Critical Chain Project Scheduling:
Do Not Oversimplify

Willy Herroelen • Roel Leus* **• Erik Demeulemeester**

April 2001

Department of Applied Economics
Katholieke Universiteit Leuven
Naamsestraat 69, B-3000 Leuven (Belgium)
Tel +32-16-32 69 70 - +32-16-32 69 67 - +32-16-32 69 72
Fax +32-16-32 67 32
e-mail: <first name>.<name>@econ.kuleuven.ac.be

*Research Assistant of the Fund for Scientific Research – Flanders (Belgium) (F.W.O.)

Critical Chain Project Scheduling: Do Not Oversimplify

Abstract

Critical Chain Scheduling/Buffer Management (*CC*/*BM*) – the direct application of the Theory of Constraints (*TOC*) to project management – has received a lot of attention in the project management literature. There still is a lot of controversy over the merits and pitfalls of the Critical Chain Scheduling/Buffer Management methodology (*CC*/*BM*). The objective of this paper is to focus on the fundamental elements of the *CC*/*BM* scheduling logic and pinpoint some intricacies that are not commonly referred to in the available literature. Our analysis is based on a critical review of the relevant sources and experimentation with both commercial *CC*/*BM* software and an in-house developed *CC*/*BM* based scheduling tool.

Keywords: critical chain; buffer management; resource-constrained project scheduling

Introduction

Critical Chain Scheduling/Buffer Management (*CC/BM*) – the direct application of the Theory of Constraints (*TOC*) to project management – has received a lot of attention in the project management literature. Subsequent to the publication of Goldratt's book *Critical Chain* in 1997 (Goldratt, 1997), numerous books (Newbold 1998, Leach 2000) and articles (Cabanis-Brewin, 1999; Globerson, 2000; Herroelen and Leus, 2000, 2001; Leach, 1999; Maylor 2000; Patrick, 1999; Pinto, 1999; Rand, 2000; Umble and Umble, 2000) have been written in order to clarify the *CC/BM* scheduling philosophy. Internet discussion groups have been set up to discuss critical chain scheduling issues (see for example http://www.prochain.com/discussion.html and http://groups.yahoo.com/group/CriticalChain). While real-world applications by companies such as Lucent Technologies and Harris Semiconductor have been described in order to demonstrate the effectiveness of the *CC/BM* approach (Leach, 1999; Umble and Umble, 2000), other sources (Zalmenson and Zinman, 2000, 2001) alert the reader to serious drawbacks and implementation failures or at least claim *CC/BM* is not at all innovative (Wilkens, 2000). There still is a lot of controversy over the merits and pitfalls of the Critical Chain Scheduling/Buffer Management methodology (*CC/BM*). Being confronted with many views and comments on *CC/BM* written by either true believers or committed naysayers who fail to offer a fair appraisal of the *CC/BM* project scheduling technique, *PM Network*, the Professional Magazine of the Project Management Institute, decided to put an Editor's Note in its January 2001 issue, stating that "*PM Network* is neutral on Theory of Constraints; the Publishing Division holds no allegiance either for or against it". The objective of this paper is not to defend or reject *CC/BM*. *CC/BM* has much to offer if applied wisely and if the practical implications and limitations are well understood. Our analysis is based on a critical review of the relevant literature and experimentation with both commercial *CC/BM* software and an in-house developed *CC/BM* based scheduling tool (Herroelen and Leus, 2000). The paper is organized as follows. A first section provides a short overview of the *CC/BM* methodology. We deliberately concentrate on the basic ideas and refer the reader to the above mentioned sources for further detail. In the next section, we focus on the fundamental elements of the *CC/BM* scheduling logic and pinpoint some intricacies that are not commonly referred to in the available literature. The last section provides overall conclusions.

The *CC/BM* methodology

The fundamentals of *CC/BM* are summarized in Table 1. *CC/BM* builds a *baseline schedule* using activity duration estimates based on a 50% confidence level. Activity due dates and project milestones are eliminated and multi-tasking is to be avoided. In order to minimize 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 that chain of precedence and resource dependent activities which determines the overall duration of a project. If there is more than one critical chain, just select one. The safety associated with the critical chain activities that was cut away by selecting aggressive duration estimates 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* (FB) 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 (Newbold, 1998; Product Development Institute, 1999), 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* (RB), 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.

Table 1

During project execution both the critical chain and the baseline schedule should be kept fixed. Project activities are executed according to the roadrunner mentality using an unbuffered *projected schedule*. This schedule is early start based, except for the gating tasks (activities without predecessors), which are to be started at their baseline scheduled start times. Early completion of activities is to be reported and activities should be started as soon as they can when work becomes available. The execution of the project is managed by using the buffers as a proactive warning mechanism. As activities are completed, project management should keep track of buffer consumption. As long as a predetermined portion of the buffer remains (buffer consumption is only in the green (*OK*) zone), everything is assumed to go well. If buffer consumption moves beyond a certain point, a warning is raised (the yellow *Watch and Plan* zone). If it deteriorates past a critical point (the red *Act* zone), corrective action is to be taken.

In a multi-project environment, *CC*/*BM* relies on the five common steps of *TOC* summarized in Table 2: (a) prioritize the organization's projects, (b) plan the individual projects according to the *CC*/*BM* fundamentals, (c) stagger the projects, (d) insert drum buffers, (e) measure and report the buffers, and (f) manage the buffers.

Table 2

Step (a) aims at avoiding multi-tasking among projects. Step (b) calls for the identification of the most constraining company resource as the "bottleneck" (strategic or drum resource). Projects are staggered (step (c)) 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 (d)) is placed before activities on the strategic resource in order to protect the strategic resource from disruptions on nonstrategic resources. The buffers can then be managed more or less the same way as done in the single-project case (streps (e) and (f)).

The *CC/BM* scheduling approach

In this section we draw attention to some essential but not entirely straightforward features of the *CC/BM* ***scheduling*** logic. For discussions on a more managerial level, we refer to Pinto (1999), Elton and Roe (1998) and McKay and Morton (1998). We focus on the procedures for identifying the critical chain in the baseline schedule, buffering the baseline schedule, using the buffers as a proactive mechanism, and the issue of stability and rescheduling. Throughout our discussion we will refer to major findings of a factorial experiment conducted by Herroelen and Leus (2001) on the well-known Patterson test problem set (Patterson, 1984) which contains test instances with a number of activities ranging from 7 to 50 requiring up to three renewable resource types each. The authors confronted the results obtained using an exact solution procedure for the resource-constrained project scheduling problem (Demeulemeester and Herroelen, 1992, 1997) with the results generated using their own computerized *CC/BM* scheduler and tested the impact of the project scheduling mechanism, the buffer size computation method, the composition of the critical chain, and the recomputation mechanism of the baseline schedule.

