INT. J. PROD. RES., 15 APRIL 2004, VOL. 42, NO. 8, 1599–1620

Taylor & Francis
Taylor & Francis Group

Robust and reactive project scheduling: a review and classification of procedures

WILLY HERROELEN† and ROEL LEUS†‡*

The vast majority of the research efforts in project scheduling over the past several years has concentrated on the development of exact and suboptimal procedures for the generation of a baseline schedule assuming complete information and a deterministic environment. During execution, however, projects may be the subject of considerable uncertainty, which may lead to numerous schedule disruptions. Predictive-reactive scheduling refers to the process where a baseline schedule is developed prior to the start of the project and updated if necessary during project execution. It is the objective of this paper to review possible procedures for the generation of proactive (robust) schedules, which are as well as possible protected against schedule disruptions, and for the deployment of reactive scheduling procedures that may be used to revise or re-optimize the baseline schedule when unexpected events occur. We also offer a framework that should allow project management to identify the proper scheduling methodology for different project scheduling environments. Finally, we survey the basics of critical chain scheduling and indicate in which environments it is useful.

1. Introduction

The vast majority of the research efforts in project scheduling over the past several years has concentrated on the development of exact and suboptimal procedures for the generation of a workable *baseline schedule* (*pre-schedule*, *predictive schedule*) assuming a deterministic environment and complete information (for an extensive discussion we refer to Demeulemeester and Herroelen 2002). The project activities are scheduled subject to technological precedence constraints and resource constraints under so-called regular objectives (e.g. project duration) or non-regular objectives (e.g. net present value of the project).

The baseline schedule serves very important functions (Aytug *et al.* 2003, Mehta and Uzsoy 1998). The first is to allocate resources to the different activities to optimize some measure of performance. If developed as a feasible finite capacity schedule, there exists at least one capacity-feasible resource allocation for the work planned and the baseline schedule allows one to identify peak and low capacity requirement periods. The baseline serves as a basis for planning external activities, such as material procurement, preventive maintenance and committing to shipping due dates to customers (Wu *et al.* 1993). Such visibility of future actions is of crucial importance within the inbound and outbound supply chain. Especially in

Revision received October 2003.

†Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven (Belgium).

‡Research assistant of the Fund for Scientific Research – Flanders (Belgium) (F.W.O.-Vlaanderen).

*To whom correspondence should be addressed. e-mail: roel.leus@econ.kuleuven.ac.be

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540310001638055

multi-project environments, a schedule often needs to be sought before the start of the project that is in accord with all parties involved (clients and suppliers, as well as workers and other resources). It may be necessary to agree on a time window for work to be done by subcontractors and to organize production resources to best support smooth schedule execution. Current Internet technology allows companies to share their production schedules with their suppliers on a continuous basis, with the expectation that the suppliers will use this information to provide just-in-time material delivery. A baseline schedule is also vital for cash flow projections and provides a yardstick by which to measure the performance of both management and shop floor personnel. Indeed, reliable baseline schedules enable organizations to estimate the completion times of their projects and take corrective action when needed. They allow for scheduling and resource allocation decisions that in turn should allow quoting competitive and reliable due dates.

During execution, however, the project is subject to considerable uncertainty, which may lead to numerous schedule disruptions. This uncertainty stems from a number of possible sources: activities may take more or less time than originally estimated, resources may become unavailable, material supplies may arrive behind schedule, ready times and due dates may have to be modified, new activities may have to be incorporated or activities may have to be abandoned due to changes in the project scope, etc.

The recognition that uncertainty lies at the heart of project planning induced a number of research efforts in the field of project scheduling under uncertainty (for an extensive review of the literature we refer to Demeulemeester and Herroelen 2002 and Herroelen and Leus 2003a). One research track involves the development of a *proactive baseline schedule* that is protected as well as possible against anticipated schedule disruptions that may occur during project execution. *Reactive scheduling* revises or re-optimizes the baseline schedule when unexpected events occur. The term *predictive-reactive scheduling* has been introduced in the literature to denote the case of a predictive baseline schedule that is developed prior to the start of the project and that may be updated during the project execution phase (Vieira *et al.* 2003).

It is the objective of this paper to offer a review and classification of the procedures for proactive and reactive project scheduling. The next section focuses on the problems and pitfalls of generating deterministic baseline schedules in the presence of uncertainty. Section 3 provides a classification of the various procedures discussed in the literature for generating predictive and reactive project schedules. Section 4 presents a framework that should allow project management to identify a proper scheduling methodology for different project scheduling environments. The last section offers a summary and conclusions.

2. Deterministic baseline scheduling

The literature on deterministic project scheduling under resource constraints mainly focuses on the development of a workable schedule that defines the scheduled start times of the project activities, satisfies both the precedence constraints and the resource constraints, and 'optimizes' the scheduling objective (most often the project duration). As an illustration, consider the project shown in figure 1 in activity-on-the-node format (Wiest and Levy 1977). The project consists of eight real activities (activities 1 and 10 are dummy activities with zero duration). The duration of an activity is shown above the corresponding node. The number shown below a node is the requirement per period (in number of units) for a single renewable resource type.

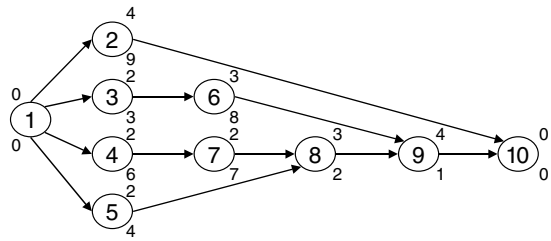

Figure 1. Project example (Wiest and Levy 1977).

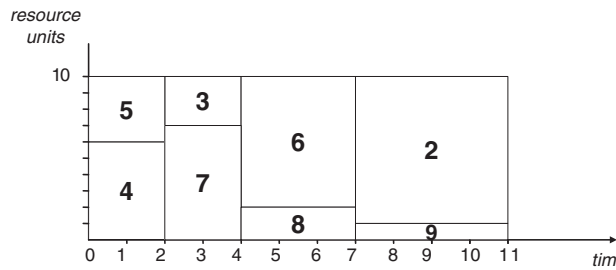

Figure 2. Resource profile.

The arcs in the network denote the finish–start precedence relations with zero time-lag. Figure 2 shows a minimum duration schedule (by pure luck the project duration of 11 periods equals the length of the critical path $<1,4,7,8,9,10>$) that yields a perfectly levelled resource profile with a constant per period resource requirement of 10 units. Stated differently, the schedule shown in figure 2 represents an optimal solution for the corresponding resource-constrained project scheduling problem (minimize the project duration subject to the finish–start precedence constraints and the resource constraints for the single renewable resource with a constant availability of 10 units) and the resource levelling problem (generate a levelled schedule that meets the precedence constraints and completes the project within the given deadline of 11 time periods).

The schedule of figure 2 shows quite a number of so-called critical sequences or critical chains. A critical sequence (Wiest 1964) is a sequence of precedence and/or resource related activities that determines the duration of the corresponding schedule; Goldratt (1997) refers to a critical sequence as a critical chain. The computed schedule has 16 critical sequences, to wit $<1,5,3,6,2,10>$, $<1,5,3,6,9,10>$, $<1,5,3,8,2,10>$, $<1,5,3,8,9,10>$, $<1,5,7,6,2,10>$, $<1,5,7,6,9,10>$, $<1,5,7,8,2,10>$, $<1,5,7,8,9,10>$, $<1,4,3,6,2,10>$, $<1,4,3,6,9,10>$, $<1,4,3,8,2,10>$, $<1,4,3,8,9,10>$, $<1,4,7,6,2,10>$, $<1,4,7,6,9,10>$, $<1,4,7,8,2,10>$, $<1,4,7,8,9,10>$.

