly on the project due date. For this reason, stability and makespan performance are recalculated for different project due dates. More precisely, we add a project buffer, whose size is expressed as a fraction of the critical chain length, to the due date of the baseline schedule. This fraction is incrementally increased from an aggressive 0% to 200%, which ensures a very safe project buffer. Figure 9 shows an example of such a project buffer, where 50% of critical chain length has been added to the due date. Stability cost is also dependent on the weighting parameter, because the weighted sum of deviations also includes the possible tardiness of the project completion, multiplied by the weight of the last activity, which is defined by the *wp*. Thus, all stability calculations have to be repeated for the considered range of *wp* values.

The aim of this paper is to compare stability and makespan performance for a makespan protecting schedule and a stable schedule. CC/BM was chosen as a method that protects the project makespan, while the RFDFF will be used as a heuristic for maximizing the stability. Thus, the same measures should be calculated for the RFDFF-heuristic. However, for this heuristic, the values of *wp* and the project due date affect not only the performance measures but also the baseline schedules (the schedule in figure 7 assumed $wp = 10$ and due date $= 33$). Thus, for each combination of *wp* and due-date prolongation (expressed as a percentage of the critical chain length), a completely new schedule has to be computed. Afterwards, averages over the 100 project executions are compared between CC/BM and RFDFF for any combination of parameter setting, weighting parameter and project due date. Results and interpretations are given in the following sections.

Note that CC/BM and RFDFF do not result in the same minimal project due date. For RFDFF, 0% of project due-date prolongation means that the project due date equals the critical chain length found by the RCPSP procedure. In contrast, in CC/BM the critical chain does not necessarily start at time 0 because of the insertion of the feeding buffers. For example, in figure 10, we note that activity 2 only starts at time 2.5. This results in a project due date of 24.5 instead of 22, the value obtained by the RCPSP procedure of figure 2. We will call this delay of 2.5 time units the *critical chain delay*. Thus, adding a zero-sized project buffer to the CC/BM schedule results in a makespan that equals the CC length plus CC delay. In order to obtain an honest comparison between both methods, we add the critical chain delay to the due date that is imposed on RFDFF in order to ensure that both methods have equal due dates.

5. Experimental results

All previous sections served to introduce the algorithms and the experimental set-up that we need to investigate whether it is advantageous to protect a schedule only for makespan performance or also for stability. Protecting for makespan performance, as done by CC/BM, certainly produces a high TPCP. On the other hand, protecting individual activities for possible disruptions, as done by RFDFF, decreases the stability cost. The interesting issue addressed in this section is the magnitude of the loss of makespan performance when protecting the intermediate milestones. We also investigate the impact of all parameter settings on the makespan/stability trade-off.

5.1 *Impact of the weighting parameter wp*

In this section, we study the results for increasing values of the weighting parameter. A higher *wp* means that the cost of not meeting the proposed project due date increases. Note that the advocates of the CC/BM philosophy typically assume that this weight is rather high. We allow the *wp* to fluctuate between 1 and 15, while all other parameter settings are kept constant (OS = 0.5, RF = 0.75, RC = 0.5).

When *wp* = 1, we notice that the RFDFF schedule needs on average a prolongation of 100% of the critical chain length (above the critical chain length plus the critical chain delay) in order to ensure a 95% TPCP. A 50% prolongation only guarantees an average of 77% of the projects to finish on time. For CC/BM, on the other hand, a project buffer of 31% of the critical chain length already protects the makespan in 95% of all cases. The stability cost for this makespan protective schedule is rather high (compared with the RFDFF schedule), but because a 100% project prolongation is simply infeasible, there is not much choice but to incur the high stability cost of CC/BM to ensure a good makespan performance. However, the setting *wp* = 1 seems rather unrealistic for many real-life projects because the cost of not meeting the project due date will most probably exceed the cost of not meeting an average planned activity starting time.

When *wp* increases, RFDFF devotes more attention to project completion. A larger portion of the total buffer size will be placed in front of the dummy ending activity. This improves the makespan performance. For example, for *wp* = 3, 45% of critical chain prolongation already ensures the proposed makespan to be met in 95% of all cases. Obviously, although makespan performance is still better for CC/BM (remains 31% because the schedule is independent of *wp*), this is a valuable alternative for CC/BM because the stability costs are much lower. Some project managers might opt for such a strategy.

