Multi-Project Scheduling and Control: A Process-Based Comparative Study of the Critical Chain Methodology and Some Alternatives

IZACK COHEN, Industrial Engineering and Management, Technion Israel Institute of Technology, Haifa, Israel 32000

AVISHAI MANDELBAUM, Industrial Engineering and Management, Technion Israel Institute of Technology, Haifa, Israel 32000

AVRAHAM SHTUB, Industrial Engineering and Management, Technion Israel Institute of Technology, Haifa, Israel 32000

ABSTRACT

Critical Chain (CC) is a popular project management technique in many multi-project organizations. It applies the Theory of Constraints (TOC) to offer a practical and easy method for planning, scheduling and control of multi-project systems. While some prior studies examined CC performance for single-project management, little attention has been given to its performance in a multi-project environment. In this paper, we examine the control mechanisms of CC and some alternatives. We demonstrate that, when CC is not enough to prevent projects' lateness, such alternatives may give rise to similar and sometimes better, possibly much better performance.

Keywords: multi-project management; critical chain; project scheduling; control

Vol. 35, No. 2, 39-50, ISSN 8756-9728/03

Introduction

Critical Chain (CC) methodology for project management (Goldratt, 1997) applies the Theory of Constraints (TOC) to multi-project scheduling and control. Specific software packages, based on CC methodology, have been developed (Speed to Market's Concerto and ProChain Solutions Inc.). In parallel, a growing number of articles relating to CC have been published—some are criticizing the approach (Herroelen & Leus, 2001; Herroelen, Leus, & Demeulemeester, 2002; Shou & Yeo, 2000) and others are praising it (Steyn, 2000; Leach, 1999). None of these papers have thoroughly examined the performance of CC methodology in a multi-project environment. Yet, a growing number of multi-project organizations have chosen the CC methodology for planning, scheduling, and control of their projects, and that has motivated our efforts to understand it better.

It is thus our intention to obtain new insights on the CC methodology in a multi-project environment, while comparing its performance with alternative methodologies. To this end, we consider an environment of multiple concurrent projects in which projects compete for the same set of scarce resources. The environment is random (stochastic) in that uncertainty plays a significant role. Projects are unique in that their operational requirements and activity durations differ. Yet, projects are also "non-unique" in that they share common characteristics that enable their classification; for example, within a class, precedence relations between projects' activities can be identical and, for each activity, the historical realizations of activity times fit a common distribution function. According to co-author Cohen's experience, such an environment prevails in organizations that process maintenance or retrofit projects, which are common in the aircraft industry (for a description of maintenance projects in the aircraft industry, see also Gemmill & Edwards, 1999). Here, the realization of each project in the project portfolio is unique (e.g., due to its unexpected delays, different technical findings, or even differences in aircraft structures). However, despite the differences between the projects, one can model such an organization as one that processes several classes of projects with, for example, each different type of aircraft giving rise to a different project class. This approach of separating projects in a multi-project organization into different classes has been found useful

in past research (e.g., Adler, Mandelbaum, Nguyen, & Schwerer, 1995, for product development projects in the chemical industry; Leung, 2002, for software maintenance projects; and Griffin, 2002, who suggested a general classification of all projects in an organization into four project classes).

A natural modeling framework for non-unique multi-projects was described by Adler et al. (1995), in terms of stochastic processing networks. The building blocks of such a network are interdependent resources that process project activities according to some pre-specified discipline. At any given moment, each activity is either receiving service from a resource, queuing up for access to a resource, or waiting to join a prerequisite activity that is being processed or delayed elsewhere. The network model is stochastic, or random; for example, activity durations are modeled by random variables. Randomness here captures unpredictable variability that is prevalent and significant in most project environments.

The stochastic network paradigm is also natural for the analysis of buffer management, as defined by the CC methodology. A buffer stores "time-units," specifically time by which the project activities corresponding to this buffer could be delayed without causing a delay to the planned project due date. A buffer, thus, stores slack time that is added during planning. Buffer management then amounts to the dynamic management of resources according to buffer contents or, equivalently, to buffer consumption levels. For example, among several competing activities, top priority in resource allocation is given to the activity whose buffer consumption is the highest, namely its slack time is the least.

Through the comparison of CC to other alternative methodologies, we address two aspects of broad significance to project management. The first is the trade-off between resource utilization and project throughput: project throughput times get longer as resources' utilization become higher. We quantitatively demonstrate this trade-off for the different management methodologies. The second aspect is implementation costs. Here we demonstrate that some simple management methodologies, requiring low implementation costs, can achieve very good performance compared to the CC methodology. We note that the implementation of CC methodology in an organization usually requires some organizational changes and considerable implementation costs (mainly training costs for both management and workers, and purchase costs for special software packages). Buffer management gives rise to additional ongoing costs.

The rest of the paper is organized as follows: In the next section, we discuss the fundamentals of CC methodology. Then, we introduce the process management approach for modeling dynamic multi-project environments. Next, we elaborate on our experimental design, accompanied by a section with notation and formulas. In the following two sections, we present our main results: a comparative simulation analysis of the CC methodology for the management of non-unique multi-projects. We then conclude with a summary and some suggestions for further research.

Fundamentals of CC Methodology

CC methodology (Goldratt, 1997) aims at developing a sound schedule, using buffer management, in order to avoid project overruns. The methodology is not well defined in the sense that it does not provide precise definitions for some project entities and scenarios. Rather, it gives a heuristic framework and guidelines for project managers on how to plan, schedule, and control their projects, and it is up to the user of the method to complete the details. (For further discussion of the merits and pitfalls in CC methodology, refer to Herroelen and Leus, 2001.) We now review the steps of CC methodology as they apply to the models discussed in the sequel. We start with a single-project environment, and then generalize to a multi-project environment.

The CC steps for **single** project planning, scheduling, and control are as follows:

Step S1: Reduce activity durations by eliminating safety margins.

Estimates of activity durations include "padding" times. Indeed, based on their experience, managers tend to quote late due dates so that they can meet them with a high degree of certainty. The result is inflated activity durations that become self-fulfilling, or even overrun due to the combined effects of stochastic variability and Parkinson's Law (whereby work expands to fill the time given for execution; Parkinson, 1957). Therefore, CC methodology, applied at the outset, reduces predicted activity times to their median (which ensures 50% probability of on-time completion, Goldratt, 1997) or to their average duration (Product Development Institute, 1999; Herroelen & Leus, 2001).

Step S2: Identify the critical chain.

A critical chain is a sequence of activities that determines the project duration, taking into consideration both precedence dependencies and resource constraints. Such a critical chain arises from a project plan that assumes deterministic estimations for the reduced activity durations and resource requirements, as in Step 1. (When a critical chain is non-unique or difficult to identify, the advice is to pick one up arbitrarily.) The project plan is then further revised according to the "late starts" of the project activities.

Step S3: Create a project buffer.

Some of the padding that was eliminated in Step 1 from activity durations is shifted to the end of the critical chain and "stored" in a time buffer. This buffer, called project-buffer, is used dynamically to protect the project due date against variations in critical chain activities. A standard approach is to set the project-buffer capacity to 50% of the total duration of the critical chain (Leach, 1999).