The schedule of figure 2 may be optimal for a deterministic project setting, but it is extremely vulnerable to uncertainty. The slightest delay in the starting time of an activity, and/or the slightest increase in the duration of any activity, for example, will lead to an immediate increase in the project duration. The true optimality of the schedule can only be ascertained in conjunction with its execution in the real world. The proposed schedule, considered to be 'optimal' in the project planning phase, clearly has insufficient built-in 'slack', or flexibility for dealing with unexpected

events. In other words, it is not *robust*. In the case of figure 2, this lack of robustness reveals itself as both a lack of stability and of quality. The term *stability* or *solution robustness* refers to the insensitivity of the activity start times to changes in the input data. The term *quality robustness* is used when referring to the insensitivity of the schedule performance in terms of the objective value. Sevaux and Sörensen (2002) refer to 'robustness in the solution space' and 'robustness in the objective function space', respectively. Robustness is closely related to *flexibility*. A schedule is called flexible if it can be easily repaired, i.e. changed into a new high quality schedule. The informal French association of researchers in scheduling GOThA (*G*roupe de recherche en *O*rdonnancement *Th*éorique et *A*ppliqué) has established a 'Flexibility working group' that regularly reflects on how to define, measure and use flexibility, and maintains a web page listing recent references (http://www.loria.fr/~aloulou/pages/biblio_gotha.html). In the next section we discuss the basic methodologies for generating baseline schedules that are quality and solution robust as well as various dynamic scheduling procedures.

3. Generating predictive and reactive project schedules

Table 1 lists various methodologies for baseline schedule generation and reactive scheduling that will be reviewed in this section.

3.1. *Dynamic scheduling using scheduling policies*

The first column in table 1 introduces the question of whether or not a baseline schedule is used. It might be possible that project management decides that no baseline schedule should be generated, but a (full) *dynamic scheduling* procedure is used during the execution of a project to decide which activity to start as time evolves. This is the strategy behind *stochastic project scheduling*. Stochastic project scheduling does not create a baseline schedule but views the problem of scheduling projects under precedence and resource constraints (the *stochastic resource-constrained project scheduling problem*) as a multi-stage decision process, which uses so-called *scheduling policies* (or *scheduling strategies*) that dynamically make scheduling decisions at stochastic decision points corresponding to the start of the project and the completion of activities, based on the observed past and the *a priori* knowledge about the activity processing time distributions. The common objective considered in the literature is to create a policy that minimizes the expected project

baseline schedule		*during project execution*	
no		dynamic scheduling (scheduling policies)	
yes	• no anticipation of variability		• no anticipation of variability
	• *proactive* (*robust*) *scheduling*	• reactive scheduling	• ex post stability
		– schedule repair	
		– full reschedule	
	– quality robust	– contingency scheduling	
	– solution robust	– activity crashing	
		– sensitivity analysis	

Table 1. Methodologies for baseline scheduling and reactive scheduling.

duration over a class of policies (Igelmund and Radermacher 1983a, b). Fernandez (1995), Fernandez *et al.* (1996) and Pet-Edwards *et al.* (1998) show how to write the corresponding optimization problem in its general form as a multi-stage stochastic programming problem. A comprehensive characterization of policies can be found in Möhring *et al.* (1984, 1985) and Stork (2001). Branch-and-bound algorithms to compute optimal policies have been developed by Stork (2001). Heuristic procedures for solving the stochastic resource-constrained project scheduling problem have been developed by Pet-Edwards (1996), Golenko-Ginsburg and Gonik (1997) and Tsai and Gemmill (1996, 1998).

3.2. *Generation of a baseline schedule*

3.2.1. *No anticipation of variability*

A common practice in project scheduling is that management decides to generate a baseline schedule before the start of the project. This can be done using a deterministic project scheduling procedure *without any anticipation of variability*. Single point estimates are used to produce deterministic values for parameters such as activity durations, and the objective is directly related to deterministic project performance. This is the field of *deterministic resource-constrained project scheduling*. For a thorough discussion of the available exact and suboptimal scheduling procedures, we refer the reader to Demeulemeester and Herroelen (2002).

3.2.2. *Proactive (robust) baseline scheduling*

Proactive or *robust scheduling* focuses on the development of a baseline schedule that incorporates a degree of anticipation of variability during project execution. The basic idea is to build protection into the pre-schedule. The field has gained from recent research efforts by both practitioners and theoreticians.

3.2.2.1. *Critical chain scheduling/buffer management.* *Critical chain scheduling/buffer management* (CC/BM) – the direct application of the theory of constraints (TOC) to project management (Goldratt 1997) – has received a lot of attention in the project management literature. The fundamentals of CC/BM are summarized in table 2 (Herroelen *et al.* 2002).

Aggressive median or average activity duration estimates
No activity due dates
No project milestones
No multi-tasking
Scheduling objectives = minimize makespan; minimize WIP
Determine a precedence and resource feasible *baseline schedule*
Identify the critical chain
Aggregate uncertainty allowances into buffers
Keep the baseline schedule and the critical chain fixed during project execution
Determine an early start based unbuffered *projected schedule* and report early completions (apply the roadrunner mentality)
Use the buffers as a proactive warning mechanism during schedule execution

Table 2. CC/BM fundamentals.

CC/BM builds a *baseline schedule* using aggressive median or average activity duration estimates. Activity due dates and project milestones are eliminated and multi-tasking (more than one activity is performed by the same resource unit at the same time) is to be avoided. In order to minimize work-in-progress (WIP), a precedence feasible schedule is constructed by scheduling activities at their latest start times based on critical path calculations. If resource conflicts occur, they are resolved by moving activities earlier in time. The *critical chain* is then defined as the chain of precedence and resource dependent activities that determines the overall duration of a project. If there is more than one such chain, an arbitrary choice is made. The safety in the durations of the activities that are on the critical chain, which was cut away by selecting aggressive duration estimates rather than 'safe' estimates (e.g. 90% quantile), is shifted to the end of the critical chain in the form of a *project buffer* (PB). This project buffer should protect the project due date promised to the customer from variability in the critical chain activities. *Feeding buffers* (FBs) are inserted whenever a non-critical chain activity joins the critical chain. Their aim is to protect the critical chain from disruptions on the activities feeding it, and to allow critical chain activities to start early in case things go well. Although more detailed methods can be used for sizing the buffers (e.g. Newbold 1998), the default procedure is to use the *50% buffer sizing rule*, i.e. to use a project buffer of half the project duration and to set the size of a feeding buffer to half the duration of the longest non-critical chain path leading into it. *Resource buffers* (RBs), usually in the form of an advance warning, are placed whenever a resource has to perform an activity on the critical chain, and the previous critical chain activity is done by a different resource.

The CC/BM baseline schedule for the project of figure 1 would allow for the identification of the 16 critical chains mentioned earlier in section 2. ProChain®, one of the best-known software packages that can be used for implementing CC/BM, selects the critical chain $<1,4,7,8,9,10>$. Using the 50% buffer rule, ProChain® generates the buffered baseline schedule of figure 3, which has a two-period feeding buffer to protect the project buffer from variation in activity 2, a three-period feeding buffer to protect critical chain activity 9 from variation in the path $<3,6>$, and a one-period feeding buffer to protect critical chain activity 8 from variation in activity 5. The reader will observe the rather surprising phenomenon that the critical chain is broken, i.e., it is no longer a contiguous chain that determines the project duration since it contains gaps. Moreover, there is no apparent reason why the 'critical chain' would have a gap between activities 7 and 8 and activities 8 and 9. A resource buffer

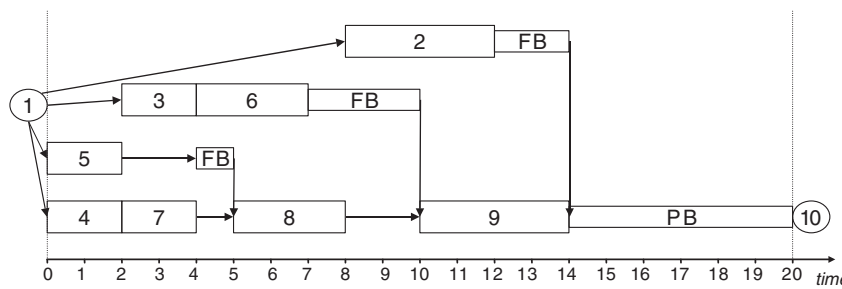

Figure 3. Buffered baseline schedule for the project of figure 1.