For projects with an even higher wp, this effect will still be stronger. When *wp* = 15, postponing the due date by 28% of critical chain length already results in a 95% TPCP. Surprisingly, this is less than the 31% for CC/BM. The reason for this is that the critical chain delay that occurs at the beginning of the CC/BM schedule will be put at the end of the RFDFF schedule. Thus, when *wp* is very high, the total buffering at the end of the project can become larger for RFDFF than the project buffer inserted by CC/BM. This will be especially the case when the critical chain delay is large.

All of this results in a paradoxical conclusion. The CC/BM philosophy tries to protect project completion because it assumes that project completion is much more important than the timely completion of intermediate activities (actually CC/BM rejects the use of milestones). However, the above remarks show us that exactly when the weight of the ending project activity is high, CC/BM becomes hard to defend.

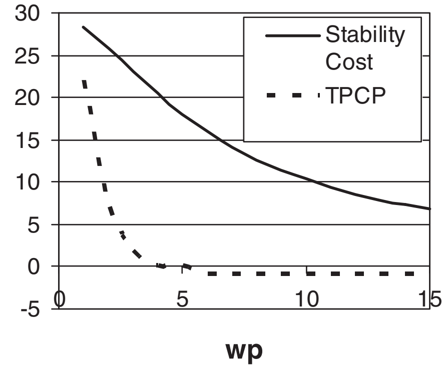

Figure 11. Comparison of RFDFF and CC/BM for total buffering equal to 50% of CC length.

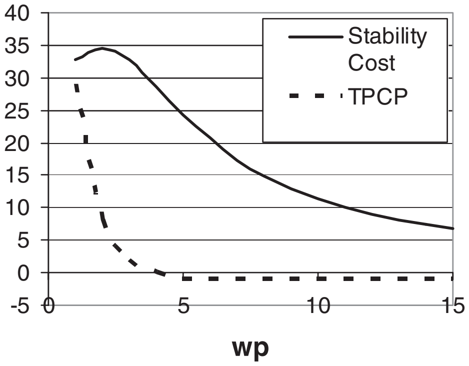

Figure 12. Comparison of RFDFF and CC/BM for total buffering equal to 50% of CC length in the non-resource constrained case.

Even if we would make abstraction of the critical chain delay, we see that the huge advantage of RFDFF in stability cannot be compensated by the difference in makespan performance. We summarize the trade-off between makespan and stability in figure 11. The bold curve indicates the ratio between the CC/BM stability cost for a 50% project due date delay compared with the RFDFF stability cost for the same due date. Obviously this advantage of RFDFF decreases for a higher *wp*, but the difference remains substantial for every *wp* value considered. The dashed curve, on the other hand, denotes the difference in TPCP percentage points between CC/BM and RFDFF. This advantage in makespan performance of CC/BM seems to decrease much more rapidly for increasing *wp*.

The above conclusions are very similar to those found in Van de Vonder *et al.* (2005) for the non-resource-constrained case. However, we must note that in both papers, different assumptions have been made concerning the stochastic distributions of activity weights. To solve this inconvenience, figure 12 represents the makespan/stability trade-off for the same networks, parameter settings (including activity weights distribution) and disturbances of figure 11 when resources are not considered to be a restricting factor. We remark that figure 11 and figure 12 also show a similar impact of the *wp* on both measures of performance. Relaxing the resource constraints does lower stability cost, but almost equally for RFDFF and CC/BM. So, the stability cost ratio will hardly change. The difference in TPCP percentage points also shifts marginally between both figures, but not to the extent that it would change any conclusion.

5.2 *Impact of the number of activities*

In the previous section, we examined the trade-off between quality robustness and solution robustness for varying *wp* values, other project settings being kept constant. In this section, we will investigate the impact of the number of activities (n) in the network on the previous conclusions.