Identifying the critical chain

One of the fundamental merits of *CC/BM* is that it explicitly recognizes that it is the interaction between activity durations, precedence relations, resource requirements and resource availabilities which determines the project duration. It is precisely this interaction which results in one or more sequences of activities, consisting of precedence related or resource dependent segments, that determines the length of the baseline schedule. Goldratt (1997) identifies such a sequence as the *critical chain*. Although this concept is not new – Wiest (1964) introduced the concept of a *critical sequence* to describe such a sequence of activities already more than thirty years ago, and it has been by used by many researchers in project scheduling to compute a lower bound on the project duration (Herroelen et al., 1998) – *CC/BM* has revealed its importance to the practitioner and as such plays a crucial role in the awakening process to the disappearance of the traditional critical path concept in the presence of resource constraints. Moreover, *CC/BM*'s argument that time protection against uncertainty should not be allocated to the individual activities but should rather be aggregated and concentrated at particular points in the baseline schedule is well-taken and commonly accepted in *TOC* based scheduling. In short, the idea of generating a deterministic baseline schedule and protecting it against uncertainty is sound and appeals to management. *CC/BM* paved the way for the recent introduction of protective buffering mechanisms into commercial project planning software (examples are *ProChain* as an add-in to *Microsoft Project* and the critical chain planning functions of Scitor's *PS*8 package).

It should be noted, however, that a baseline schedule may contain more than one critical chain, and the composition of the critical chain(s) is entirely dependent on the procedure used for generating the baseline schedule. Creating a precedence and resource feasible baseline schedule that minimizes the project duration – the number one objective used by *CC/BM*'s baseline scheduling logic – is not easy (most resource-constrained project scheduling problems are *NP*-hard). Goldratt (1997, p. 217) seems to minimize

the issue and claims that “in *each case* the impact on the lead time of the project is less than even half the project buffer” and suggests to cut for each step a piece of paper so that “the length represents time. This way we can move them around until there is no contention”. It should also be mentioned that there is a lack of consensus in the *CC/BM* literature about the activity duration estimates to be used for generating the initial baseline schedule. Goldratt (1997) suggests to use the median while the Product Development Institute (1999) argues in favour of the mean. Herroelen and Leus (2001) conclude from their full factorial experiment that the use of the *mean* activity duration provides the safest estimates of the project duration.

In order to illustrate that the length of the baseline schedule and the collection of possible critical chains to choose from is entirely dependent on the scheduling procedure used, consider the example network given in Figure 1 (Demeulemeester et al., 1994). The numbers above each node denote the activity duration of the corresponding activity (node 1 and node 9 are dummy nodes with zero duration). It is assumed here that these activity duration estimates do not include individual safety provisions. The numbers below each node of the network in Figure 1 represent the number of units of a renewable resource (e.g. workers) required during every period of the corresponding activity’s duration interval. It is assumed that the renewable resource has a constant availability of 5 units per period.

Figure 1

Figure 2 gives a minimum duration schedule (this can be obtained, for example using the branch-and-bound procedure of Demeulemeester and Herroelen (1992, 1997)). As can easily be verified, the project duration is 7 time periods (by poor luck this corresponds to the length of the longest (critical) path 1-2-6-7-9). The baseline schedule reveals three critical chains: the chain 2-6-7, the chain 2-6-3-8, and the chain 5-4.

Figure 2

Figure 3 shows the baseline schedule generated by the commercial software *Microsoft Project*. This baseline schedule has a length of nine time periods and reveals two possible critical chains, different from the ones present in the optimal duration schedule of Figure 2: the chain 4-7 and the chain 5-6-7.

Figure 3

If the project planner would rely on the *ProChain* software, the schedule of Figure 4 would be obtained. The schedule now has a length of eight time periods and has two critical chains: the chain 3-5-8 and the chain 3-6-7. Both activity 2 and activity 4 have been right-shifted, i.e. scheduled as late as possible within the scheduling horizon determined by the critical chains. The software selects the chain 3-6-7 as the critical chain.

Figure 4

If the user relies on *PS*8 for generating the baseline schedule, the optimal seven-period schedule of Figure 2 results, with the critical chains 2-6-7, the chain 2-6-8-3, and the chain 5-4. The software selects the critical chain 2-6-7.

In short, four different baseline schedules have been obtained for the same project, ranging in duration from 7 to 9 periods and each providing the planner with a different population of critical chains to choose from. Goldratt (1997) contends that it does not really matter which critical chain is chosen. The above simple example provides evidence of the fact that generating a good baseline schedule does matter, and that statements such as "the critical chain is never ambiguous" (Leach, 2001b) must be interpreted with sufficient care. The length of the baseline schedule, the set of candidate critical chains and, as a result, the activities in the chosen critical chain, depend on the procedure used for generating the baseline schedule.

Adding a 50% integer length project buffer according to *CC*/*BM* logic to the baseline schedules of Figures 2, 3, 4 and 5, would inspire management to make the customer promise to deliver the project in at least 11, 14, 12 or 11 periods, respectively (these should be interpreted as "lower bounds"; it is to be expected (and confirmed below) that the insertion of feeding buffers may well lead to larger committed project durations). If one fails to use the minimum duration baseline schedule as the starting point for buffering and negotiating the project due date with the customer, one may well lose the contract!

Buffer insertion and management

CC/*BM* argues that safety time should best be eliminated from individual activity duration estimates and aggregated in the form of buffers at crucial locations in the baseline schedule. The buffers should offer protection against statistical variation and should act as safeguards that provide a proactive protection mechanism. It should be observed that the experiment of Herroelen and Leus (2000) revealed that the 50% rule for sizing the buffers may lead to a serious overestimation of the required project buffer size, and consequently, of the project duration committed to the customer.

By buffering the baseline schedule so that the completion time of the last positioned critical chain activity is scheduled to occur much later than the completion of any other non-critical chain of activities, *CC/BM* avoids the cumbersome task of performing generalized PERT calculations (Elmaghraby, 1977) to make reliable predictions about project completion in the presence of resource constraints. The feeding buffers decouple the critical chain from the remaining activities in the network and as such they help management in keeping focus and setting realistic due dates. As a result, there is a subtle difference between the interpretation of a feeding buffer and the CPM notion of activity float. On the other hand, the management of the buffers during project execution does not notably differ from float management *in the absence* of resource constraints. As was already noted by Wiest (1964), however, we must recognize the

conditional nature of slack when resources are limited: slack values are associated with a particular schedule. Moreover, the (re-)scheduling policy adopted will also influence activity slack (we refer to Bowers (1995), who works in an environment where resource allocation remains unchanged).

When the feeding buffers are inserted by pushing back their feeding chains, the critical chain may no longer be the longest chain in the network. This has serious implications for the projected schedule, which is unbuffered and in which every gating task (task without real predecessors) is set to start at its scheduled time in the baseline schedule, and where the roadrunner mentality dictates the start times of the other activities. It may well be that the critical chain, that is the most important chain of activities that is supposed to determine the duration of the project, is started later than non-critical chain activities, a rather counter-intuitive way of work. Also, gaps may be created in the critical chain of the buffered baseline schedule. *CC/BM* offers a pragmatic response to these problems in that the critical chain is simply defined as the longest chain before the buffers are inserted (Product Development Institute, Chapter 5, 1999). Finally, as will be discussed later, inserting the feeding buffers may lead to immediate resource conflicts which must be resolved with sufficient care in order to avoid an unnecessary increase of the length of the baseline schedule.

While the buffer protection idea is certainly defendable and widespread within scheduling practice, the practical implications should be well understood. First of all, project management should be sufficiently alert for the idiosyncrasies of the project planning software package in use and should interpret the buffered schedules generated by commercial software packages with extreme care. Figure 5 shows the buffered baseline schedule generated by a commercial software package with integrated *CC/BM* scheduling logic for the project of Figure 1. The planned project duration is 13 periods. The critical chain activities 2-6-7 are kept in series but the chain exhibits a one-period gap.