Step S4: Create feeding buffers.

Delays in non-critical activity chains (activity chains merging into the critical chain) could cause undesir-

able delays of the critical chain. Consequently, feeding buffers are added at the end of each non-critical activity chain ("pushing" the latter back in time, in response to a "late start"). The feeding buffers thus protect the critical chain from variations of non-critical chains and allow critical-chain activities to start early, when possible. According to Leach (1999), a feeding-buffer capacity is set to 50% of the duration of its non-critical activity chain.

Step S5: Control.

Buffer monitoring provides a quick grasp of project status, which, in turn, enables adaptive control. Specifically, buffer consumption that reaches a predefined threshold (e.g., two-thirds of the buffer size or, equivalently, one-third of the slack time remains unused; Leach, 1999) triggers an early warning toward taking some preventive managerial action. More details are provided later in this paper.

Multiple projects are accommodated by combining single-project scheduling with TOC (Goldratt, 1984) and CC principles, notably the emphasis on reducing multi-tasking (Herroelen & Leus, 2001; Leach, 1999). To this end, project start-times are staggered, which turns the multi-project system into a "pull" system with newly determined release/start times. Following are the relevant details.

Scheduling and control of a **multi-project** system:

Step M1: Treat each project as a single project.

Individually schedule each of the multi-projects, using the four steps for scheduling a single project, as described in Steps S1–S4.

Step M2: Stagger projects according to the bottleneck resource.

First identify the bottleneck, namely the most constraining resource (often by simply using managerial experience). Then release projects sequentially, by staggering them, so that the bottleneck works continuously and there is no idle time.

Step M3: Create a capacity buffer.

A time buffer, called a capacity buffer, is associated with the bottleneck, and its role is to ensure bottleneck availability. The capacity buffer decouples between bottleneck activities that belong to successive projects, thus determining projects' start times. Since, based on a literature survey, there is no standard way to set the size of this capacity buffer, we set its base-case size at 50% of the duration of the bottleneck activity. We then analyze the effect of alternative sizes by varying the values through 8.3%, 16.7%, 83.3% and 116.7%.

From Project to Process Management

Following Adler et al. (1995), we model a multi-project organization as a stochastic processing network. Adler et al. (1995) validated the model based on an actual research and development organization, showing that the model simulated quite accurately its performance.

In the model of a stochastic processing network, each network node represents a group of (one or more) statistically identical resources, who perform the same type of activities and who are able to do so in parallel. When several activities of a project can start being processed at the same time, we refer to the phenomenon as a "fork;" when an activity cannot begin until its predecessor activities have been completed, we call it a "join." (Consequently, such models are often referred to as fork-join queues. For example, see Nelson & Tantawi, 1988.) The time required to complete an activity is called its processing time (duration) and the intervals between successive project releases are "inter-arrival times." The reciprocal of the

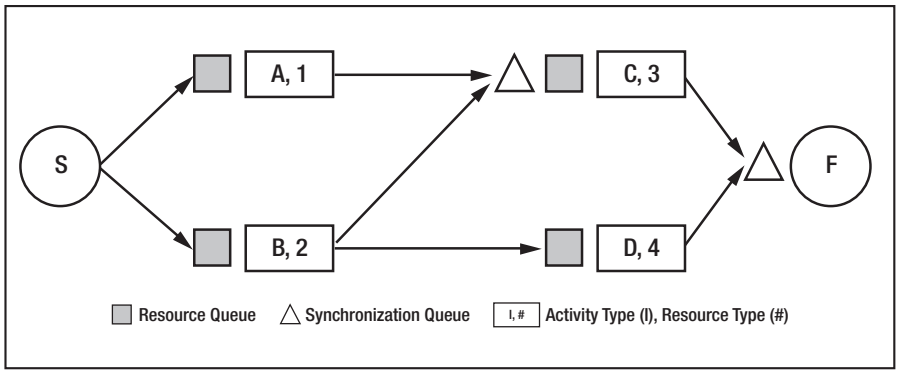


Figure 1. The Stochastic Processing Network approach for representing a multi-project system

Step M4: Control.

As with single projects, scheduling control of multi-projects is buffer-based: when allocating an idle resource, top priority is given to critical-chain activities over non-critical-chain activities; secondary priority is given to activities of projects with the highest level of project buffer utilization or, equivalently, the least slack time. Least priority, in turn, is given to activities of projects with the highest feeding buffer consumption.

	Resource Type	Number of Resources	Time Distribution
Inter-arrival			Exp(1/3.25)
Activity A	1	3	Exp(1/6)
Activity B	2	2	Exp(1/5)
Activity C	3	3	Exp(1/4)
Activity D	4	1	Exp(1/3)

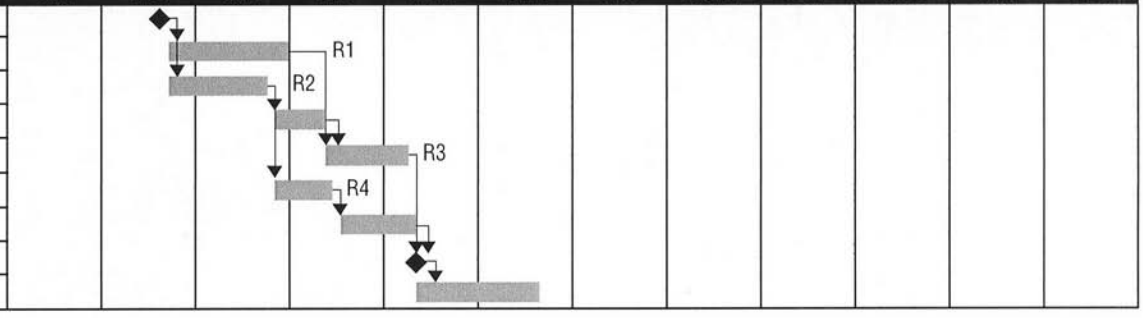
Table 1. Characteristics of our multi-project system: number of resource-units per type, processing time distribution and inter-arrival time distribution. The notation Exp(λ) represents an exponential distribution with probability density function $f(t)= \lambda e^{-\lambda t}$ (and expectation $1/\lambda$)

mean inter-arrival time is the project's input rate (expressed in number of projects per unit of time).