While n was always kept equal to 20 in the previous settings, we have rerun all calculations for networks of 10 and 30 activities. Table 4 shows the average critical chain length and the average deterministic CC/BM due date for the three examined values of n. We note that the CC/BM due date is proportional to the number of activities. It is important to remark that also the critical chain delay increases with increasing number of activities.

Figure 13 shows the difference in required due date delay to ensure a 95% TPCP between RFDFF and CC/BM. We immediately observe that both the trade-off and the paradox that we introduced in the previous section persist. For 10-activity networks, the difference decreases even faster for low *wp* values, which will favour RFDFF. Also, the more activities are in the network, the better RFDFF becomes for very high *wp* values. This can be explained by the positive correlation between n and the critical chain delay. A large critical chain delay means that CC/BM has

Table 4. Critical chain length as a function of n.

n	CC/BM due date $= a$	CC length $= b$	CC delay $= a - b$
10	34.9	33.9	1.0
20	69.4	65.4	4.0
30	103.7	96.0	7.7

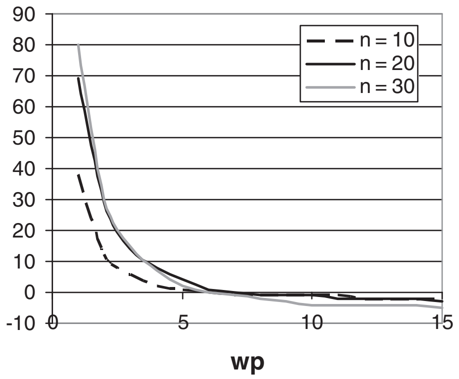

Figure 13. Impact of n on makespan performance for RFDFF and CC/BM.

a buffer at the beginning of the schedule, while the stable schedule can spread this buffer over all activities. If *wp* is very high, RFDFF will concentrate this delay at the end of the schedule and thus protect the makespan. By consequence, a larger number of activities leads to an increasingly better makespan performance for the quality robust schedules when *wp* is high.

When the number of activities exceeds 30, our stable heuristic may consume a lot of computational time in some instances. The bottleneck in the heuristic is obviously the branch-and-bound procedure that, although very efficient, may consume a large amount of CPU time for a specific combination of OS, RF and RC. In this case, we would have to truncate it or replace it by a heuristic procedure for solving the RCPSP. Our results have indicated that this excessive CPU time requirement is only needed by a few network instances. Thus, we have put an upper limit on the CPU time allowed for the branch-and-bound procedure. If this time limit is surpassed, the suboptimal truncated branch-and-bound result will be used.

5.3 *Impact of the order strength*

In this section, we analyse the effect of the order strength on the stability/makespan trade-off. Figure 14 reveals the difference in required due date delay to obtain 95% TPCP between CC/BM and RFDFF for order strengths respectively equal to 0.3, 0.5 and 0.7. We note that the maximal *wp* value for which CC/BM has an advantage over RFDFF is higher when the density of the network increases. For example, for $\mathrm{OS} = 0.3$, the advantage in makespan already disappears when $wp = 6$, while it only disappears when $wp = 9$ for $\mathrm{OS} = 0.7$.

Table 5 indicates that the CC/BM project due date and critical chain delay increase for increasing OS, but this effect is much smaller than for the number of

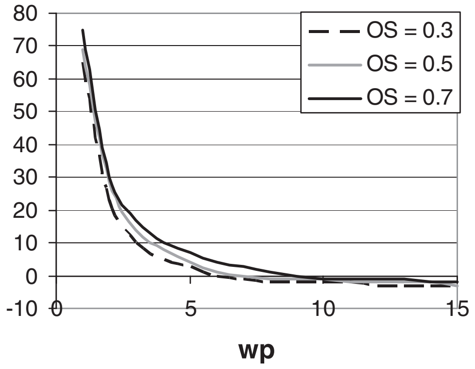

Figure 14. Impact of OS on makespan performance for RFDFF and CC/BM.

Table 5. Critical chain length as a function of *OS*.