Figure 5

Figure 6 shows the buffered baseline schedule obtained for the same project with an add-in commercial software package. The planned project duration is also 13 periods, but now management is invited to focus on a different critical chain 3-6-7 which is no longer a chain. The critical chain activities are not kept in series. Critical chain activity 3 is shifted forward in time and is scheduled in parallel with critical chain activity 7.

Figure 6

Feeding buffers should be inserted with sufficient care, otherwise an unwanted serious increase in the length of the baseline schedule may well be the result. Rather than using the *CC/BM* textbook approach to simply shift back and forth activities till resource contention is resolved, we recommend to treat the buffers as artificial activities and reschedule the network. Also, we want to alert the reader to the fact that

it may well be the case that the buffer protection mechanism fails to be proactive. In order to illustrate the first point, consider the project shown in Figure 7. The numbers above each node again denote the activity duration, the numbers below each node represent the renewable resource requirement per period for a single resource type with a constant availability of 3 units per period. As such, activity 5 has a duration of 5 periods and requires 2 units of the renewable resource type during each of its execution periods.

Figure 7

Figure 8(a) shows a minimum duration schedule for the project. The critical chain is identified as chain 1-2-3-8 with a duration of 17 periods. As indicated in Figure 8(b), three feeding buffers must be inserted: the buffer FB5 should be inserted after activity 5, the buffer FB6 should be inserted after activity 6 and the buffer FB7 should be placed after activity 7. If we apply the 50% rule for buffer sizing and if we assume integer length buffers, FB5 should have a length of 4 periods, FB6 should have a length of 3 periods, while FB7 should be 2 periods in length. If the buffers are inserted by pushing the chain of activities feeding the feeding buffer backwards in time, *immediate* resource conflicts may occur. If we start by pushing backwards activity 5 an immediate resource conflict is created with activities 6 and 7. Shifting activity 7 backwards in time does not create sufficient room for activity 5. Activity 4 is a predecessor of both activities 5 and 6. Shifting the chain of activities 4-6 backwards in time does not release sufficient resources for the placement of activity 5. Activity 5 may be allowed to jump over activity 6, but cannot jump over its predecessor activity 4. Shifting backwards the chain of activities 4-5 which feeds the buffer FB5 does not help either. We are forced by the resource constraint to increase the project duration by almost 50%, as shown in Figure 8(b). If, however, we treat the buffers as dummy activities with a positive duration and compute again a minimal duration schedule, keeping the critical chain unchanged, the schedule of Figure 8(c) results. It has exactly the same length as the original baseline schedule of 8(a), and only differs in the planning of the non-critical chain activities. Sufficient room is now available for the proper insertion of the 3 feeding buffers.

Figure 8

In order to illustrate that the penetration of feeding buffers may *instantaneously* lead to resource conflicts which prohibit the feeding buffers from acting as a proactive warning mechanism, consider the example network shown in Figure 9(a). The numbers above each node again denote the corresponding activity durations. Six resource types are used in this project. Resource types *A*, *B*, *C*, *D*, and *F* have a constant availability of one unit per time period, while resource type *E* has a constant availability of two units per time period. The activity resource requirements are indicated below each network node.

Figure 9

The buffered baseline schedule is shown in Figure 9(b). The critical chain consists of the activities 1-4-8. Assume now that during project execution, activity 2 is the subject of a *very small* delay. This delay will cause a penetration of feeding buffer FB3-4. Given the small delay, this penetration will not exceed the first third of the buffer, so that no management action is called for. However, the delay will also lead to an immediate resource conflict with activity 6, which requires the same resource *B*. Delaying activity 6 will not only lead to a penetration of feeding buffer FB7-8 (again, this penetration will be too small to call for management action), but will also lead to a resource conflict with critical chain activity 4 which requires the same resource *B*. The feeding buffers do not protect the critical chain from the merge bias. The result is an *immediate delay in the critical chain*. This will be the case even if a resource buffer (wake-up call) for resource *B* would be present in front of activity 4. Clearly, the feeding buffers FB3-4 and FB7-8 fail to act as a warning mechanism in anticipation of future difficulties. The feeding buffers used by *CC/BM* are *time* buffers. It is important to note that *time* buffers may fail to cushion *resource* conflicts. Such immediate resource conflicts must be resolved, possibly requiring actions to repair the schedule. In the simple example described here, the required schedule repair actions are rather trivial. This will no longer be the case, however, in real-life project network structures where multiple activities and multiple resource types may be involved.

A final point to be made about buffer sizes is that they may very well be updated as the project progresses. One of the purposes of the project buffer is that it provides an estimate of the completion time of the project that can be realized with an appropriate probability and, hence, an estimate of the project due date. As shown in the experiments conducted by Herroelen & Leus (2001), it may be beneficial to update the size of the project buffer during project execution, as a reaction to the dynamic process of re-evaluating the chances of meeting the due date. A related idea would be to shift the alert tresholds of buffer consumption backward in time, assessing the consumption of buffers relative to project or critical chain completion. Patrick discusses various techniques more in depth in messages of the Yahoo discussion group. His preference, as does ours, goes to the recalculation of the project buffer size: “One advantage of this approach is that rather than depend on some arbitrary percentage of the buffer (...), the treshold is determined not at the launch of the project but in a “just-in-time” manner that allows the process to take into consideration new information encountered during the execution of the project.”

Rescheduling and stability

Consider again the buffered schedule of Figure 8(c). If activity 7 were to take 0 time, or activity 2 were to take 3 time units more than anticipated, activity 6 could be repositioned before activity 5. More generally, it makes sense at every project status update, to check for opportunities to speed up the projected schedule by rearranging jobs. This is usually done implicitly by available software packages, but it is interesting to explicitly recognize the option.

Figure 10

Figure 11

We invite the reader to have a look at the example project in Figure 10(a). We assume that all the activities have the same unit duration. Activities *F* and *H* cannot be scheduled in parallel because they compete for the same resource with unit availability; the same holds for *D* and *E*. Resource usage of the other activities is not restrictive. One possible critical chain would be *B-E-H-F-I*. The corresponding buffered baseline schedule is shown in Figure 10(b). The unbuffered projected schedule is depicted in Figure 10(c). Suppose now that at the foreseen end of activity *D*, this activity is perceived to take longer than expected. Keeping the critical chain unchanged would yield the new projected schedule shown in Figure 11(a). Clearly, the makespan of the projected schedule has gone up. Assume now that we no longer require the critical chain to remain in series. In this case, this means that it is no longer necessary to schedule activity *H* in front of activity *F*. The result is the projected schedule shown in Figure 11(b), which has a smaller makespan. However, the new projected schedule is actually an implicit recognition of the fact that we are dealing with a new critical chain! It would be logical to identify this chain and insert/resize the buffers accordingly. This also allows to fully exploit the updated project buffer size in making reliable project completion time estimates, as mentioned earlier. The foregoing observations were also made as a result of the simulation experiments in Herroelen and Leus (2001): if the critical chain is required to be kept in series during project execution, the makespan can be improved by regularly re-evaluating the "tightness" of the critical chain. This is logical: one would prefer to avoid concentrating on an ex ante derived critical chain, that is no longer the real constraint on project makespan.