We use network diagrams, as in Figure 1, to illustrate activities' precedence requirements. The network is stochastic since inter-arrival times, processing times, and precedence requirements are subject to random (stochastic) variability. Projects are of the same type if they are characterized by the same set of probability distributions

a

ID	TASK NAME
1	Start
2	Activity A
3	Activity B
4	Feeding Buffer 1
5	Activity C
6	Activity D
7	Feeding Buffer 2
8	Finish
9	Project Buffer



b

ID	TASK NAME
1	Start
2	Activity A
3	Activity B
4	Feeding Buffer 1
5	Activity C
6	Activity D
7	Feeding Buffer 2
8	Capacity Buffer
9	Finish
10	Project Buffer
11	Start
12	Activity A
13	Activity B
14	Feeding Buffer 1
15	Activity C
16	Activity D
17	Feeding Buffer 2
18	Capacity Buffer
19	Finish
20	Project Buffer
21	Start
22	Activity A
23	Activity B
24	Feeding Buffer 1
25	Activity C
26	Activity D
27	Feeding Buffer 2
28	Capacity Buffer
29	Finish
30	Project Buffer

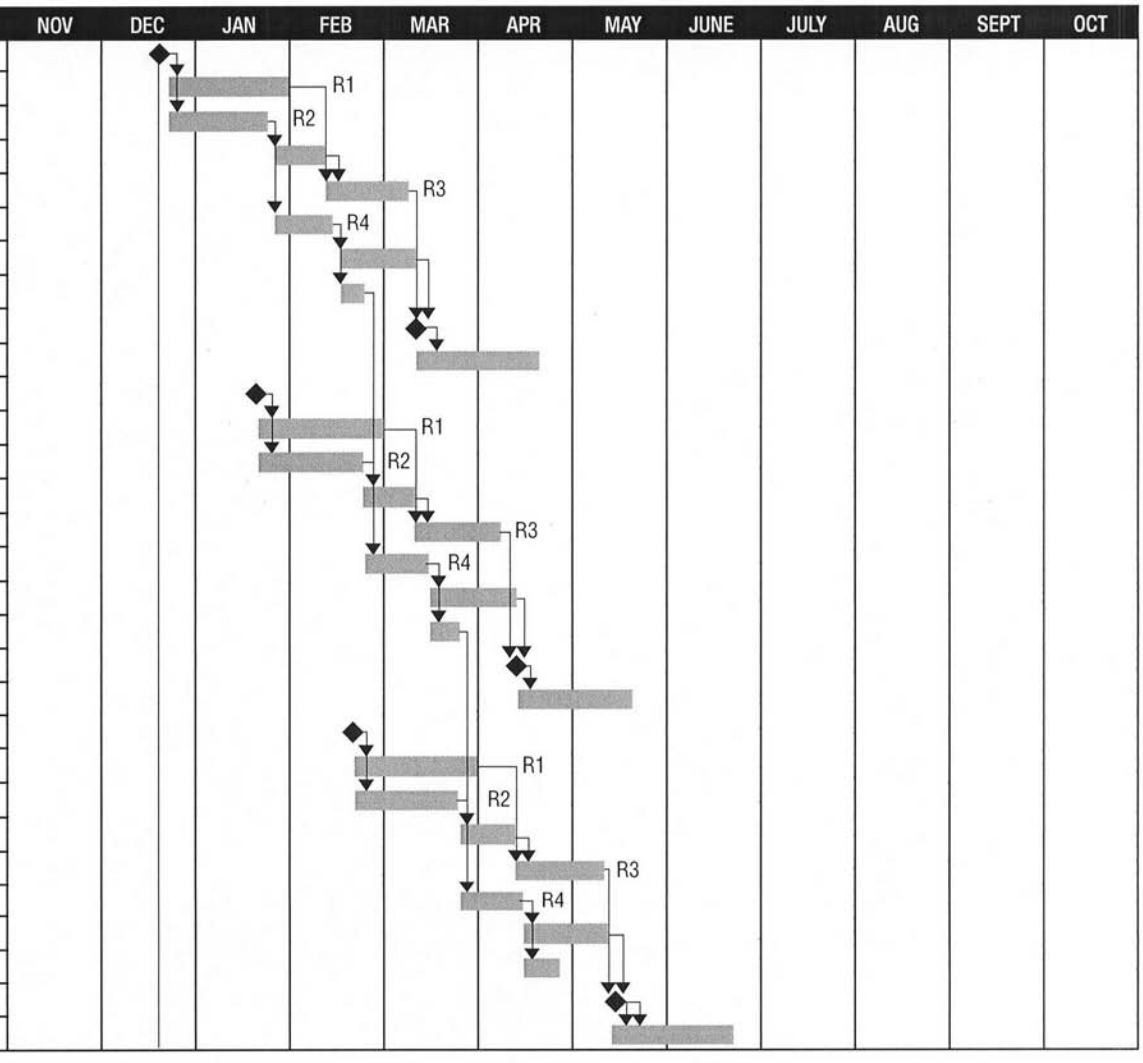


Figure 2. Critical Chain representation of a single project (a) and a corresponding multi-project system (b). For concreteness, one time unit is a week. The critical chain consists of Activity A and Activity C. Feeding buffers are added at the end of non-critical chains: the non-critical chain that includes Activity B (Feeding Buffer 1) and one that includes activities B and D (Feeding Buffer 2). At the end of the critical chain we add a project buffer. Capacity buffers decouple successive projects: such a buffer is placed after an activity performed by Resource 4 (bottleneck resource) in one project and this activity in the following project

(precedence, inter-arrival times, and processing times). This representation can thus model a multi-project system with different project types. When a project activity "arrives" to a node/resource, it either starts its processing immediately or it joins a queue and waits there till its processing starts. Such queues are called resource-queues—they are managed according to priority rules and are subject to resource availability. Another type of waiting takes place in synchronization queues, where activities are delayed due to precedence constraints. We shall now make these abstract notions concrete via a simple example of a processing network.

Model Construction

Consider the multi-project system depicted in Figure 1. The system has four resource types, numbered 1–4, which process projects of a single type. Each project consists of four activities, denoted A–D: Type 1 resources are dedicated to processing type A activities, type 2 resources are dedicated to type B, etc. (The more general model could have a resource type processing several activities.) The start and finish activities are milestones— they have neither a duration nor a resource requirement. Figure 1 could be viewed as representing a simple multi-project organization, for example, an aircraft maintenance company or a chemical product development process.

The system characteristics are given in Table 1. From the table, we read that release times between successive projects have an exponential distribution with a mean 3.25 units of time, that three resources of type 1 are dedicated to processing activities of type A, and that the processing durations of such type A activities have an exponential distribution with an average of 6 units of time. (The dependence of performance on the distribution of the processing-duration will be discussed in our concluding section.) The bottleneck resource, namely the resource that determines the system's processing capacity (Step M2), is resource 4. This choice finds ample support in our subsequent analysis (see Table 2). But it can be roughly justified, by observing that a mere single type 4 resource is dedicated to activity D, with an anticipated utilization level of about 3/3.25 = 92% in steady state, which is by far the highest among all the resources.

Figure 2 illustrates the buffered single- and multi-project systems, according to the CC scheduling methodology described in Steps S1–S4 and M1–M3, respectively; as already noted in Step M3, we set the size of the capacity-buffer to 50% of the duration of the bottleneck activity D.