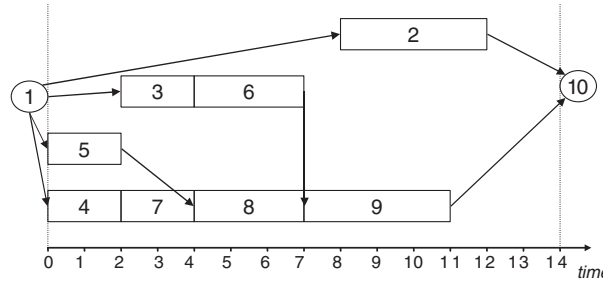

Figure 4. Projected schedule.

is inserted in front of activity 7 to give a warning signal to the extra resource unit needed for the execution of critical chain activity 7, but this is not indicated in figure 3.

For executing a project, on the other hand, the CC/BM approach does not rely on the buffered schedule but on a so-called *projected schedule*. The projected schedule is precedence and resource feasible, contains no buffers and is to be executed according to the roadrunner mentality, i.e. the so-called gating tasks (activities with non-dummy predecessors) are started at their scheduled start time in the buffered schedule while the other activities are started as soon as possible. Figure 4 shows the projected schedule corresponding with the buffered schedule of figure 3. As can be seen, activities 8 and 9 are left-shifted allowing the gaps in the critical chain to be closed, while the start time of gating task 2 is left unchanged. The projected schedule is recomputed when distortions occur. Similarly to the unprotected minimum duration baseline schedule shown in figure 2, this is all but a stable schedule. Therefore, using the schedule for the purposes we indicated earlier (section 1) will prove virtually impossible and the same is true for the baseline schedule in figure 3.

Herroelen and Leus (2001) have validated the working principles of CC/BM through a full factorial computational experiment using the well-known 110 Patterson test problems (Patterson 1984). They reach the conclusion that (a) updating the baseline schedule and the critical chain at each decision point provides the best intermediate estimates of the final project duration and yields the smallest final project duration, (b) using a clever project scheduling mechanism such as branch-and-bound has a beneficial effect on the final makespan, the percentage deviation from the optimal final makespan obtainable if information would be perfect, and the work-in-progress, (c) using the 50% rule for buffer sizing may lead to a serious overestimation of the project buffer size, (d) the beneficial effect of computing the buffer sizes using the root-square-error method increases with problem size, (e) keeping the critical chain activities in series is harmful to the final project makespan, and (f) recomputing the baseline schedule at each decision point has a strong beneficial impact on the final project duration. Herroelen *et al.* (2002) have studied the practical implications of the scheduling procedure and warn against oversimplifications of the approach.

In a multi-project environment, CC/BM relies on the common steps of TOC, applied as follows:

1. Prioritize the organization's projects;
2. Plan the individual projects according to the CC/BM fundamentals;
3. Stagger the projects;

4. Insert drum buffers;
5. Measure and report the buffers;
6. Manage the buffers.

Step 1 aims to avoid multi-tasking among projects. Step 2 calls for the identification of the most constraining company resource as the 'bottleneck' (strategic or drum resource). Projects are staggered (Step 3) on the basis of the schedule for the strategic resource (drum plan). This is done by placing a capacity buffer in front of a strategic resource activity that is on the critical chain in the strategic resource schedule. A drum buffer (Step 4) is placed before activities on the strategic resource to protect the strategic resource from disruptions on non-strategic resources. The buffers then can be managed more or less the same way as done in the single-project case (Steps 5 and 6).

3.2.2.2. *Robust precedence feasible schedules*

1. Solution robust schedules in the absence of resource constraints. Herroelen and Leus (2003b) develop mathematical programming models for the generation of *stable* baseline schedules under the assumption that the proper amount of resources can be acquired if booked in advance based on the pre-schedule and that a single activity disruption (duration increase) may occur during schedule execution (see also Leus 2003 who discusses extensions to the multiple disruption case). They use as stability measure the expected weighted deviation of the start times in the schedule realized after project execution from those in the pre-schedule. They derive a linear programming model, the dual of which corresponds to a minimum cost network flow problem, which can be solved efficiently.

The procedure can be applied to the project network of figure 1, assuming, for example, the following parameters: a project deadline of 14 periods (for comparison reasons set equal to the CC/BM project length shown in figure 3), equal disruption probabilities for the activities, and disruption scenarios specifying that when an activity is disrupted there is 50% chance that its duration is increased by 1 period and 50% chance that its duration is increased by 2 periods. The resulting proactive baseline schedule is shown in figure 5. The schedule is *solution robust* to the described disruption setting (the weighted expected deviation between the activity start times in the baseline schedule and the schedule realized during project execution has been minimized). It has been generated under the assumption that a proper resource allocation can be performed that provides sufficient resource units to execute the activities planned in each period of the schedule horizon, and that on schedule

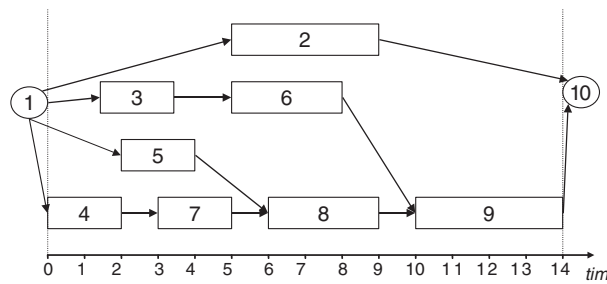

Figure 5. Solution robust baseline schedule without resource constraints.

implementation, activities are not started earlier than planned, in order to respect the initial schedule as much as possible, in order to maximally exploit its advantages set out in the introduction. If the 10 unit resource constraint mentioned earlier were invoked for the project, the schedule would not be resource feasible (there would be a peak resource requirement of 19 resource units in periods 7 and 8).

We notice that a stable schedule attempts to spread out the activities across the scheduling horizon, such that small disruptions in activity durations are smoothed out and do not propagate through the network, and the basic functions of a schedule as described in section 1 can be optimally exploited. The corresponding objective function value for the CC/BM based projected schedule in figure 4 will be at least twice the value of the schedule in figure 5, in light of the choice for project end at time 12 and the evolution of the curve in figure 6. We now ask the reader to have a look at figure 7: these are the observed density functions of the project makespan on simulation of the project, when the initial schedule is either figure 4 or figure 5 and the actual activity durations are stochastic variables that behave according to a triangular distribution with mode equal to the initial deterministic duration estimate, minimum equal to half this value, and maximum equal to twice the initial estimate. The maximum makespan is seen to be 22, the minimum is 10. We disregard resource

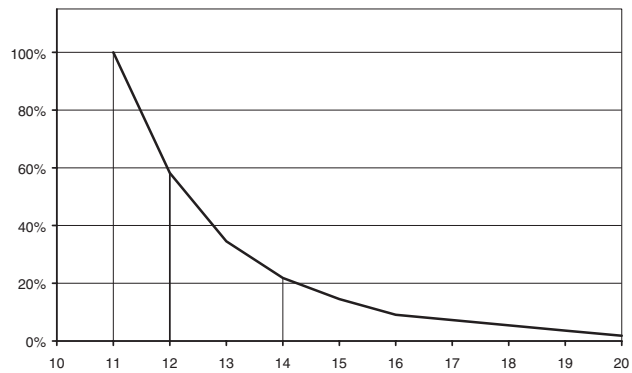

Figure 6. Evolution of the expected weighted deviation objective as a function of the deadline, as a percentage of the score for the case where the deadline is 11 (critical path length).