OS	CC/BM due date $= a$	CC length $= b$	CC delay $= a - b$
0.3	64.4	61.4	3.0
0.5	69.4	65.4	4.0
0.7	76.6	72.5	4.1

activities. By consequence, we observe in figure 14 that for large *wp* values, the difference in required due date delay between both methods is not dependent on OS. A higher-order strength favours CC/BM for every *wp* and thus makes the trade-off less explicit.

5.4 *Impact of the resource factor*

In previous sections the resource factor RF (Pascoe 1966) was kept equal to 0.75, meaning that an activity uses on average three out of four resource types. We will now investigate the effect of a different RF value, ceteris paribus. RF values of 0.5 and 1 will be considered. Table 6 again gives the CC/BM due date, critical chain length and critical chain delay for the different parameter values.

We note that a higher RF results in a larger makespan. This is no surprise because a higher RF means that more activities require the same resource and thus more flow precedence relations need to be added. Also, a smaller RF seems to induce a larger critical chain delay. Indeed, a small RF means that many activities are scheduled in parallel, and as a result, more feeding buffers are inserted, which can have a stronger effect on the rather aggressive makespan. Following the intuition of previous sections, this negative correlation between RF and the critical chain delay is expected to result in a better makespan performance for CC/BM for high *wp* values. Indeed, figure 15 confirms this expectation, but differences are rather small. Compared with the results in previous and subsequent sections, the RF is the parameter with the smallest impact on the stability/makespan trade-off.

Table 6. Critical chain length as a function of *RF*.

RF	CC/BM due date = a	CC length = b	CC delay = $a - b$
0.5	57.9	51.0	6.9
0.75	69.4	65.4	4.0
1	83.7	82.4	1.3

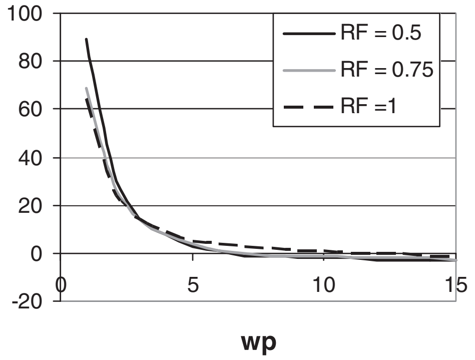

Figure 15. Impact of RF on makespan performance for RFDFF and CC/BM.

5.5 *Impact of the resource constrainedness (RC)*

The resource factor informs us about the percentage of all resource types used on average, but does not give any information about the average resource amount required by the project activities. This is exactly what RC measures. In the previous settings, we used RC = 0.5, meaning that an activity that uses a resource type needs on average 50% of its availability. In this section, we will examine the impact of respectively a lower (0.3) and a higher (0.7) value of RC on the makespan/stability trade-off. Table 7 shows the effect of these RC values on the average makespan.

RC can be seen to have the same impact as RF. A higher RC also means that the resource requirements are more restrictive and thus that less activities can be scheduled in parallel. Less parallel activities make the effect of including feeding buffers smaller, which results in a negative correlation between RC and the critical chain delay. However, unlike RF, RC has a strongly significant effect on the stability/makespan trade-off. For small *wp* values, Figure 16 shows that networks with a small RC need a much larger prolongation of the project due date to ensure 95% TPCP. On the other hand, for high *wp* values, a network with a high RC will result in a very clear makespan performance advantage for RFDFF. The paradox is nowhere more clear-cut than here.

5.6 *Impact of the buffer sizes in CC/BM*

In the previous sections, we have compared the performance of a stable scheduling heuristic with a makespan protecting schedule constructed by the CC/BM approach.

Table 7. Critical chain length as a function of *RC*.

RC	CC/BM due date = a	CC length = b	CC delay = $a - b$
0.3	48.1	38.1	10.0
0.5	69.4	65.4	4.0
0.7	89.5	88.6	0.9

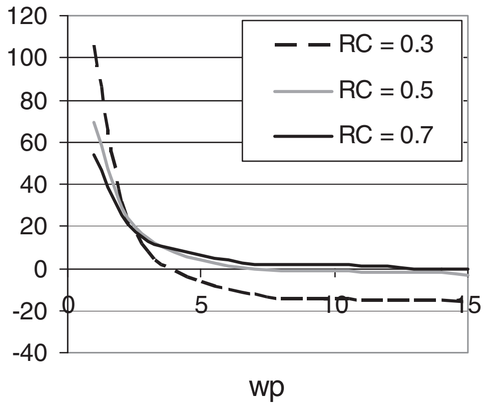

Figure 16. Impact of RC on makespan performance for RFDFF and CC/BM.