Experimental Design

Our experimental tool is a simulation model, written in Visual Basic on a personal computer. Each simulation run started with a warm-up period

λ	Instance	No Control	CC	MinSLK	QSC(6)	QSC(3)	ConPIP
0.31	$\bar{X}$	51.42	32.44	30.40	20.18	17.33	32.93
	$t_{n-1,1-\alpha/2}\, s/\sqrt{n}$	3.86	1.24	0.88	0.19	0.11	0.69
	SD	33.46	18.92	17.67	9.42	8.64	13.45
	$\rho_1,\rho_2,\rho_3,\rho_4$	62,78,42,93	61,78,41,92	61,76,40,90	58,72,39,86	52,66,35,78	61,76,41,92
0.29	$\bar{X}$	32.44	21.87	21.59	18.62	16.62	22.79
	$t_{n-1,1-\alpha/2}\, s/\sqrt{n}$	1.75	0.41	0.37	0.13	0.12	0.20
	SD	20.55	11.15	11.57	8.80	8.28	9.44
	$\rho_1,\rho_2,\rho_3,\rho_4$	57,72,38,85	57,72,38,85	57,71,38,85	56,69,36,82	51,63,33,75	57,72,38,86
0.22	$\bar{X}$	18.9	15.50	15.27	15.25	14.60	15.33
	$t_{n-1,1-\alpha/2}\, s/\sqrt{n}$	0.27	0.12	0.09	0.07	0.12	0.07
	SD	10.10	7.71	7.97	7.74	7.52	7.56
	$\rho_1,\rho_2,\rho_3,\rho_4$	45,56,30,67	45,56,29,67	44,55,29,66	45,56,29,66	42,53,28,63	44,55,30,67
0.18	$\bar{X}$	16.49	14.17	14.26	14.26	14.12	14.81
	$t_{n-1,1-\alpha/2}\, s/\sqrt{n}$	0.17	0.08	0.06	0.05	0.09	0.07
	SD	8.72	7.32	7.53	7.48	7.44	7.52
	$\rho_1,\rho_2,\rho_3,\rho_4$	37,46,25,55	36,45,24,55	37,46,24,55	37,46,24,56	36,45,23,55	36,45,24,54
0.15	$\bar{X}$	15.32	13.71	13.84	14.00	13.69	14.48
	$t_{n-1,1-\alpha/2}\, s/\sqrt{n}$	0.10	0.07	0.05	0.12	0.06	0.06
	SD	7.99	7.18	7.42	7.23	7.33	7.51
	$\rho_1,\rho_2,\rho_3,\rho_4$	32,40,21,48	31,39,20,46	31,39,20,47	32,39,21,48	31,38,20,46	31,38,20,46

Table 2. Summary for different control methodologies. NPIP values for ConPIP control are: 13,8,4,4,4 for throughput rates 0.31,0.29,0.22,0.18,0.15, respectively. QSC controls' λ_{eff} values corresponding to λ= 0.31,0.29,0.22,0.18,0.15 are: QSC(6) 0.29,0.27,0.22,0.18,0.15, respectively. QSC(3) 0.26,0.25,0.21,0.18,0.15, respectively

that lead to steady state. This transient phase was discarded from the analysis. The system was then simulated in steady-state for a predefined time interval that was chosen according to Welch's moving average procedure, as described in Law and Kelton (1991). Each simulation was replicated 50 times.

The experiment compares system performance under CC with performance under alternative control methodologies. Performance measures are mean project duration, its standard deviation, and throughput rate, that is, the number of completed projects per unit of time.

We analyze two types of controls: open control, under which all candidate projects are actually initiated, and closed or semi-closed control, where projects must adhere to some predefined criteria in order to be started. Our open controls are termed No Control, CC and MinSLK; the closed and semi-closed group consists of ConPIP and QSC. Here are their details:

Open Controls:

1. **No Control**—A push system with FCFS (first come first served) queues priority rules.

(A thorough analysis of this specific model is carried out in Barron & Mandelbaum, 2003.)

2. **Critical Chain (CC)**—New projects are initiated according to the CC methodology, as was described in Step M2-M3. Guided by Step M4, buffer management determines the priority of the next activity to be processed by a resource.

When viewing CC methodology as a process management model, we make the following observations. Buffer consumption determines the priority of its corresponding activity in a resource queue: the higher the consumption (less slack), the higher the priority. While an activity is delayed in queue (either synchronization or resource), its buffer consumption increases until it reaches a predefined threshold. In this scenario, the threshold is two-thirds of the associated buffer size, for both feeding and project buffer, and when it is reached, a resource is allocated to the activity by preempting activities with lower buffer consumption.

3. **Highest Priority in Queue to a Minimum Slack Activity (MinSLK)**—When an activity is completed, the prevalent critical path is reevaluated and slack times are updated for the rest of the projects' activities. (Here, slack-time is defined as the difference between the late start time and early start time.) MinSLK (also called MinSLK[DD] by Bock & Patterson, 1990) employs the following priority rule for allocating resources to activities: the lower the slack time, the higher the priority. Thus, as a project is delayed, the priorities of its activities increase. (Note that slack times can turn negative, specifically when the start time of an activity is later than the late start time that is needed in order to complete the project on its due date.)

Closed and Semi-Closed Controls:

4. **Constant Number of Projects in Process (ConPIP)**—New projects are started based on a predetermined number of projects in process, called NPIP (Adler et al., 1995; and Anavi-Isakow & Golany, 2003). Specifically, an arriving project starts its processing immediately if the number of projects concurrently in process within the system is below NPIP; otherwise, it is placed in an external queue and waits till it can be processed. We define the throughput time of a project as the time span from its start-time (S), when it leaves the external queue, until its finishing time (F), when it leaves the system. For varying values of project input rates, we determined NPIP values as the maximal number of projects allowed concurrently in the system, so that the mean waiting time in the external queue does not exceed half of the average throughput time. (The latter condition is plausible for the aircraft maintenance industry, according to the experience of one of the authors.)

5. **Queue Size Control (QSC)**—A predetermined maximal number of activities is allowed, at any given time, within the resource queue of the bottleneck. An arriving project is then allowed into the system to be processed if the length of the bottleneck's resource queue is below this maximal number; otherwise, the arriving project is discarded, never to return.

Notation and Formulas

Consider n replications, each of length m; Y_{ij} is the duration of the i^{th} project from the j^{th} replication ($i=1,2,\ldots,m; j=1,2,\ldots,n$). The average project duration for replication j is taken to be (Law & Kelton, 1991):

$$X_j=\frac{\sum_{i=l+1}^{m} Y_{ij}}{m-l},$$

where l is the length of the transient period, counted in number of projects.

Assuming that replications are independent, the X_j's are independent random variables with $E(X_j)\approx\mu$ (the true steady-state mean value). The overall mean duration $\bar{X}$, where

$$\bar{X}=\sum_{j=1}^{n}\frac{X_j}{n}$$

is approximately an unbiased point estimator for μ. An alpha-level confidence interval is given by:

$$\bar{X}\pm t_{n-1,1\alpha/2}\frac{S}{\sqrt{n}},$$

where S^2, the estimator for the variance equals

$$S^2=\frac{\sum_{j=1}^{n}(X_j-\bar{X})^2}{n-1}.$$

The mean standard deviation (SD) of a replication is

$$SD=\frac{\sum_{j=1}^{n}\sqrt{\frac{\sum_{i=l+1}^{m}(Y_{ij}-X_j)^2}{m-l-1}}}{n}.$$