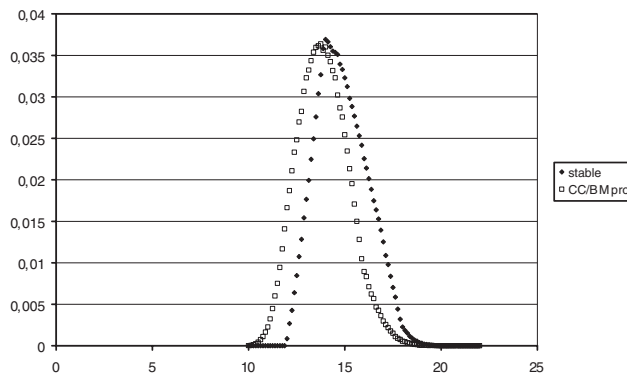

Figure 7. Observed density functions.

constraints for the moment. Protection of a committed makespan of 20 time units as provided by the end of the project buffer in figure 3 is not severely compromised by switching from figure 4 to figure 5, but we nevertheless notice that the probability mass of the stable schedule is shifted to the right (higher makespan) compared with the projected schedule. We are therefore faced with a trade-off between makespan and stability of a schedule. We will provide guidelines for resolution of this trade-off in section 4.

Tavares *et al*. (1998) study the risk of a project as a function of the uncertainty of the duration and the cost of each activity and the adopted schedule. They increase the earliest activity start times by the product of the total float of the activity and a float factor α, $0 \leq \alpha \leq 1$, and prove that the adapted start times yield a feasible schedule. Herroelen and Leus (2003b) have adapted the float factor model allowing the float factor to vary among the project activities, in order to pursue stability in the schedule. They have also adapted the linear programming based heuristic (LPH), developed by Mehta and Uzsoy (1998) to insert idle time in a job shop schedule, to a project environment. They demonstrate that both the adapted float factor and LPH procedures are clearly outperformed by the network flow based procedure discussed earlier.

The discussion in this section so far only concentrated on one type of schedule disruption that may occur during project execution, to wit an increase in the duration of project activities. Obviously, other types of schedule disruptions do exist and should be anticipated in a proactive baseline schedule: decreases in activity duration (activities finish earlier than predicted), changes in the execution mode of the activities (planned or unplanned activity pre-emption, choice of a different method or technique for executing one or more activities having a duration impact), delays in the starting times of activities, modification of the structure of the project (e.g. new activities have to be inserted in the baseline schedule, changes in the structure of the precedence relations), etc. Incorporating these different types of schedule disruptions in the analysis constitutes a viable area of future research. The discussion so far also assumed that the generation of the baseline schedule could be done without taking into consideration the requirements and availabilities of various types of resources. The generation of resource feasible proactive baseline schedules will be discussed in section 3.2.2.3.

Moreover, the stability measure referred to in this section – the expected deviation in activity start times in the baseline schedule and the schedule realized during project execution – is only one possible stability measure. Other metrics for measuring the deviation between a baseline and realized schedule may be used: the number of activities that were distorted, the number of times that an activity is re-planned, etc. Again, the generation of solution robust project schedules under various stability metrics constitutes a viable area of future research.

2. Quality robust schedules. The discussion in (1) above focused on the generation of stable or solution robust schedules. Quality robust schedules aim at maximizing quality robustness, i.e. the insensitivity of the schedule to disruptions that affect the value of the performance metrics used to evaluate the quality of the schedule. The metrics may relate to the *average quality robustness*, i.e. the expected difference between the optimal value of the objective function (e.g. the optimal makespan) realized on the problem instance and the makespan realized by the application of the proactive and reactive scheduling algorithms. The average quality

robustness metric may be adapted to express the expected quality robustness that holds for all possible reactive scheduling algorithms applied to a given proactive baseline schedule. Expected quality robustness does not guarantee the schedule performance for each realization of the schedule. *Worst case quality robustness* is an absolute guarantee that a schedule performance will be obtained (GOThA 2002). The worst case robustness metric may be the maximum deviation over all possible schedules obtainable for a proactive scheduling algorithm between the performance of the proactive schedule and the optimal performance of the realized schedule measured over all possible reactive scheduling algorithms. The worst case metric may relate to the deviation between a negotiated performance value and the performance obtained by the proactive and reactive scheduling algorithms, or it may relate to the probability that the performance (e.g. makespan) obtained by a predictive-reactive schedule stays below a certain threshold value.

For the case without resource constraints, optimizing the average expected makespan as well as the worst case makespan performance is easy because the solution is always the earliest start schedule.

3.2.2.3. *Solution and quality robust schedules in the presence of resource constraints.* Consider the baseline schedule of figure 8 derived for the problem example of figure 1 under the renewable resource availability constraint of 10 units per period, a project deadline set to 14 and the same disturbance scenario as mentioned above. The schedule is precedence and resource feasible.

In comparison with the buffered CC/BM schedule of figure 3 and the projected schedule of figure 4, the schedule has gained stability (solution robustness). For example, a distortion in activity 3 in the schedules of figures 3 and 4 has an immediate impact on the start time of activity 6, while this is not the case in the schedule of figure 8: activity 3 can be right-shifted two periods without violating the precedence and/or resource constraints. Similarly, while a distortion of activity 4 or 5 in the schedule of figure 4 immediately affects the start times of activities 3 and 7, this will not be the case in the protected baseline schedule of figure 8: both activities can have a one-period precedence and resource feasible right-shift.

Simulation can again be invoked to compare different schedules as done in section 3.2.2.2, but in the presence of stringent resource constraints, a number of assumptions need to be submitted with regard to the reactive policy that will be adhered to during project execution (cf. section 3.3). The same is true in fact for the unconstrained case, where we implicitly assumed a simple right-shift policy as best alternative.

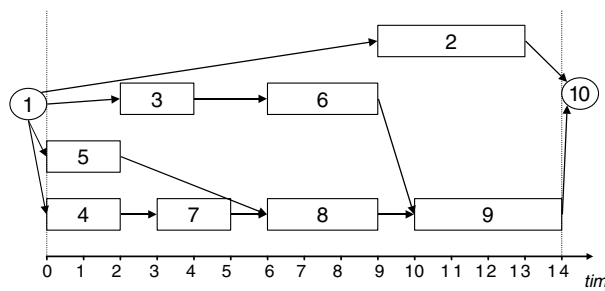

Figure 8. Resource feasible, protected baseline schedule.

Given a protected precedence and resource feasible baseline schedule such as the one shown in figure 8, a crucial issue remains to be solved: the precise *resource allocation* among the various activities. Leus and Herroelen (2003) have studied the problem of generating a robust resource allocation under the assumption that a feasible baseline schedule exists and that some advance knowledge about the probability distribution of the activity durations is available. It is also assumed that the resource allocation is not changed during project execution. They explore the fact that checking the feasibility of a resource allocation solution can easily be done using maximal flow computations in a transformed precedence network and develop a branch-and-bound procedure enhanced with constraint propagation that solves the robust resource allocation problem in exact and approximate formulations.