CC/BM recommends that the baseline schedule and the selected critical chain should not change during project execution (except for very fundamental disruptions which consume the protection offered by the project buffer). It is argued that rescheduling and changing the critical chain leads the project team into losing focus. Leach (1999, 2001a, 2001b) illustrates the point through an analogy with the funnel experiment conducted by Deming (1982) to illustrate the differences between common cause and special cause variation. Common cause variation refers to a cause that is inherent in the system. Special cause variation refers to a cause that is specific to some group of workers, to a particular production worker, to a specific machine, or to a specific local condition (Leach, 1999). Special cause variation requires management action, while common cause variation does not. The experimental set-up used in the funnel experiment consists of a target on the floor, a funnel on a movable stand, and a marble to drop from the funnel. The aim of the system is to cause the marble to land on the center of the target. In the context of project scheduling, the target can be thought of as the committed project completion date and cost. The funnel experiment shows that adjusting a stable process to compensate for an undesirable result or an extraordinarily good result will produce output that is worse than if the process had been left alone. Moving the funnel in the opposite direction whenever the marble misses the target (Leach (2001a) relates this to expediting a newly identified critical path), or moving the funnel back to zero each time before making the adjustment (Leach (2001a) relates this to updating the project baseline schedule with a change

or adjusting the plan to actual before taking the management control action) leads to an increase in variation. Leach (2001a) claims that these actions boil down to an adjustment for statistical fluctuation (common cause variation) instead of real change (special cause variation).

There is, however, a subtle difference between the parameter setting of a univariate or bivariate statistical process, and the rather combinatorial nature of project scheduling, be it with variable activity durations: the analogy does not hold. In many companies it will indeed be the case that rescheduling is undesirable, but this will mostly be because of organisational choices for co-ordinating resources, rather than because of underlying statistical intuition, as is the case in the funnel experiment. Statistical process control is about observing the same statistical process multiple times (production environment), whereas every single activity is only executed once in a project, which has by itself a unique character. This observation makes any intimate comparison between the two settings faulty.

What is more, the closest equivalent obtainable in the funnel experiment to receiving new information about activity durations, would not be the case where one tampers with the settings of the funnel after one observation, but rather where one adjusts the location of the funnel because the variability of the output is reduced (the project moves towards completion) and the location of the funnel can unambiguously be better positioned. This is because project activities are precedence and resource dependent. During the execution of a project, the occurrence of disturbances changes the structure of that part of the project which remains to be executed: they provide additional information. This information should not just be treated as the result of statistical fluctuation, but should be interpreted as special cause variation which may require management action. Opportunities for speeding up the remaining part of the on-going project may be exploited by rearranging the schedule. Discarding *from the outset* information that will allow us to complete the remainder of the schedule in the fastest way given the currently available information, may not always be wise. The factorial experiment performed by Herroelen and Leus (2000) confirms that – whether we like it from a practical project management viewpoint or not – regularly recomputing the baseline schedule and updating the critical chain leads to a significantly smaller realized project duration. This reactive type of approach has also been discussed by Jørgensen (1999) in a more theoretical setting.

Admittedly, in a number of cases, rescheduling will not be desirable. This, however, is motivated by some authors that argue that workers will get to think management does not know what it is doing, if rescheduling of the baseline is applied on a routine basis. Note, however, that the schedule that indicates when workers are to start their next job, the projected schedule, is updated anyway every time project status updates occur: Herroelen and Leus (2001) clearly show that this mechanism is most similar to a dispatching approach. The fact that at times it may be a different critical chain that determines the projected schedule updates and job dispatching, should not upset workers much more than in the unchanging critical chain case, especially not if they see the project can be accelerated in this way! On the other hand, it is imperative that dispatched jobs are not normally interrupted because of shifting priorities; such decisions

will be counterproductive and introduce system nervousness: it might be very difficult to transfer expert staff and special resources between activities at short notice (Bowers, 1995).

Other sources of the need for schedule stability will often result from ex ante imposing (limits on) certain activity start and end times to co-ordinate resources, for instance across multiple projects. This is one reason why piloting a single project according to the *CC*/*BM* methodology in a multi-project organisation will demand extra care: it may be necessary in a multi-project environment to make advance bookings of key staff or equipment to guarantee their availability (Bowers, 1995). Other sources of the need for such stability can be hard delivery dates of suppliers or subcontractors, or in a larger sense, a hard due date for (intermediate or final) deliverables (e.g. milestones), in other words any time restriction that is external to the project itself. If delivery dates or "ready times" are tight and/or due dates are not, the plan may have to be created backward from those dates; remember that it is standard *CC*/*BM* procedure to introduce a separate project buffer per deliverable (ProChain, 1999). If the due dates are tight on the other hand (after exhausting all possible "activity crashing" alternatives), a baseline schedule loses its significance and implicit buffer sizes follow from comparing the projected schedule with the imposed dates, and there is no real distinction anymore between buffer management and "classical" float management.

Based on the foregoing, it is clear that the decision to repair the schedule or engage in rescheduling should be taken with extreme care. Simple, clear-cut answers do not seem to exist. This being said, it is worth noting nevertheless that the projected schedule will require regular adaptation anyway to account for resource contention. Also, proponents of unconditionally keeping the critical chain unchanged under all except very grave circumstances, should ask themselves whether they refuse to revise the baseline schedule because it introduces system nervousness, or rather because such schedule management in the *CC*/*BM* way becomes rather arduous. For, re-evaluating what part of the schedule has become most critical in such situations will regularly boil down to completely repositioning the buffers and the critical chain. For feeding buffers for instance, Wilkens (2000) states: "feeding buffers are another story altogether, simply because they are a pain in the neck to manage (...). As work is replanned to incorporate changes and workaround plans, the buffers need to be relocated and adjusted." Clearly, the need for intelligent scheduling/repair mechanisms remains and additional research is needed in the development of powerful mechanisms for the creation of robust baseline schedules and the deployment of effective proactive warning mechanisms.

Conclusions

The *CC*/*BM* scheduling methodology has acted as an important eye-opener in project management practice. It correctly recognizes that the interaction between the time requirements of the project activities, the precedence relations defined among them, the activity resource requirements and the resource availabilities has a crucial impact on the duration of a project. This point, largely unrecognized by many

project management practitioners and blind critical path fanatics, is well-taken and completely in line with the profound insights gained in the resource-constrained project scheduling literature. Given the complexity and the moderate research progress of stochastic resource-constrained project scheduling, the idea of constructing and protecting a mean activity duration based precedence and resource feasible deterministic baseline schedule is sound. It is often the best thing one can hope for and rely on from a computational point of view. The methodology of inserting and managing feeding, resource and project buffers – essentially the application of the Theory of Constraints to project scheduling – provides a *simple* and *workable* tool for setting realistic project due dates and for monitoring the project during execution. It emerged into project management software which allows project management to shift from a time-oriented (critical path based) to a resource-constrained (critical chain based) view. Daily project management practice was in a desperate need for such a shift in focus. The danger, however, lies in the oversimplification. McKay and Morton (1998) express this as follows: "We believe that most project managers will benefit from a conscious reflection of what Goldratt has written (...). However, a danger exists for the project manager who does not understand the necessary preconditions...". Indeed, the practical implications of applying *CC*/*BM* need to be clearly recognized. Pinto (1999) concludes: "my concern lies in the strong potential for many companies, who do not understand both TOC's strengths *and* weaknesses, to latch on to it as TNBT (The Next Big Thing, meaning a management fad) regardless of their particular and unique circumstances".