The arrival rate of projects, λ, is a given parameter, while the effective throughput rate, λ_{eff}, is a measured output of the model—or simulation. For each resource type, we estimate its mean traffic intensity ρ_k by

$$\rho_k=\frac{\sum_{j=1}^{n}\frac{\sum_{i=l+1}^{m}T_{jik}}{U_j}}{n*N_k}.$$

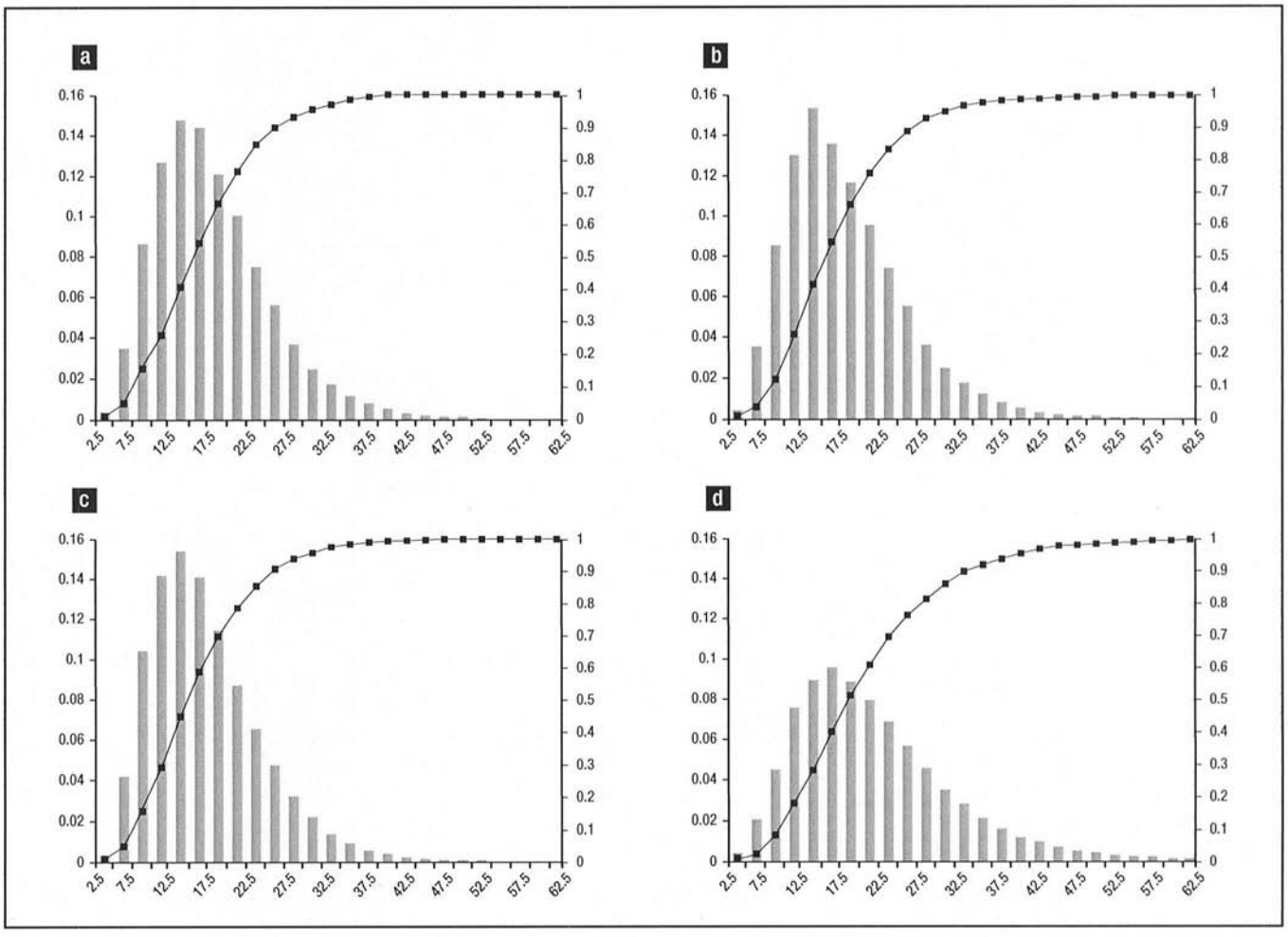


Figure 3. Throughput time distribution for throughput rate, λ=0.22, for the following controls: (a) CC. λ_{eff}=0.22, $\bar{X}$=15.34, SD=7.66, (b) MinSLK, λ_{eff}=0.22, $\bar{X}$=15.27,SD=7.97, (c) QSC (3). λ_{eff}=0.21,$\bar{X}$=14.60,SD=7.52 (d) No control λ_{eff}=0.22,$\bar{X}$=18.9, SD=10.10

Here, T_{jik} is the processing-time of project i by resource type k in replication j; U_j is the simulation duration in steady-state for replication j; and N_k is the number of resource units for resource type k.

Simulation Results

Table 2 summarizes the performance measures of our simulation experiments, in steady state. Applying CC methodology, specifically Step S2, to the multi-project system described by Figure 1 and Table 1, yields a project plan with throughput time of 17.5 time units and a throughput rate of 0.22 projects per unit of time. Our simulation analysis reveals a mean project throughput time of 15.50, which is below the planned 17.5 value. Nevertheless, 34% of the projects are expected to have a throughput time that exceeds the plan: P(T>17.5)=0.340, where T denotes a random variable that models project duration in steady state, and its distribution is here determined by the histogram in Figure 3a.

A reduction of the capacity buffer (from the 50% that the CC methodology often guides as default) results in an increase of both project throughput time and its standard deviation. Reduction of buffer capacity to 16.67% resulted in an increase of mean throughput time to 21.87 and a standard deviation of 11.15 (51.0% of the mean). Further reduction to 8.33% buffer size resulted in a mean throughput time of 32.65 and standard deviation of 18.92 (57.9% of the mean).

We compared the performance of the different control methodologies to CC. For each throughput rate, we performed a modified t-test confidence interval test, at the 0.95 confidence level (using Welch's approach for comparing performance measures without assuming equal variances; see Law & Kelton, 1991, pp. 588–594).

When the system operated with No Control, the mean and standard deviation were significantly higher than in CC, for all throughput rates.

Performance of the MinSLK priority rule was not significantly different from CC at throughput rates of 0.29, 0.22 and 0.18, and slightly better for throughput rate 0.31. The throughput time distribution for throughput rate 0.22 (see Figure 3b) was similar to that of CC, with P(T>17.5)=0.341.

Queue Size Control gave rise to lower values of both mean throughput time and standard deviation. The improvement is most noticeable at high utilization levels and, hence, high throughput rates (0.31, 0.29 and 0.22). However, there is some decrease in the effective throughput rate, as described

below in Table 2. For example, in arrival rate of 0.31 project per time unit, reducing the mean traffic intensity from about 90% (in CC) to 85% (in QSC(6)) resulted in a 38% decrease in mean throughput time, while the effective throughput rate reduction was 6.4%.Figure 3c presents the throughput time distribution for an effective throughput rate of 0.21 (max queue size of 3 projects). In this case, the throughput time for 30% of the projects is over 17.5 time units: P(T>17.5)=0.300.

In Figures 4a–4d we display the throughput time distribution for different controls under a relatively heavy load (high throughput rate of 0.31). CC and MinSLK exhibit a very similar distribution, with over 70% of projects expected to exceed the throughput time of 17.5 time units, and the medians equal to 32.44 and 30.40, respectively. The performance of the uncontrolled system is no less than catastrophic: for about 90% of the projects; the throughput time is expected to be more than 17.5 and, indeed, the median increases to 51.42. In contrast, the closed control QSC(3) exhibits fairly good performance, with only 40% of the projects expected to be late for their planned due date, and a median of 17.33. Note that this is at the cost of an effective throughput rate of 0.26, which amounts to giving up about 16% of the projects.