In the schedule of figure 8, activities 4 and 5 are scheduled in parallel using six and four resource units in each of the periods 1 and 2, respectively. Activity 3 needs three resource units in periods 3 and 4, while activity 7 needs seven resource units in periods 4 and 5. Management should now study the uncertainty involved in the execution of activities 4 and 5. If activity 4 is deemed less risky than activity 5, it should pass upon its completion three of its six allocated resource units to activity 3 and the remaining three resource units to activity 7, which receives the remaining four resource units from activity 5. If activity 5 is deemed less risky than activity 4, it should on its completion pass three of its resource units to activity 3 and the remaining unit to activity 7. The latter activity can then obtain six remaining units from activity 4 upon the completion of that activity. A similar argument can allow for a proper resource allocation from activities 6 and 8 to 2 and 9. If the anticipated risk of activity 6 is deemed smaller that the risk associated with activity 8, it should pass eight resource units to activity 2 so that this activity can obtain the remaining resource unit from activity 8, which passes its other resource unit to activity 9.

The remarks made in the previous section on the use of other solution and quality robust metrics and different types of schedule disruptions also hold for the resource-constrained environment discussed in this section.

3.3. *Reactive scheduling*

Reactive scheduling refers to the schedule modifications that may have to be made during project execution. The use of a baseline schedule in combination with reactive scheduling procedures is sometimes referred to as *predictive-reactive scheduling*, which is to be contrasted with completely reactive scheduling discussed in section 3.1, which dispatches activities on-line or real-time and does not use a pre-schedule. The reactive scheduling action may be based on various underlying strategies. At one extreme, the reactive effort may rely on very simple techniques aimed at a quick schedule consistency restoration. We refer to these approaches as *schedule repair*. At the other extreme, the reactive scheduling approach may involve a full scheduling pass of that part of the project that remains to be executed at the time the reaction is initiated. Such an approach is referred to as (*full*) *rescheduling*.

3.3.1. *Schedule repair*

Schedule repair may involve simple control rules such as the well-known *right-shift rule* (Sadeh *et al*. 1993, Smith 1994). This rule moves forward in time all the activities that are affected by the schedule breakdown because they were executing on the resource(s) causing the breakage or because of the precedence relations. It should

be clear that this reactive strategy can lead to poor results, as it does not re-sequence activities.

3.3.2. *Rescheduling*

Rescheduling may use any deterministic performance measure, such as the new project makespan (in manufacturing systems, this approach is sometimes referred to as (complete) regeneration). In a sense, schedule repair can be viewed as a heuristic rescheduling pass. An extensive simulation experiment exploring the advantages of project rescheduling for makespan minimization is described in Yang (1996).

Artigues and Roubellat (2000) study the case where, in a multi-project, multi-mode setting with ready times and due dates, it is desired to *insert a new unexpected activity into a given baseline schedule* such that the resulting impact on maximum lateness is minimized. The authors perform a clever rescheduling pass in which they restrict the solution to those schedules in which the resource allocation remains unchanged. Using a resource flow network representation they develop a stepwise procedure for generating a set of dominant insertion cuts for the network. From each dominant insertion cut, they then derive the best execution mode and valid insertion arc subset included in the dominant insertion cut. The procedure strikes a balance between simple schedule repair and full rescheduling. The authors have validated their polynomial insertion algorithm on the 110 Patterson test problems (Patterson 1984) against complete rescheduling using the MINSLACK priority rule within a serial schedule generation scheme. In terms of computational burden the insertion method clearly outperforms the rescheduling method. The mean increase of the makespan of the schedule with the inserted activity above the makespan of the baseline schedule stays below the duration of the activity to be inserted. Moreover, the mean makespan increase is smaller for the insertion algorithm.

In case of rescheduling, the new schedule can differ considerably from the baseline schedule and this is not always desirable, a fact that was already referred to before. Frequent schedule regeneration can result in *instability* and lack of continuity in detailed plans, resulting in increased costs attributable to increased shop floor *nervousness*. Some sources prefer heuristic repair actions that change schedules locally to global regeneration of the schedule, because only limited changes are applied to the input schedule. Alternatively, the rescheduling pass can explicitly aim to generate a new schedule that deviates from the original schedule as little as possible. Such a *minimum perturbation strategy* aims at ex post stability, and relies on the use of exact or suboptimal algorithms using as objective the minimization of a function of the differences between the start time of each activity in the new and original schedule (El Sakkout and Wallace 2000), or the minimization of the number of activities to be performed on different resource units (Alagöz and Azizoglu 2003), etc. Calhoun *et al*. (2002) make a distinction between *re-planning* (fixing the schedule before the start of the work period) and *re-scheduling* (re-assigning tasks and resources during the work period). The overall problem is formulated as a goal programming model for the generalized multi-mode resource-constrained project scheduling problem, where goal i corresponds to scheduling the activities in priority class i with the available resources and within the generalized precedence constraints, and minimizing the makespan is added as the lowest priority goal. They use a heuristic to provide an initial solution that is subsequently improved using a tabu search procedure. Adding the minimum number of changed activities as an extra

goal they offer a tabu search procedure for re-planning and for re-scheduling the activities that are not locked in time.

In pursuit of re-scheduling stability, algorithms have also been proposed that use a *match-up* point, i.e. the time instance where the state reached by the revised schedule is the same as the initial schedule (Akturk and Gorgulu 1999, Bean *et al.* 1991, Wu *et al.* 1993). The idea is to follow the pre-schedule if no disruption occurs. The goal is to match up with the pre-schedule at a certain time in the future, whenever a deviation from the initial parameter values (mainly deviations from the activity duration projections) arise.

3.3.3. *Contingent scheduling*

It may happen in practice that management makes manual changes to the schedule during project execution. A number of algorithms have been developed to aid the manager in making this type of decision. Billaut and Roubellat (1996a, b) suggest generating for every resource a so-called *group sequence*, i.e. a totally or partially ordered set of groups of operations, and to consider all the schedules obtained by an arbitrary choice of the ordering of the operations inside each group. In this way the decision maker is not provided with one feasible schedule but with several ones. The hope is that during the real-time execution of the schedule, it becomes possible to switch from one solution to the other in the presence of a disruption without any loss in performance. Maugière *et al.* (2002) and Aloulou *et al.* (2002) explore this sequence *flexibility* idea in the context of single machine scheduling. Briand *et al.* (2002) extend the methodology used by Billaut and Roubellat (1996b) to the case of multi-mode scheduling with minimal and maximal time-lags. Artigues *et al.* (1999) study multi-mode project scheduling problems where the projects have a release date and a due date. They propose a generation procedure for finding group sequences based on a new priority rule. They also propose and test an efficient local search procedure to improve the feasibility of a group sequence.

3.3.4. *Activity crashing*

It should be mentioned that during project execution corrective actions may be taken by crashing some (or maybe all) activity durations. A number of exact and sub-optimal procedures have been developed for solving the associated time/cost trade-off and time/resource trade-off problems, as well as the multi-mode and multiple crashable modes problems and mode identity problems (for an extensive discussion we refer to Chapter 8 in Demeulemeester and Herroelen 2002).

3.3.5. *Sensitivity analysis*

An interesting area of research is the sensitivity analysis of project schedules. Sensitivity analysis addresses 'What if...?' types of questions that arise from parameter changes. Hall and Posner (2000a, b) have studied polynomially solvable and intractable machine scheduling problems and try to answer a number of fundamental questions such as (a) what are the limits to the change of a parameter such that the solution remains optimal?, (b) given a specific change of a parameter, what is the new optimal cost?, (c) given a specific change of a parameter, what is a new optimal solution?, etc. An interesting area of further research is to pose similar questions in a project scheduling environment.