Resource-constrained project scheduling problems are hard: constructing and protecting a baseline schedule is not an easy pursuit. *CC*/*BM* offers a practical, software supported project scheduling tool. The merits and pitfalls of the *CC*/*BM* scheduling mechanism have been discussed in this text. It turns out that identifying the critical chain is not an easy matter. Due to the complexity of the resource-constrained project scheduling problem involved, management is forced to rely on heuristic procedures for generating a precedence and resource feasible baseline schedule for real-life projects. Different scheduling methods may yield different project durations and/or a different set of critical chains. The procedures used for inserting the buffers may have a strong impact on the project completion times committed to the customer, and commercial software packages may provide ambiguous results. We have shown that during project execution, the buffers may fail to act as a real proactive protection mechanism and pointed out that management must face the trade-offs involved in schedule repair and rescheduling. Simple, clear-cut answers do not seem to exist. Further research is needed in the development of workable and effective mechanisms for creating robust baseline schedules and effective mechanisms for schedule repair.

References

Bowers, J.A. (1995). Criticality in resource constrained networks. *Journal of the Operational Research Society*, 46, 80-91.

Cabanis-Brewin, J. (1999). "So … So What??" Debate over CCPM gets a verbal shrug from TOC guru Goldratt. *PM Network*, 13 (December), 49-52.

Demeulemeester, E.L. and Herroelen W.S. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12), 1803-1818.

Demeulemeester, E.L. and Herroelen W.S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11), 1485-1492.

Demeulemeester, E., Herroelen, W.S., Simpson, W., W.P., Baroum, S., Patterson, J.H. and Yang, K.-K. (1994). On a paper by Christofides et al. for solving the multiple-resource constrained, single project scheduling problem, *European Journal of Operational Research*, 76, 218-228.

Deming, W.E. (1982). *Out of the crisis*. MIT, Center for Advanced Educational Services, Cambridge, Massachusetts.

Elmaghraby, S.E.E. (1977). *Activity networks – Project planning and control by network models*. Wiley Interscience.

Elton, J. and Roe, J. (1998). Bringing discipline to project management, *Harvard Business Review*, 76 (March-April), 153.

Globerson, S. (2000). PMBOK and the critical chain. *PM Network*, 14(5), 63-66.

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

Herroelen, W., De Reyck, B. and Demeulemeester, E. (1998). Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research*, 25(4), 279-302.

Herroelen, W. and Leus, R. (2000). On the merits and pitfalls of critical chain scheduling. Proceedings of the PMI Research Conference 2000, June 21-24, Paris, France, 283-295.

Herroelen, W. and Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, to appear.

Jørgensen, T. (1999). *Project Scheduling – A Stochastic Dynamic Decision Problem*. Doctoral dissertation, Norwegian University of Science and Technology, Trondheim, Norway.

Leach, L.P. (1999). Critical chain project management improves project performance. *Project Management Journal*, June, 39-51.

Leach, L.P. (2000). *Critical chain project management*. Artech House Professional Development Library.

Leach, P. (2001a). Putting quality in project risk management – Part 1: Understanding variation. *PM Network*, 15(2), 53-56.

Leach, P. (2001b). Putting quality in project risk management – Part 2: Dealing with variation. *PM Network*, 15(3), 47-52.

Maylor, H. (2000). Another silver bullet? A review of the TOC approach to project management, Paper presented at the 7th International Annual EurOMA Conference, Ghent, June 4-7.

McKay, K.N. and Morton, T.E. (1998), Critical chain, *IIE Transactions*, 30, 759-762.

Newbold, R.C. (1998). *Project management in the fast lane – Applying the Theory of Constraints*. The St. Lucie Press, Boca Raton.

Patrick, F.S. (1999). Critical chain scheduling and buffer management – Getting out from between Parkinson's rock and Murphy's hard place. *PM Network*, 13 (April), 57-62.

Patterson, J.H. (1984). A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. *Management Science*, 30, 854-867.

Pinto, J.K. (1999). Some constraints on the Theory of Constraints – Taking a critical look at the critical chain. *PM Network*, 13 (August), 49-51.

ProChain Solutions Inc. (1999). *Introduction to ProChain*.

Product Development Institute (1999). Tutorial: Goldratt's Critical Chain Method. A one project solution, http://www.pdinstitute.com.

Rand, G.K. (2000). Critical Chain: the Theory of Constraints applied to project management. *International Journal of Project Management*, 18(3), 173-177.

Umble, M. and Umble, E. (2000). Manage your projects for success: an application of the Theory of Constraints. *Production and Inventory Management Journal*, Second Quarter, 27-32.

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

Wilkens, T.T. (2000). Critical path, or chain or both? *PM Network*, 14(7),68-74.

Zalmenson, E. and Zinman, E. (2000). People oriented project management. http://www.cybermanage.net.

Zalmenson, E. and Zinman, E. (2001). TOC analysis. http://www.cybermanage.net.

TABLE CAPTIONS

FIGURE CAPTIONS

***CC/BM* fundamentals**
50% probability 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 1

***CC/BM* multi-project scheduling**
Prioritize the projects
Plan the individual projects according to *CC/BM* fundamentals
Stagger the projects by inserting capacity buffers
Insert drum buffers
Measure and report the buffers
Manage the buffers