In this latter approach, however, we made an assumption about buffer sizes that might substantially influence the performance of the approach: the feeding buffer size was fixed at 50% of the length of the non-critical path that it is protecting. We will investigate other feeding buffer sizes in this section. Also, some conclusions about the required project buffer size for CC/BM will be formulated later in this section.

5.6.1 Feeding buffer size. In CC/BM, a feeding buffer is inserted wherever a non-critical chain joins the critical chain. Thus, all non-critical activities are started earlier in time than their late start time. Otherwise, any delay in the non-critical chain would directly affect the critical chain activities. Starting all activities as early as possible is not applied in CC/BM because this would enormously increase the work in process of the project. Traditional CC/BM literature proposes feeding buffers that are 50% of the length of the chain it has to protect. Figure 17 explores the correlation between the feeding buffer size and the obtained TPCP. γ denotes the percentage of the critical chain length that is added as a project buffer to the original project makespan to produce a project due date, and δ denotes the feeding buffer size as a percentage of non-critical chain length. The resulting curves are represented for γ values of 0, 30 and 50. At first sight, we would conclude that large feeding buffers improve the makespan performance for all three γ values because higher percentages of TPCP are recorded. However, we must stress that the project due dates are not equal and thus dependent on δ. Indeed, increasing the feeding buffer sizes increases the possibility that a non-critical chain will start before the start of the critical chain. This means that the critical chain will not start at time 0, as already shown in figures 9 and 10, which results in an increase of the project due date. The relation between feeding buffer size and makespan performance (in terms of obtained makespan) can be investigated by fixing the due dates. Figure 18 again represents the TPCP values for different δ, but now γ gives the percentage of the critical chain length that is added to the critical chain length (and not to the project makespan), which is independent of the feeding buffer size. By doing this, we can compare the TPCP for different δ values for an equal due date, although it is not the due date that will be

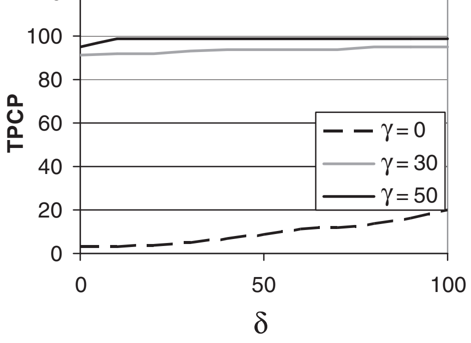

Figure 17. Correlation between feeding buffer size and TPCP.

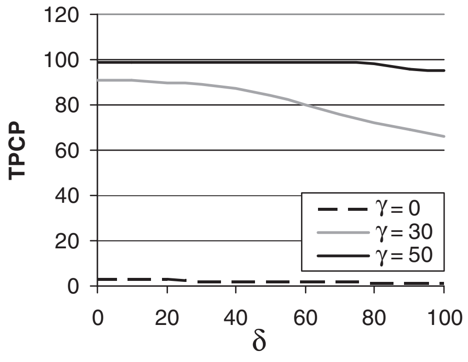

Figure 18. Correlation between feeding buffer size and obtained makespan.

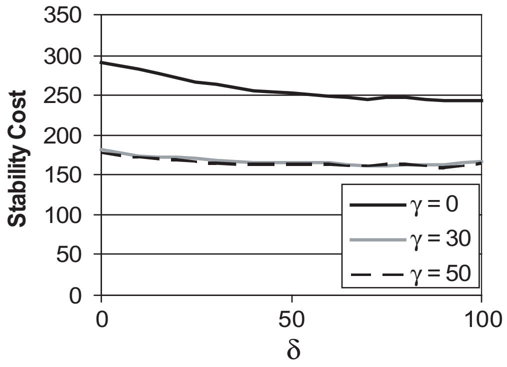

Figure 19. Stability cost for different feeding buffer sizes when $wp = 5$.

proposed by CC/BM. Figure 18 shows that the TPCP now decreases for larger feeding buffers. In fact, this means that the makespan obtained will increase for increasing δ, which is the opposite of what figure 17 suggests.