Another interesting research track is the establishment of *sensitivity guarantees* of scheduling algorithms. Penz *et al.* (2001) determine the sensitivity guarantee of

off-line single and parallel machine scheduling algorithms. For a problem instance I and a minimization objective function f, the performance guarantee of scheduling algorithm ALG for problem instance I, $\rho_{ALG} = \sup_I\{\rho_{ALG}(I)\}$, where $\rho_{ALG}(I) = f_{ALG}(I)/f_{OPT}(I)$ is the theoretical or off-line performance ratio of algorithm ALG for problem instance I, and $f_{ALG}(I)$ and $f_{OPT}(I)$ represent the objective value achieved on problem instance I by algorithm ALG and the optimal objective value, respectively. Variability in the project environment will inevitably influence the activity and problem characteristics and is captured in vector $\vec{\varepsilon}$. The *effective performance ratio* obtained after schedule execution is then $\rho^{\vec{\varepsilon}}_{ALG}(I) = f^{\vec{\varepsilon}}_{ALG}(I)/f^{\vec{\varepsilon}}_{OPT}(I)$. The numerator and denominator in the right-hand side of the expression represent the objective value with perturbations $\vec{\varepsilon}$ of the ALG schedule for I, and the optimal value ex post, with perfect knowledge, respectively. The *sensitivity guarantee* of an off-line algorithm ALG is a function $s_{ALG}(\varepsilon)$ such that for any off-line instance I and any perturbation $\vec{\varepsilon}$ with $\|\vec{\varepsilon}\| \leq \varepsilon$, $s_{ALG}(\varepsilon)$ is the smallest real value satisfying $\rho^{\vec{\varepsilon}}_{ALG}(I) \leq s_{ALG}(\varepsilon)\rho_{ALG}(I)$.

Sotskov (1991) and Sotskov *et al.* (1997) consider the problem of scheduling a job shop with minimum makespan and other objectives. A schedule is represented by an orientation of arcs in the disjunctive graph representation and it is studied for what deviations in activity durations the chosen orientation of arcs remains optimal, ex post. The distance between duration vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ is assessed by means of the Chebyshev metric $r(\mathbf{d}_1,\mathbf{d}_2) = \max_i|d_{1i}\text{-}d_{2i}|$. The stability radius of an optimal solution s for durations $\mathbf{d}$ is then defined as the maximum value of the radius ρ such that s remains optimal for all duration vectors $\mathbf{d}$' with $r(\mathbf{d},\mathbf{d}') \leq \rho$. The authors give computational results on small job shop instances and conclude that calculation of the stability radius is very complicated and time consuming. We point out to the reader that the term 'stability' has a different meaning in this context.

Finally, we should point out that the use of sensitivity analysis for general decision making under uncertainty has been the subject of critique. We refer the reader to Wallace (2000), who stresses that flexibility options are not appropriately recognized when using deterministic models with sensitivity analysis. Wallace points out that the technique is appropriate when analysing allowable variation in controllable parameters.

4. Different approaches to the multi-project scheduling problem

The generation of quality and solution robust schedules is crucial in a multi-project environment. The state of the art was reviewed in the previous section. Clearly no single best method for managing an arbitrary multi-project organization exists: the best way to coordinate and schedule resources and control schedule performance depends on the project environment. While it may not make sense to identify one specific scheduling methodology based on limited information, table 3 does compile some hints with respect to how to handle the project scheduling problem. We distinguish two key determinants: the degree of general variability in the work environment and the degree of independence of the project.

4.1. *Dependency versus variability*

The *variability* factor involves a joint impression of the uncertainty and variability associated with the size of the project parameters (time, cost, quality), uncertainty about the basis of the estimates (activity durations, work content), uncertainty about the process (what is to be done, how, when, by whom and at

	totally dependent	*rather dependent*	*rather independent*	*totally independent*
low variability	7 stable plan; satisficing	5 stable drum; efficient remainder	3 efficient drum; efficient remainder	1 deterministic
high variability	8 process mgmt.; rough plan with sufficient slack	6 stable plan with queuing	4 stable drum; dispatch or predictive-reactive	2 dispatch or predictive-reactive

Table 3. Different approaches to the (multi-)project scheduling problem.

what cost), uncertainty about the objectives, priorities and acceptable trade-offs, and uncertainty about fundamental relationships between the various project parties involved (responsibilities, communication, contractual conditions and effects, mechanisms of coordination and control). Ward and Chapman (2003) suggest that management should strive for a shift from a threat focus towards greater concern with understanding and managing all sources of uncertainty, and explore and understand the origins of uncertainty before seeking to manage it.

Projects are said to be *dependent* if they need to coordinate with non-project parties, be they internal or external to the organization. This refers to both shared resources as well as dependence on outside contractors. The degree of shared resources has been referred to by Hendricks *et al*. (2002) as the *project scatter factor*: it measures to what extent projects consist of full-time members and the project team constitution is fixed. Hendricks *et al*. point out that the resource dedication profile (the degree of specialization of the resources, from all-round employees to experts in a certain matter) strongly influences the project scatter factor. Complementarily, we see dependence as the degree in which free dispatching or scheduling of activities is possible. Therefore, the use of a certain number of intermediate milestones in the project also increases dependence (tight due dates), and so does the dependency on unreliable suppliers (uncertain or tight ready times). We refer to *drum activities* as all activities that are dependent: either they are performed by shared internal or external resources, or their start or completion time is constrained. In the following, we refer to the remaining project activities as the *remainder*.

It is imperative that the reader bear in mind that we assume that swift project completion is always a key objective for the project. At the same time, on a higher level, it is profitability, manageability and control of the portfolio of projects that is an objective for senior management, and this will sometimes preclude scheduling or dispatching solely with makespan objective. Let us now turn to some case by case comments, referring to the labelled cells in table 3.

1. *Low variability, totally independent project*. In this extreme case, a project can be executed with fully dedicated resources and without outside restrictions. A *deterministic schedule* generated under well-known regular (e.g. makespan) or non-regular deterministic project scheduling objectives (e.g. net present value) can be used as the baseline schedule (see section 3.2.1). This baseline schedule will be the subject of only minor disruptions.
2. *High variability, totally independent project*. A detailed schedule covering the entire project will be subject to a high degree of uncertainty during execution. The project can be executed according to some *scheduling policy* (comparable

with the 'roadrunner' mentality): the resources are always 100% available to the project, so *dispatching* is possible in order to complete the project as quickly as possible (see section 3.1). Alternatively, a *predictive-reactive* approach can be followed (sections 3.2.2 and 3.3). Dispatching is preferable in cases of very high uncertainty; for moderate uncertainty we advise a predictive-reactive method.

3. *Low variability, rather independent project*. If a regular objective such as project duration is used, a feasible schedule for the drum activities is generated, and in this drum plan, no large provision for uncertainty needs to be made. The drum activities can be scheduled for *efficiency* and *solution quality*. The remaining (independent) tasks can then be planned around this drum plan, for example with minimal makespan as primary objective. If uncertainty and complexity are limited, one scheduling pass per project can suffice. Since projects are normally added to the portfolio dynamically over time and have to be aligned with the existing schedules of the current portfolio, a single integrated multi-project scheduling pass will sometimes already produce difficulties.
4. *High variability, rather independent project*. The *drum plan* should now be generated for *robustness* in light of the high amount of variability. The remaining activities can be dispatched as a function of the progress on the drum, or a predictive-reactive approach can be followed; the same remark as in case 2 can be made with regard to the choice between these two approaches.
5. *Low variability, rather dependent project*. In this type of environment, a large number of resources are shared, and/or a large number of activities have constrained time windows. A *robust plan* should be set up for these activities, such that small disruptions do not propagate throughout the overall plan. The remainder should be planned in as efficiently as possible.
6. *High variability, rather dependent project*. Given the high variability and the high degree of dependence, a *robust drum plan* should be set up that is not overly detailed, thus leaving sufficient opportunity for adapting to uncertain events. The bottleneck resources will probably have a queue of jobs waiting to be processed. The 'independent' activities are now completely dependent on the (uncertain) execution time of the drum activities.
7. *Low variability, totally dependent project*. Since the degree of uncertainty in this type of environment is not very high, the generation of an *aggregate plan* seems to be sufficient to handle this type of situation. The goal of resource allocation will be to obtain a feasible plan with a minimal number of conflicts (a 'satisficing' pass). Where possible, small amounts of slack should be integrated into the plan, in order to smooth out small disruptions.
8. *High variability, totally dependent project*. This extreme environment refers to a situation that is best dealt with from a process management viewpoint: the resources are often workstations that are visited by (or visit) work packages and pass these on to the appropriate successor resources after completion. A *rough ballpark* plan can be constructed to come up with intermediate milestones, which can be used for setting priorities for the resources in choosing the next work package to consider.