Table 2

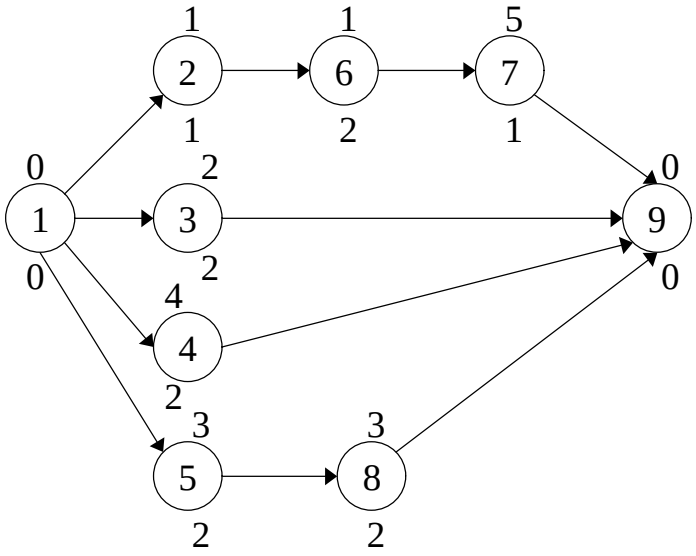


Figure 1

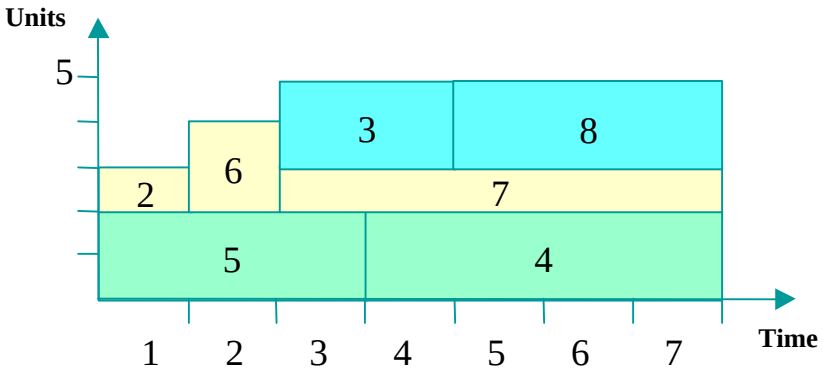


Figure 2

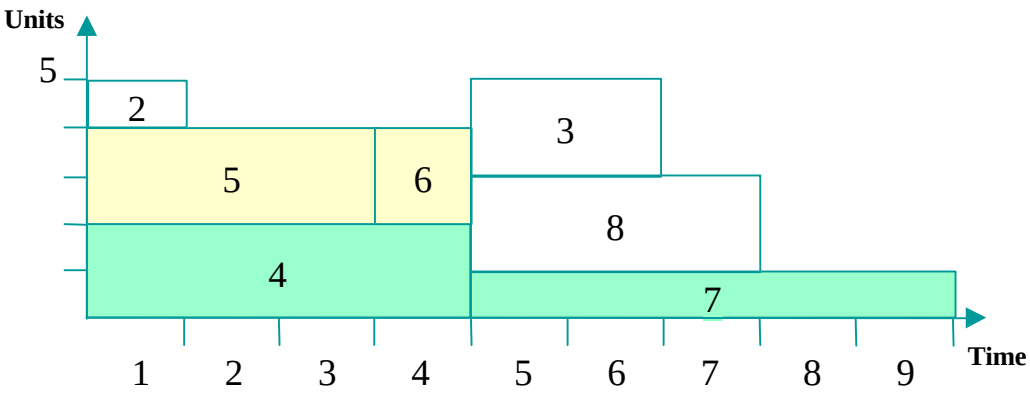


Figure 3

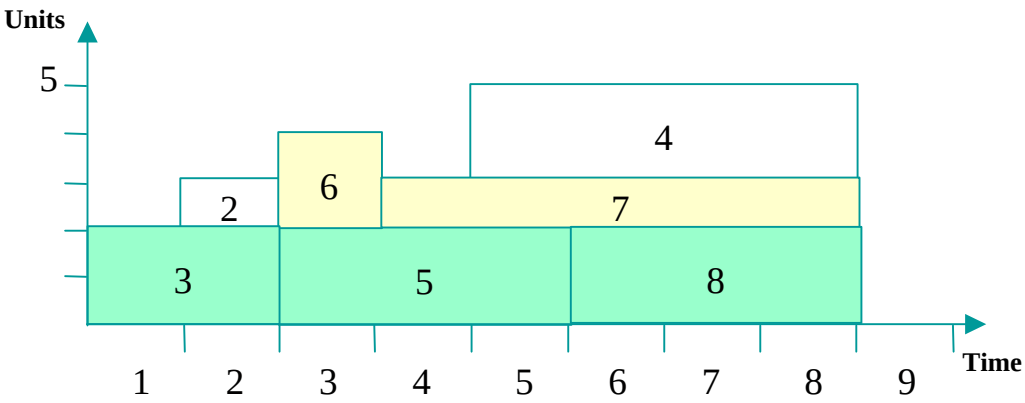


Figure 4

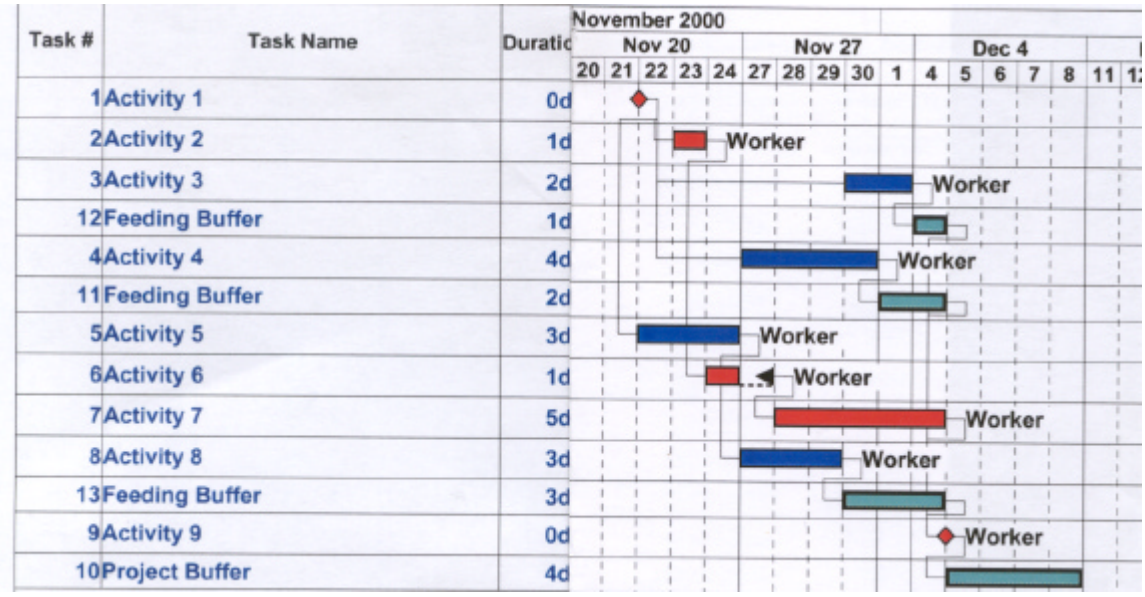

Task #
Task Name
Duratic
November 2000
Nov 20
Nov 27
Dec 4
20 21 22 23 24 27 28 29 30 1 4 5 6 7 8 11
1 Activity 1 0d
2 Activity 2 1d
3 Activity 3 2d
12 Feeding Buffer 1d
4 Activity 4 4d
11 Feeding Buffer 2d
5 Activity 5 3d
6 Activity 6 1d
7 Activity 7 5d
8 Activity 8 3d
13 Feeding Buffer 3d
9 Activity 9 0d
10 Project Buffer 4d
Worker