The average stability costs (when $wp = 5$) for different values of δ and γ are given in figure 19. We note that stability costs disfavour small feeding buffers. Once the feeding buffer size surpasses 50% of the chain length, we note that stability costs remain quite stable, especially for realistic project buffer sizes. Thus, from the stability point of view, we note that starting as early as possible is the best option, while a good makespan performance requires that we avoid delaying the critical chain. From those two measures, we would thus advise inserting maximal feeding buffers such that the non-critical chain will never start earlier than the critical chain. However, by doing this, we would neglect the major reason why CC/BM departs from a late start schedule, namely to reduce the work in process. By taking this into account, we can conclude that the 50% feeding buffer size generates good results and is certainly not outperformed by a different δ value. However, more advanced sizing rules have also been proposed in the literature (Newbold 1998, Herroelen and Leus 2001). Those rules will probably yield better results, but because the focus of this paper is on a trade-off between protecting the makespan and protecting for stability, we do not intend to give a complete review of CC/BM and refer to the specialized literature for an extensive overview.

5.6.2 Project buffer size. In this section, we will take a look at the required project buffer size for CC/BM. The influence of project characteristics on the required buffer size has already been discussed in previous sections, but will be centralized here. Traditionally, a project buffer that equals 50% of the critical chain length is proposed. There seems to be a negative correlation between required project buffer size and *n*, OS and RC, respectively. Note that tables 4, 5 and 7 show that an increase in *n*, OS and RC also triggers an increase in the makespan. Thus, a buffer size equal to a smaller percentage of the critical chain is needed when the critical chain is longer. However, table 6 shows that RF also influences the critical chain length and the project makespan, but the percentage of project buffer required is not dependent on RF.

In general, however, it is no surprise that projects with a large makespan require a smaller percentage of project buffering. In absolute terms, the buffers are still increasing for increasing makespan. For example, it is perfectly possible that a two week project needs a one week buffer, while a one year buffer will most likely be much too safe for a two year project. When we take a 95% TPCP as an absolute requirement, we note that 50% is too much safety for most projects, especially for projects that have a large makespan. It is difficult to give advice about the best project buffer size to make sure that the required TPCP will be obtained, but we must stress that this percentage is dependent on the project characteristics.

6. Conclusions and further research

This paper has described a trade-off between makespan performance and stability, which is an important issue for every project. We have observed that the advantages of the two scheduling approaches developed in this paper depend highly on the project characteristics and especially on the relative importance of timely project completion compared with the importance of timely completion of the intermediate activities. The paradoxical fact that makespan protecting schedules (such as CC/BM) were shown to be hard to defend when makespan becomes very important is the main conclusion of this paper. Improving project managers' awareness of the different scheduling strategies and their strengths and weaknesses is the ultimate aim of our research.

We have made a comparison between makespan protecting scheduling and stable scheduling by applying two algorithms, CC/BM and RFDFF, on a set of projects. Because we were not aware of any algorithm that optimizes the RCPSP for stability under uncertainty, the RFDFF heuristic was developed for this purpose. Developing new stable scheduling heuristics and comparing those with various makespan protecting schedules can be an interesting topic for further research.

In this paper, we build a proactive schedule that includes safety buffers to absorb disruptions. However, during execution, even this stable schedule can become infeasible, and thus a reactive policy will be needed to decide how to react in that case. We have opted for preserving the resource flows between activities whenever a disruption occurs and afterwards planning as early as possible within these restrictions.

However, other reactive policies could be implemented. An interesting open research topic is to investigate the effect of these reactive policies on makespan and stability.

Acknowledgements

This research has been supported by project OT/03/14 of the Research Fund K.U. Leuven and project G.0109.04 of the Research Foundation—Flanders (FWO—Vlaanderen). Roel Leus is partly funded as Postdoctoral Fellow of the Research Foundation—Flanders.