When variability is low, an integrated deterministic plan is set up, while high variability makes it difficult to predict the exact starting times of (a number of) activities before the start of the project. Single-project management techniques can be applied in cases 1 and 2. The majority of the project scheduling research efforts relate to these categories. In cases 6 and 8, tight milestones will induce a permanent 'fire-fighting' mode, since lead times are hard to estimate and strongly depend on the current organizational load. The size of the projects can influence the appropriate methodology: when individual projects are large and complex, the organization will tend to operate within an environment that is described in the left columns of the table. Lack of coordination or insufficient stability will entail 'switching' from the top row to the bottom row, since management is unable to control the variability in the organization.

4.2. *The role of CC/BM*

Critical chain project management (CC/BM), the application of the theory of constraints to project management, has been discussed in section 3.2.2.1. It is an integrated methodology for project planning and execution, combined with a number of related work-organization issues. The methodology has become a scheduling hype but certainly needs to be credited for the attention it draws to duration estimation problems, Parkinson's law, the student syndrome and multi-tasking. For a single-project environment, the methodology seems practical and well thought-out. It has been devised for application especially in environment 2 in table 3, but it imposes extra constraints on project execution in order to facilitate makespan estimation, avoid undesirable behaviour of workers and assist in problem detection. Nevertheless, for single projects, the unconditional focus on a 'critical chain' seems useless to us: it obscures extra scheduling options, and enforces a rigid focus on what was critical at the start of the project but may no longer be crucial after a certain lapse of time. Makespan estimation is approximate anyway because of the merge bias and the simplification of the resolution of resource conflicts. Practical makespan estimations can be obtained based on scenario analysis, and/or the incorporation of an aggregate protection measure under the form of a project buffer alone. The experience of the scheduler will have the final word. The durations are often based on the behaviour of human resources, so one should not rely on overly sophisticated statistical techniques for modelling them (for one, because their variability can be influenced by management).

Rescheduling is disapproved of by the majority of CC/BM adepts. In part, this is because the combinatorial aspects of scheduling are not always understood: CC/BM resorts to 'levelling' and 'shifting' for elimination of resource conflicts, and the use of 'as late as possible schedules' for initial planning and 'as soon as possible schedules' for project execution does not facilitate interpretation. It is also claimed that scheduling in itself is not necessary, because of the 'dispatching' character of CC/BM. Nevertheless, multiple CC/BM sources, not in the least the software vendors, speak of 'scheduling driven management' and 'aggressive schedules', which is confusing. Rescheduling is said to be harmful to stability, but it follows from an investigation of the detailed techniques of CC/BM, both from the available literature and from the existing software, that a 'projected schedule' (section 3.2.2.1) is unavoidable, which needs to be rescheduled anyway. Stability within separate projects is far from guaranteed, given the fact that CC/BM boils down to dispatching based on a constantly rescheduled projected schedule, although many of the proponents of the methodology

do not see this and startle at the term 'reschedule', condemning it for introducing system nervousness.

Being based on dispatching and thus geared towards high uncertainty, CC/BM seems nevertheless not adapted to environments with very high uncertainty such as new product development, because of the iterative nature of many tasks (so called 'loops') or the unforeseen addition of new tasks: this uncertain evolution structure of the projects certainly requires frequent baseline rescheduling.

On the other hand, the *drum schedule* as conceived in multi-project CC/BM does introduce a certain degree of stability across the various projects in progress. Resource allocation implemented as permanent additional precedence constraints (Leus and Herroelen 2003) does make sense for bottleneck resources and other shared resources in multi-project environments. Nevertheless, the critical chain approach falls short of covering the scheduling needs of every multi-project organization; it applies especially to environment 4 in table 3. Raz and Barnes (2001) state that CC/BM 'deals with a multi-project environment by staggering the projects around the constraining resource. In principle, at any given point in time, there could be several constraining resources, each leading to a different schedule. The premise that there is a single constraining resource seems more applicable to manufacturing and operations environments than most project environments.' It would make sense to distinguish the shared resources with high load, of which there will regularly be more than one type, and build a stable schedule for these first. In this schedule, different projects can be prioritized, in order to avoid multi-tasking at project level. The drum resources should be booked ahead of time based on time windows for their approximate period of need, which in turn can be based on 'rough cut' single project schedules. This may be based on 'backward' scheduling from a due date, but much detail may not be needed. In the presence of a single bottleneck, this description roughly follows the multi-project CC/BM approach: the goal of the *drum* buffer seems to be to assure stability. Afterwards, the individual projects can be fine-scheduled, and the objective then will mostly be to minimize makespan. However, the single projects need not necessarily contain a 'critical chain'. This second scheduling pass deserves more attention than the simple 'staggering' advocated by CC/BM, which may well result in low throughput if gone about overly simplistically. In the critical chain methodology, the result of as late as possible baseline scheduling and as soon as possible execution is that tasks are left-shifted *within* the original schedule horizon, and subsequent staggering aims to avoid multi-tasking at project level. Nevertheless, especially in low variability environments, it may make sense for both stability and throughput reasons to spread out those tasks further, and disregard the 'latest' safe start dates for the gating tasks altogether. Additionally, a considerable number of internal process- and product-development projects will not have clear due dates, and they will need to be scheduled in a 'forward' manner right from the start. Thus we increase throughput of projects by multi-tasking those projects, which is not bad per se, only should it be avoided to overload the system, since then throughput times increase in a non-linear fashion. Decisions about the resolution of multitasking should be made in function of system capacity and degree variability.

5. Summary and conclusions

It was the objective of this paper to review the methodologies for proactive and reactive project scheduling and to present some hints that should allow project

management to identify a proper project scheduling methodology for different project scheduling environments. Research efforts aimed at generating solution and quality robust schedules in combination with an effective reactive scheduling mechanism are still in the burn-in phase. The generation of solution robust schedules in the absence of resource constraints and the allocation of resources has been achieved for activity duration disruptions under the objective of minimizing the expected deviation in planned and realized activity start times. The critical chain scheduling methodology has attracted a lot of attention, is definitely an important eye-opener, but suffers from a serious oversimplification of the issues involved and is not universally applicable. Our on-going research involves the extension of the methodology for generating quality robust schedules to other types of schedule disruptions and other stability measures in the presence of resource constraints. The literature on the generation of precedence and resource feasible schedules that are both solution and quality robust is void, and constitutes an important area of future research.

References

Akturk, M. S. and Gorgulu, E., 1999, Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, **112**, 81–97.

Alagöz, O. and Azizoglu, M., 2003, Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research*, to appear.

Aloulou, M. A., Portmann, M.-C. and Vignier, A., 2002, Predictive-reactive scheduling for the single machine problem. Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia, 3–5 April, 39–42.