ConPIP control has not shown a significant difference in mean throughput time relative to CC. The standard deviation and confidence interval for the mean throughput time (0.31 and 0.29, respectively) were lower for high throughput rates.

The experience of projects, as they progress through the system, is summarized in Table 3. We are using, what we call, time-profiles: these are the fractions of time that projects spend either waiting for a resource, waiting for activity synchronization, or actually being worked on. Specifically, Table 3 displays time-fractions for the following performance measures: Waiting time in resource queues; Synchronization time (where activities wait until their precedence constraints are fulfilled); and Processing time of activities by the resources.

From Table 3, we learn that a significant fraction of projects' throughput time is spent waiting, either in synchronization or resource queues. For example, the time profile for CC with throughput rate 0.22 is as follows: processing time 52%, waiting for resources 13%, and waiting in synchronization queues 35%. As the throughput rate gets higher, the differ-

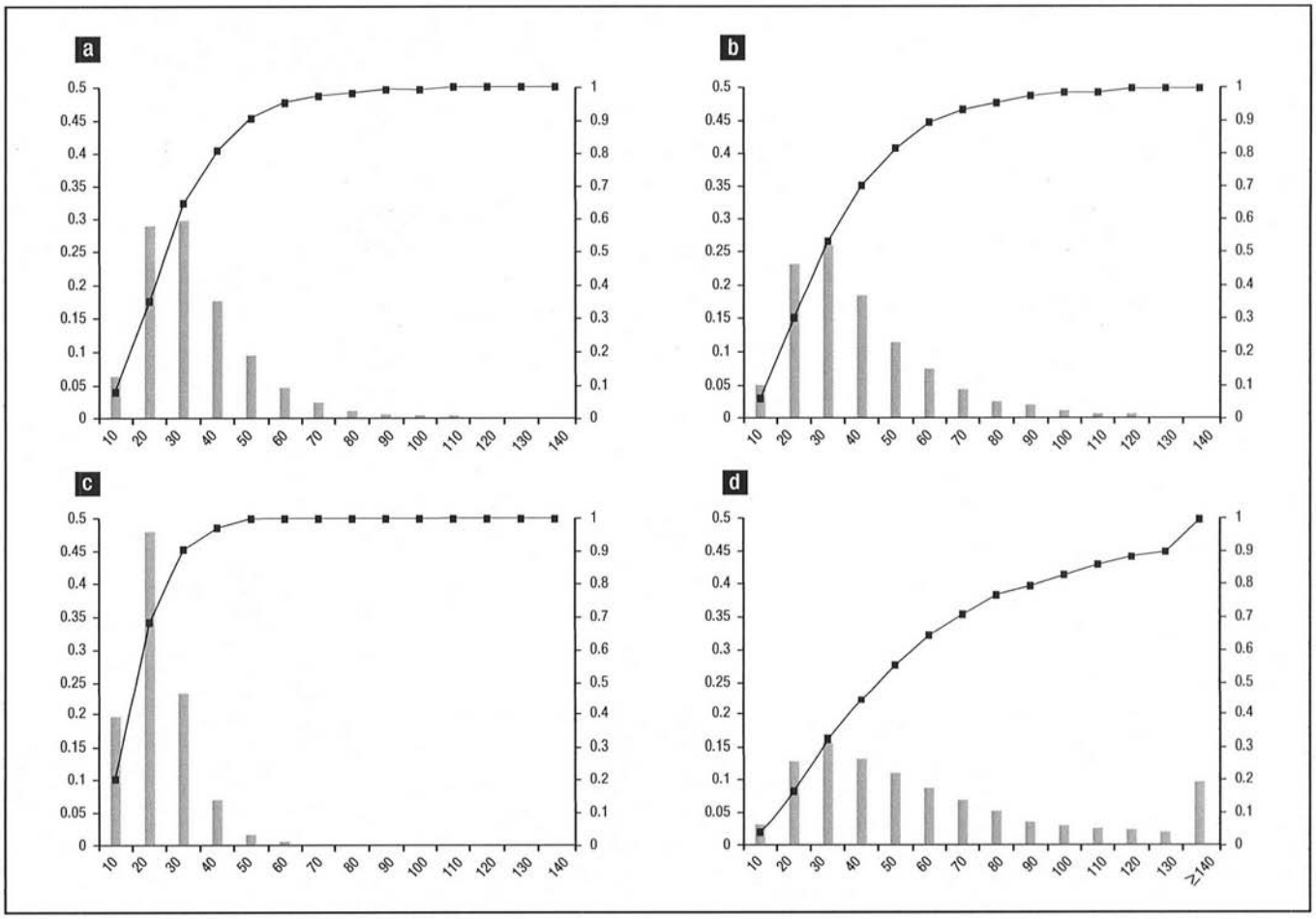


Figure 4. Throughput time distribution for throughput rate, λ=0.31, for the following controls: **(a)** CC. λ_{eff}=0.31, $\bar{X}$=32.44, SD=18.92, **(b)** MinSLK, λ_{eff}=0.31, $\bar{X}$=30.40,SD=17.67, **(c)** QSC (3). λ_{eff}=0.26,$\bar{X}$=17.33,SD=8.64 **(d)** No control λ_{eff}=0.31,$\bar{X}$=51.42, SD=33.46

λ	Time Profile	No Control	CC	MinSLK	QSC(6)	QSC(3)	ConPIP
0.31	% waiting time	43	36	38	24	18	38
	% synchronization time	40	38	35	34	34	36
	% processing time	17	26	27	42	48	26
0.29	% waiting time	36	27	26	21	16	29
	% synchronization time	37	35	35	34	34	34
	% processing time	27	38	39	45	50	37
0.22	% waiting time	22	13	12	11	9	12
	% synchronization time	34	35	35	35	36	35
	% processing time	44	52	53	54	55	53
0.18	% waiting time	15	7	7	7	6	9
	% synchronization time	35	36	36	36	37	36
	% processing time	50	57	57	57	57	55
0.15	% waiting time	20	5	5	5	4	8
	% synchronization time	32	37	37	37	37	36
	% processing time	48	58	58	58	59	56

Table 3. Time profile analysis. Waiting time in resource queues, synchronization time and processing time

ence between the time profiles in an open system and in a closed system gets larger. For example, CC with a throughput rate 0.31 exhibits a time profile consisting of processing time: 26%, waiting for resources: 36%, and waiting in synchronization queues: 38%. Under QSC(3) with the same throughput rate, this profile changes to processing time: 48%, waiting for resources: 18%, and waiting in synchronization queues: 34%.

Discussion

The most important trade-off that an organization's management should consider is that between resource utilization and project throughput time: typically, the higher the former, the longer the latter, and vice versa. The first-order question is, thus, whether to work at high traffic intensity levels and have long throughput times or, alternatively, lower traffic intensity to gain lower throughput times and lower standard deviation, hence, also more predictability.