Figure 5

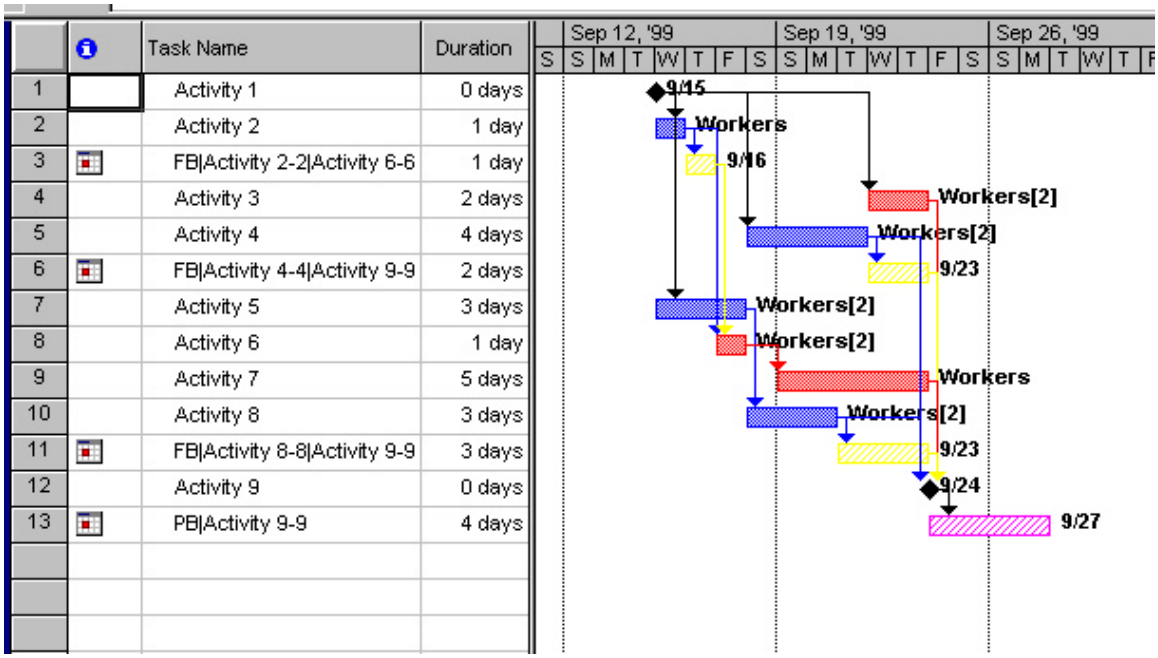

Task Name
Duration
Activity 1
0 days
Activity 2
1 day
FB|Activity 2-2|Activity 6-6
1 day
Activity 3
2 days
Activity 4
4 days
FB|Activity 4-4|Activity 9-9
2 days
Activity 5
3 days
Activity 6
1 day
Activity 7
5 days
Activity 8
3 days
FB|Activity 8-8|Activity 9-9
3 days
Activity 9
0 days
PB|Activity 9-9
4 days
Sep 12, '99
Sep 19, '99
Sep 26, '99
9/15
Workers
9/16
Workers[2]
Workers[2]
9/23
Workers[2]
Workers[2]
Workers
Workers[2]
9/23
9/24
9/27

Figure 6

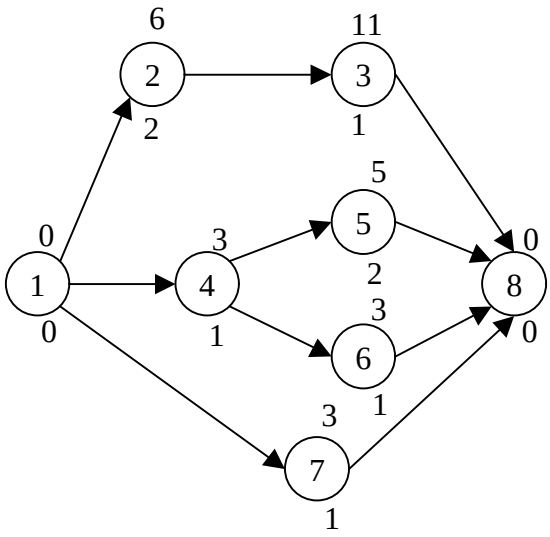
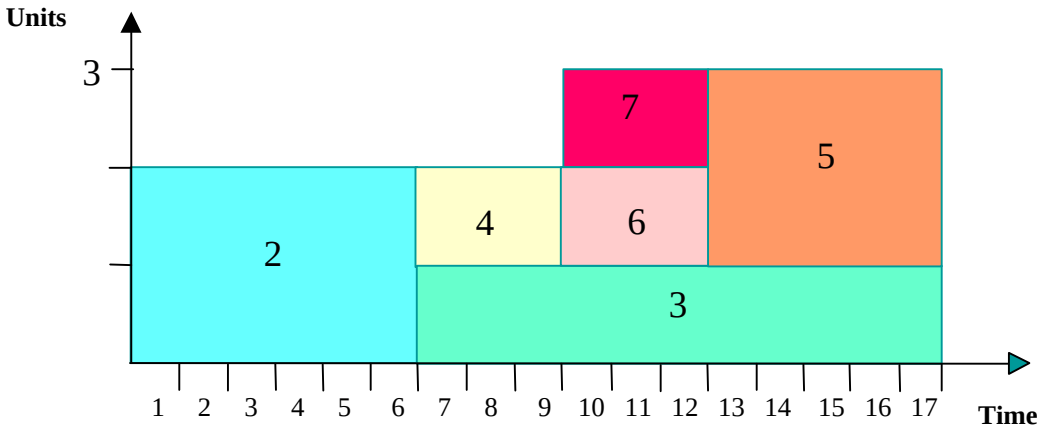
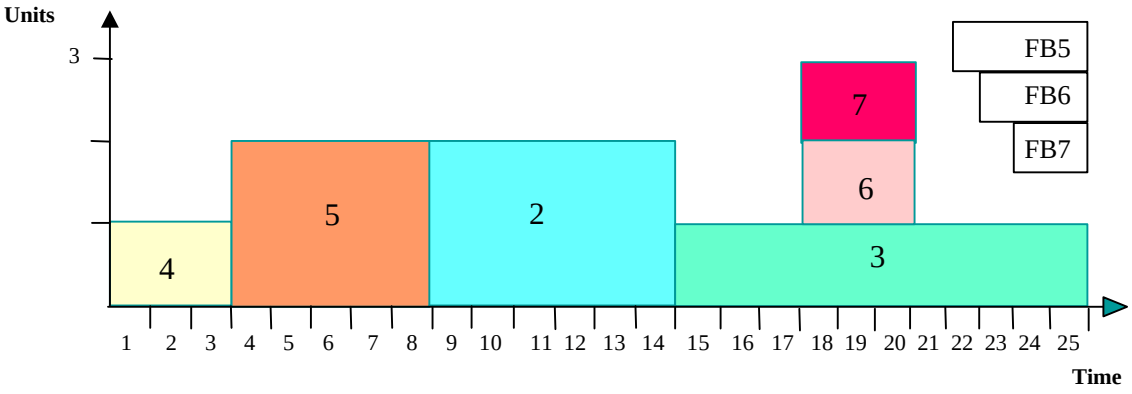