References

Agrawal, M.K., Elmaghraby, S.E. and Herroelen, W.S., DAGEN: a generator of testsets for project activity nets. *Eur. J. Oper. Res.*, 1996, **90**, 376–382.

Alvarez-Valdés, R. and Tamarit, J.M., Heuristic algorithms for resource-constrained project scheduling. In *Advances in Project Scheduling*, edited by R. Slowinski and J. Weglarz, pp. 134–143, 1989 (Elsevier: Amsterdam).

Artigues, C. and Roubellat, F., A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *Eur. J. Oper. Res.*, 2000, **127**, 294–316.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E., Resource constrained project scheduling: notation, classification, models and methods. *Eur. J. Oper. Res.*, 1999, **112**, 3–41.

Demeulemeester, E., Dodin, B. and Herroelen, W., A random activity network generator. *Oper. Res.*, 1993, **41**, 972–980.

Demeulemeester, E. and Herroelen, W., A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Manage. Sci.*, 1992, **38**, 1803–1818.

Demeulemeester, E. and Herroelen, W., New benchmark results for the resource-constrained project scheduling problem. *Manage. Sci.*, 1997, **43**, 1485–1492.

Demeulemeester, E. and Herroelen, W., *Project Scheduling—A Research Handbook*. Vol. 49 of International Series in Operations Research & Management Science, 2002 (Kluwer Academic Publishers: Boston, MA).

Demeulemeester, E., Vanhoucke, M. and Herroelen, W., RanGen: a random network generator for activity-on-the-node networks. *J. Sched.*, 2003, **6**, 17–38.

European Commission, Press Release, MEMO/04/207, 2 September 2004.

Goldratt, E., *Critical Chain*, 1997 (North River Press: Great Barrington, MA).

Herroelen, W. and De Reyck, B., Phase transitions in project scheduling. *J. Oper. Res. Soc.*, 1999, **50**, 148–156.

Herroelen, W., De Reyck, B. and Demeulemeester, E., Resource-constrained scheduling: a survey of recent developments. *Comput. Oper. Res.*, 1998, **25**, 279–302.

Herroelen, W. and Leus, R., On the merits and pitfalls of critical chain scheduling. *J. Oper. Manage.*, 2001, **19**, 559–577.

Herroelen, W. and Leus, R., The construction of stable project baseline schedules. *Eur. J. Oper. Res.*, 2004, **156**, 550–565.

Herroelen, W. and Leus, R., Project scheduling under uncertainty—survey and research potentials. *Eur. J. Oper. Res.*, 2005, **165**, 289–306.

Kolisch, R. and Hartmann, S., Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In *Project Scheduling: Recent Models, Algorithms and Applications*, edited by J. Weglarz, 1999 (Kluwer Academic: Boston, MA).

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

Kolisch, R., Sprecher, A. and Drexl, A., Characterization and generation of a general class of resource-constrained project scheduling problems. *Manage. Sci.*, 1995, **41**, 1693–1703.

Leus, R., The generation of stable project plans. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2003.

Leus, R. and Herroelen, W., Stability and resource allocation in project planning. *IIE Trans.*, 2004, **36**, 1–16.

Mastor, A., An experimental and comparative evaluation of production line balancing techniques. *Manage. Sci.*, 1970, **16**, 728–746.

Newbold, R., *Project management in the fast lane—Applying the theory of constraints*. APICS Series on Constraints Management, 1998 (The St. Lucie Press: Boca Raton, FL).

Pascoe, T., Allocations of resources C.P.M. *Rev. Fr. Rech. Opér.*, 1966, **38**, 31–38.

Patterson, J., Project scheduling: the effects of problem structure on heuristic scheduling. *Naval Res. Logist.*, 1976, **23**, 95–123.

Tavares, L., Ferreira, J. and Coelho, J., On the optimal management of project risk. *Eur. J. Oper. Res.*, 1998, **107**, 451–469.

Van de Vonder, S., Demeulemeester, E., Herroelen, W. and Leus, R., The use of buffers in project management: the trade-off between stability and makespan. *Int. J. Prod. Econ.*, 2005, **97**, 227–240.