Research on multi-project planning based on CC (Leach, 1999) claims that resource competitions are resolved via buffer management, which protects against schedule variations. The claim is based on a small-scale example of a system that processes few projects; hence, it is in a transient phase of operation. We, on the other hand, are analyzing steady-state phenomena: buffers and queues have filled up to their steady-state levels and, consequently, they are prone to being incapable of absorbing all stochastic variations.

Figure 5 demonstrates variation in throughput time during the transient period until the system reaches its steady state. It is clear that analyzing only the transient period would bias the estimate of throughput time to be greatly overoptimistic. Indeed, the buffers are "unchallenged" during this transient period. In contrast, some steady-state scenarios are such that the buffers fail to protect the planned due date for a significant fraction of the projects.

A comparison of alternative control methodologies to the CC methodology shows that, for a given throughput rate, we can get the same or better performance using the priority rule MinSLK. This is done through dynamic control of the project, determination of its current critical path, and prioritizing the activities with minimum slack. This MinSLK priority rule and buffer management are almost identical—both give priority to the critical activity (critical in the sense that it is the latest activity, or the activity with least slack). We believe that MinSLK outperforms CC in higher loads because of its adaptive determination of the critical path—it helps determine the current "most critical" activities to allocate resources. Buffer management, in contrast, always gives priority to activities belonging to a pre-determined critical chain, which, as throughput rates increase, is likely to have changed.

When turning the system into a closed system, as done in Queue Size Control, one can achieve a dramatic reduction of the mean throughput time—in high throughput rates—by reducing the mean traffic intensity of the bottleneck resource. For example, in an arrival rate of 0.31 projects per time unit, reducing the mean traffic intensity from about 90% (in CC) to 85% (in QSC(6)) resulted in a 38% decrease in mean throughput time, while the effective throughput rate reduction was 6.4%. This conclusion, which is also supported by Queueing Theory, suggests that a modest reduction in the traffic intensity of a highly loaded bottleneck is likely to result in a meaningful reduction of throughput time: its mean and in fact the percentiles (see Figure 6). This reduction can be implemented in many differ-

ent ways: for example, increasing resource allocation of the bottleneck or turning the system into a closed system by not bidding on a small fraction of the potential projects.

As we stated in the introduction, our model applies to maintenance lines in the aircraft industry (both civilian and military). Here, ConPIP control is natural, since such organizations typically have some flexibility in altering the preplanned project start time. This is achieved without incurring costs to the customer, who continues to utilize the aircraft. It is especially true, and can, in fact, become a necessity, when the organization is doing fleet maintenance for a customer. In that situation, the customer normally cannot release an aircraft to maintenance until another aircraft returns to service.

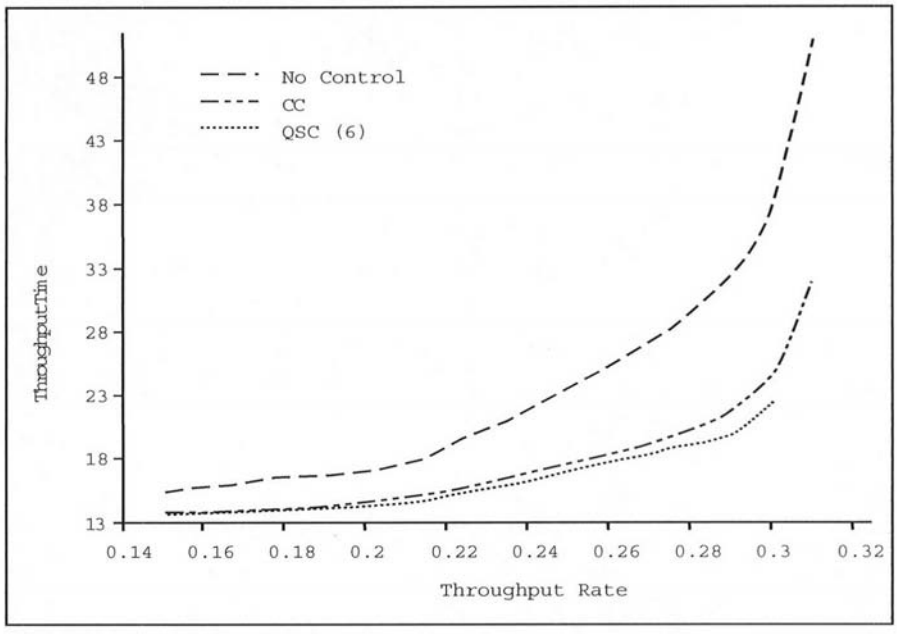


Figure 6. Throughput time as a function of throughput rate, for three controls: No control, CC and QSC (6)

ConPIP-based control gives rise to a performance that is similar to the other open control policies. For a high throughput rate, ConPIP has had a stabilizing effect that resulted in a lower standard deviation and tighter confidence intervals of mean throughput time. Applying ConPIP control to an organization of the kind we are discussing involves a very small effort compared to CC or MinSLK. The key is in identifying the desired NPIP, namely a limit to the number of projects that are allowed to exist concurrently within the system: large NPIP yields high throughput rates and possibly excessive throughput times; reducing NPIP will improve the latter but at a cost to the former.

It is worth noting that Anavi-Isakow and Golany (2003) improve ConPIP performance further by applying more sophisticated priority rules for queue management, rather than the standard FCFS that we have used.

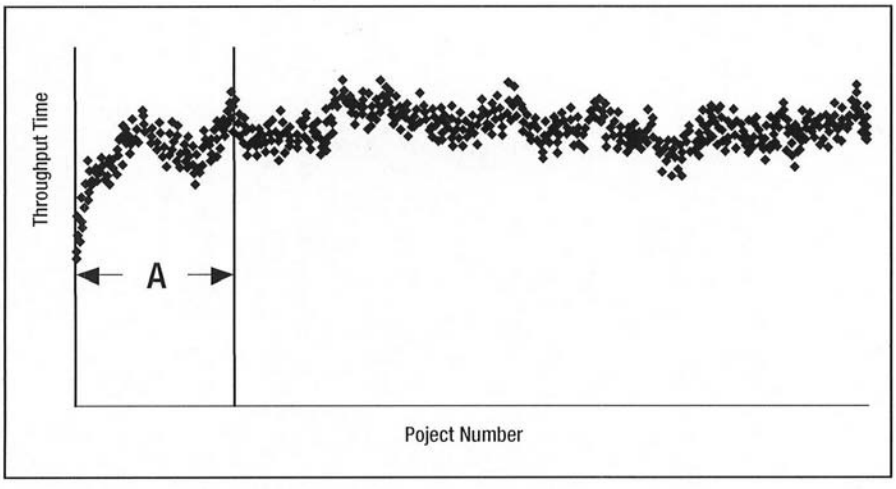


Figure 5. An example of projects' throughput time as it changes over a transient warming period (A) toward equilibrium

Conclusions

Critical Chain (CC) methodology is a popular project management technique in many multi-project organizations. It offers an intuitive method for planning, scheduling, and control of multi-project systems. CC acknowledges correctly the interaction between activities' precedence relations and resource constraints. Time buffers (feeding, project and capacity buffers) are introduced as a systematic method for dealing with stochastic (unpredictable) variability. The methodology offers some loose guidelines for choosing the buffer sizes. Buffer management is applied to control projects' progress, spot schedule deviations, and act correctively when a buffer is consumed beyond a certain predefined threshold.