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

Artigues, C., Roubellat, F. and Billaut, J.-C., 1999, Characterization of a set of schedules in resource-constrained multi-project scheduling problem with multiple modes. *International Journal of Industrial Engineering – Theory, Applications and Practice*, **6**(2), 112–122.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S. and Uzsoy, R., 2003, Executing production schedules in the face of uncertainties: a review and some future directions. *European Journal of Operational Research*, to appear.

Bean, J. C., Birge, J. R., Mittenthal, J. and Noon, C. E., 1991, Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research*, **39**(3), 470–483.

Billaut, J. C. and Roubellat, F., 1996a, A new method for workshop real time scheduling. *International Journal of Production Research*, **34**(6), 1555–1579.

Billaut, J .C. and Roubellat, F., 1996b, Characterization of a set of schedules in a multiple resource context. *Journal of Decision Systems*, **5**(1–2), 95–109.

Briand, C., Despontin, E. and Roubellat, F., 2002, Scheduling with time lags and preferences: a heuristic. Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia, 3–5 April, 77–80.

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

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

El Sakkout, H. and Wallace, M., 2000, Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, **5**(4), 359–388.

Fernandez, A. A., 1995, The optimal solution to the resource-constrained project scheduling problem with stochastic task durations. Unpublished doctoral dissertation, University of Central Florida.

FERNANDEZ, A. A., ARMACOST, R. L. and PET-EDWARDS, J., 1996, The role of the non-anticipativity constraint in commercial software for stochastic project scheduling. *Computers and Industrial Engineering*, **31**, 233–236.

GOLDRATT, E., 1997, *Critical Chain* (Great Barrington: The North River Press).

GOLENKO-GINSBURG, D. and GONIK, A., 1997, Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics*, **48**, 29–37.

GOTHA, groupe de flexibilité, 2002, Flexibilité et robustesse en ordonnancement. Bulletin de la ROADEF, **8**, 10–12. Also available at http://www.roadef.org/lf/flexordo.ps

HALL, N. and POSNER, M., 2000a, Sensitivity analysis for intractable scheduling problems. Research paper, Ohio State University.

HALL, N. and POSNER, M., 2000b, Sensitivity analysis for efficiently solvable scheduling problems. Research paper, Ohio State University.

HENDRICKS, M. H. A., VOETEN, B. and KROEP, L. H., 2002, Human resource allocation in a multiproject research and development environment. In PENNYPACKER, J. S. and DYE, L. D. (eds.), *Managing Multiple Projects. Planning, Scheduling, and Allocating Resources for Competitive Advantage* (New York: Marcel Dekker, Inc.), Chapter 2.

HERROELEN, W. and LEUS, R., 2001, On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, **19**, 559–577.

HERROELEN, W. and LEUS, R., 2003a, Project scheduling under uncertainty – survey and research potentials. *European Journal of Operational Research*, to appear.

HERROELEN, W. and LEUS, R., 2003b, The construction of stable project baseline schedules, *European Journal of Operational Research*, to appear.

HERROELEN, W., LEUS, R. and DEMEULEMEESTER, E., 2002, Critical chain project scheduling: do not oversimplify. *Project Management Journal*, **33**(4), 48–60.

IGELMUND, G. and RADERMACHER, F. J., 1983a, Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, **13**, 1–28.

IGELMUND, G. and RADERMACHER, F. J., 1983b, Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, **13**, 29–48.

LEUS, R., 2003, Project planning under uncertainty. Ph.D. dissertation, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

LEUS, R. and HERROELEN, W., 2003, Stability and resource allocation in project planning. IIE Transactions, to appear.

MAUGIÈRE, P., BILLAUT, J.-C. and ARTIGUES, C., 2002, Grouping jobs on a single machine with heads and tails to represent a family of dominant schedules. Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia 3–5 April, 265–269.

MEHTA, S. V. and UZSOY, R. M., 1998, Predictive scheduling of a job shop subject to break-downs. *IEEE Transactions on Robotics and Automation*, **14**, 365–378.

MÖHRING, R. H., RADERMACHER, F. J. and WEISS, G., 1984, Stochastic scheduling problems I – set strategies. *ZOR – Zeitschrift für Operations Research*, **28**, 193–260.

MÖHRING, R. H., RADERMACHER, F. J. and WEISS, G., 1985, Stochastic scheduling problems II – general strategies. *ZOR – Zeitschrift für Operations Research*, **29**, 65–104.

NEWBOLD, R. C., 1998, *Project Management in the Fast Lane – Applying the Theory of Constraints* (Cambridge: The St Lucie Press).

PATTERSON, J. H., 1984, A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. *Management Science*, **30**, 157–178.

PENZ, B., RAPINE, C. and TRYSTRAM, D., 2001, Sensitivity analysis of scheduling algorithms. *European Journal of Operational Research*, **134**, 606–615.

PET-EDWARDS, J., 1996, A simulation and genetic algorithm approach to stochastic resource-constrained project scheduling. Southcon Conference Record 1996, IEEE, Pascataway, NJ, 333–338.

PET-EDWARDS, J., SELIM, B., ARMACOST, R. L. and FERNANDEZ, A., 1998, Minimizing risk in stochastic resource-constrained project scheduling. Paper presented at the INFORMS Fall Meeting, Seattle, 25–28 October.

RAZ, T. and BARNES, R., 2001, A critical look at critical chain project management. Paper available at http://www.robertb.co.nz/CLCC.html.

SADEH, N., OTSUKA, S. and SCHELBACK, R., 1993, Predictive and reactive scheduling with the MicroBoss production scheduling and control system. Proceedings of the IJCAI-93 workshop on knowledge-based production planning, scheduling and control, 293–306.

Sevaux, M. and Sörensen, K., 2002, A genetic algorithm for robust schedules in a just in time environment. Research report LAMIH/SP-2003-1, University of Valenciennes, France.

Smith, S. S., 1994, Reactive scheduling systems. In Brown, D.E. and Scherer, W.T. (eds.), *Intelligent Scheduling Systems* (Boston: Kluwer Academic Publishers), 155–192.

Sotskov, Y. N., 1991, Stability of an optimal schedule. *European Journal of Operational Research*, **55**, 91–102.

Sotskov, Y., Sotskova, N. Y. and Werner, F., 1997, Stability of an optimal schedule in a job shop. *Omega*, **25**(4), 397–414.

Stork, F., 2001, Stochastic resource-constrained project scheduling. Ph.D. thesis, Technische Universität, Berlin.

Tavares, L. V., Ferreira, J. A. A. and Coelho, J. S., 1998, On the optimal management of project risk. *European Journal of Operational Research*, **107**, 451–469.

Tsai, Y. W. and Gemmil, M. D., 1996, Using a simulated annealing algorithm to schedule activities of resource-constrained projects. Working paper 96-124, Iowa State University.

Tsai, Y. W. and Gemmil, M. D., 1998, Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research*, **111**, 129–141.

Vieira, G. E., Herrmann, J. W. and Lin, E., 2003, Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling*, **6**, 39–62.

Wallace, S. W., 2000, Decision making under uncertainty: is sensitivity analysis of any use? *Operations Research*, **48**(1), 20–25.

Ward, S. and Chapman, C., 2003, Transforming project risk management into project uncertainty management. *International Journal of Project Management*, **21**, 97–105.

Wiest, J. D., 1964, Some properties of schedules for large projects with limited resources. *Operations Research*, **12**(May–June), 395–418.

Wiest, J. D. and Levy, F. K., 1977, *A Management Guide to PERT/CPM: with GERT/PDM/CPM and Other Networks* (Englewood Cliffs, NJ: Prentice-Hall).

Wu, S. D., Storer, H. S. and Chang, P.-C., 1993, One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research*, **20**(1), 1–14.

Yang, K.-K., 1996, Effects of erroneous estimation of activity durations on scheduling and dispatching a single project. *Decision Sciences*, **27**(2), 255–290.