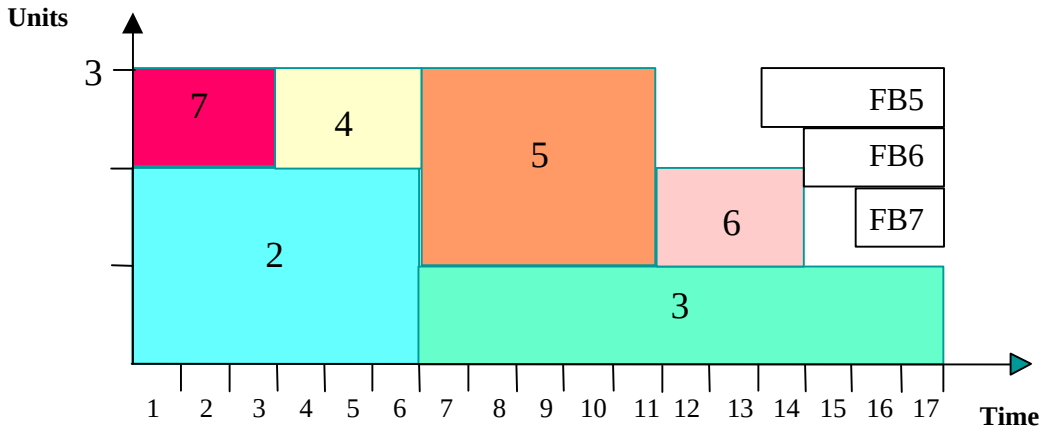
Figure 7



(a) Minimal duration schedule

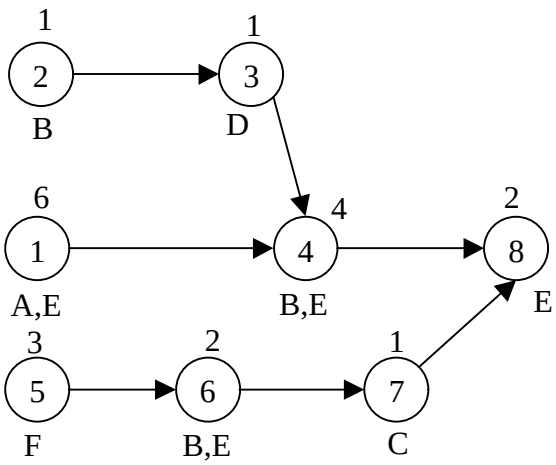


(b) Buffered schedule using push-back method



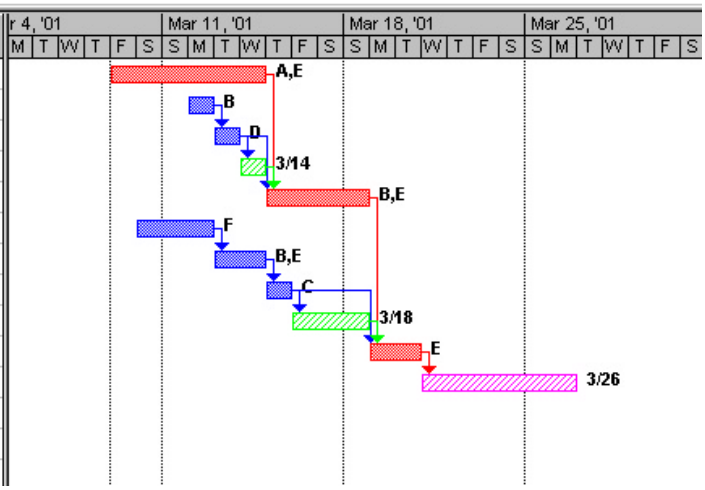
(c) Optimal buffered schedule

Figure 8



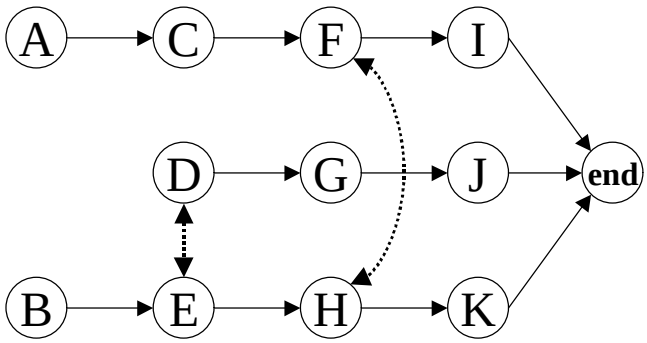
(a) Project network example

	Task Name	Duration
1	Activity 1	6 days
2	Activity 2	1 day
3	Activity 3	1 day
4	FB|Activity 3-3|Activity 4-4	1 day
5	Activity 4	4 days
6	Activity 5	3 days
7	Activity 6	2 days
8	Activity 7	1 day
9	FB|Activity 7-7|Activity 8-8	3 days
10	Activity 8	2 days
11	PB|Activity 8-8	6 days

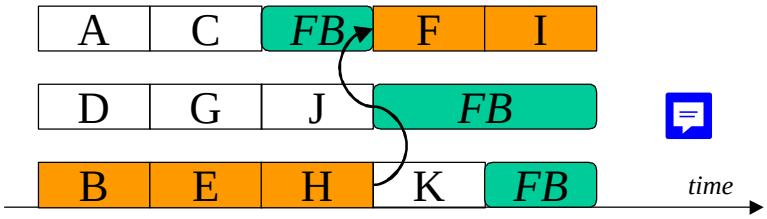


(b) Buffered baseline schedule

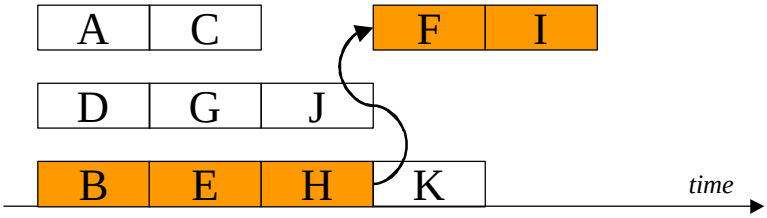
Figure 9



(a) Example project

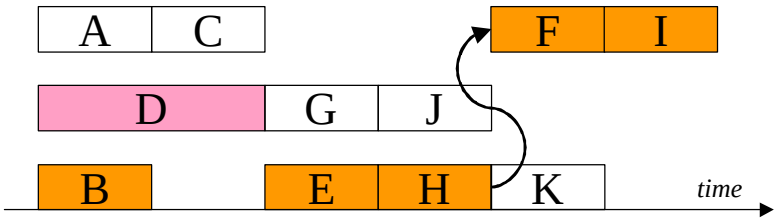


(b) Buffered baseline schedule

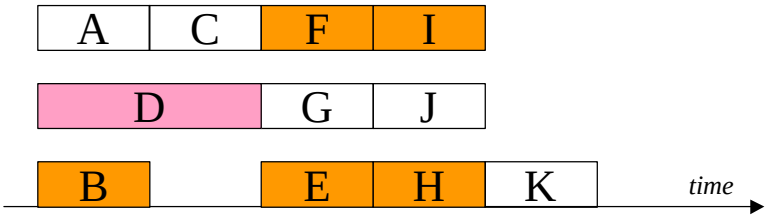


(c) Projected schedule

Figure 10



(a) Projected schedule keeping critical chain in series



(b) Projected schedule with new critical chain

Figure 11

BIOGRAPHICAL DETAILS

Willy Herroelen holds a Ph.D. in applied economics and is currently Professor in Operations Management at the Department of Applied Economics at the Katholieke Universiteit Leuven in Belgium. He teaches courses in the area of production scheduling and sequencing, facilities planning, organisation of the production process and project management. He has widely published on resource-constrained project scheduling. His current research activities focus on scheduling and sequencing under various resource and capacity constraints. He is a member of PMI and several professional management science and operations management associations.

Roel Leus holds a Commercial Engineering degree from the Department of Applied Economics at the Katholieke Universiteit Leuven (Belgium). He is currently research assistant of the Fund for Scientific Research – Flanders (Belgium) at the same department where he prepares a Ph.D. dissertation on robust and reactive project scheduling. He is a member of PMI and INFORMS.

Erik Demeulemeester holds a Ph.D. in applied economics and is currently Professor in Operations Management at the Department of Applied Economics at the Katholieke Universiteit Leuven in Belgium. He teaches courses in the area of quality management and production management. He has widely published on resource-constrained project scheduling. His current research activities focus on scheduling and sequencing under various resource and capacity constraints. He is a member of PMI and several professional management science and operations management associations.