Our study examines control mechanisms in a multi-project environment, which is typical of organizations in the aircraft industry. We demonstrate that buffer management may not be enough to meet the planned schedule. Some control methodologies such as QSC, ConPIP, and MinSLK can give similar and sometimes better performance. In particular, turning the system into a closed system (for example, QSC) enables one to work in higher throughput rates than those recommended by CC, while still maintaining desired delays. (Here, the comparison is against CC with capac-

ity buffer size set at 50% of the duration of the bottleneck activity—our base-case size)

We also demonstrate that ConPIP (Adler et al., 1995, Anavi-Isakow and Golany, 2003), which is a simple control policy that requires a minimal organizational investment, provides an effective alternative when there is flexibility in project start times. As already indicated, such is the case with maintenance or retrofit projects, which one of us has had experience with in the aircraft industries

We conclude with four observations that provide insight into the performance of multi-project systems:

1) Mean project throughput time is monotone increasing in throughput rate, and the better the control, the milder the increase. (Refer to Figure 6: a control is superior to another one if it gives rise to a shorter mean throughput time and lower stochastic variability for the same throughput rate; for example, QSC (6) controls the system better than CC.)

2) Some empirical evidence has suggested that the exponential distribution provides a reasonable fit to project duration times. But in case it does not, our framework can easily accommodate any other distribution, including actual empirical distributions. Moreover, our insights and conclusions remain intact.

3) An accurate inference of statistical characteristics is highly advisable. Indeed, changing the distribution of processing times or project arrivals could change, sometimes dramatically, system performance. Roughly speaking, performance deteriorates as stochastic variability increases in either processing times or arrivals (see Hopp & Spearman, 1996, pp. 282–310). Consequently and practically speaking, standardization in either activity processing or project starts would reduce, under equal throughput rates, the throughput times; or alternatively, a higher throughput rate will be achievable at equal throughput times.

4) As an overall observation, we feel safe to conclude that most reasonable controls would improve the performance of an uncontrolled system, typically significantly in heavy traffic. But, to achieve further improvement would require more, for example additional resources and/or less projects admitted.

Further research is recommended for developing more robust scheduling and control mechanisms for multi-project stochastic environments. This research could examine different multi-project environments by starting with "mini-systems," as done here, and then implementing the findings in realistic environments. The hope is also that such simulation analysis will provide sufficient insight and stimulus for further theoretical research on planning, scheduling, and control.

References

Adler, P.S., Mandelbaum, A., Nguyen, V., & Schwerer, E. (1995). From project to process management: An empirically-based framework for analyzing product development time. *Management Science, 41*(3), 458–484.

Anavi-Isakow, S., & Golany, B. (2003). Managing multi-project environments through constant work-in-process. *International Journal of Project Management, 21*(1), 9–18.

Barron, Y., & Mandelbaum, A. (2003). Performance Analysis of Dynamic Stochastic PERT/CPM Networks, Technical Report, Industrial and Engineering and Management, Technion, Israel.(http://iew3.technion.ac.il/serveng2003/Lectures/DSPERT.pdf).

Bock, D.B., & Patterson, J.H. (1990). A comparison of due date setting, resource assignment, and job preemption heuristics for the multi-project scheduling problem. *Decision Sciences, 21*, 387–402.

Gemmill, D.D., & Edwards, M.L. (1999). Improving resource-constrained project schedules with look-ahead techniques. *Project Management Journal, 30*(3), 44–55.

Goldratt, E.M. (1984). *The goal.* Great Barrington, MA: The North River Press.

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

Griffin, A. (2002). Product development cycle time for business-to-business products. *Industrial Marketing Management, 31*, 291–304.

Herroelen, W., & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management, 19*(5), 559–577.

Herroelen, W., Leus, R., & Demeulemeester, E. (2002). Critical chain project scheduling: Do not oversimplify. *Project Management Journal, 33*(4), 48–60.

Hopp, W.J., & Spearman, M.L. (1996). *Factory physics: Foundations of manufacturing management.* Chicago: Irwin McGraw-Hill.

Law, A.M., & Kelton, W.D. (1991). Simulation modeling and analysis. Chicago: McGraw-Hill.

Leach, P.L. (1999). Critical chain project management improves project performance. *Project Management Journal, 30*(2), 39–51.

Leung, H.K.N. (2002). Estimating maintenance effort by analogy. *Empirical Software Engineering, 7*, 157–175.

Nelson, R., & Tantawi, A.N. (1988). Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers, 37*, 739–743.

Parkinson, C.N. (1957). *Parkinson's Law.* Cambridge: The Riverside Press.

Product Development Institute. (1999). Tutorial: Goldratt's critical chain method, a one-project solution. http://www.pdinstitute.com.

Shou, Y., & Yeo, K.T. (2000). Estimation of project buffers in critical chain project management. IEEE ICIMT 2000, 162–167.

Steyn, H. (2000). An investigation into the fundamentals of critical chain project scheduling. International *Journal of Project Management, 19*(1), 363–369.

IZACK COHEN is serving in the Israeli Air Force. He holds a BSc in Chemical Engineering and his MSc is in Materials Engineering from the Technion Institute of Technology in Haifa, Israel. He is currently a PhD candidate at the Faculty of Industrial Engineering and Management at Technion. Mr. Cohen has over 10 years of experience in managing engineering projects in different technological areas. His current research activities are focused on developing new methodologies for managing multi-project systems.

AVISHAI MANDELBAUM is the Benjamin and Florence Free professor in Operations Research and Service Engineering, at the Faculty of Industrial Engineering and Management at the Technion Institute of Technology in Haifa, Israel. He has a BSc in Mathematics and Computer Science, an MA in Statistics and a PhD in Operations Research from Cornell University. His first academic position was at the Graduate School of Business at Stanford University, where he became interested in Project Management. At Technion, Professor Mandelbaum has been teaching courses in probability, stochastic processes and Service Engineering. His recent research activities have concentrated on queueing networks and their applications to Service Operations, with a focus on teleservices, such as telephone call/contact centers. Professor Mandelbaum was an associate editor of *Mathematics of Operations Research (MOR)* from 1991–1999. He is currently an associate editor of the journals *Management Science, Queueing Systems Theory and Applications (QUESTA)* and *Manufacturing and Services Operations Management (MSOM).*

AVRAHAM SHTUB is the Sharon and Stephen Seiden professor of Project Management in the Industrial Engineering and Management faculty at the Technion Institute of Technology in Haifa, Israel. Professor Shtub has a BSc in Electrical Engineering from Technion (1974), an MBA from Tel Aviv University (1978), and a PhD in Management Science and Industrial Engineering from the University of Washington (1982). He is a senior member of the Institute of Industrial Engineering (USA) and a certified Project Management Professional (PMP) by the Project Management Institute (PMI). He is the recipient of the Institute of Industrial Engineering's 1995 "Book of the Year Award" for his book *Project Management: Engineering, Technology and Implementation* (co-authored with John Bard and Shlomo Globerson). Professor Shtub is also on the Editorial Boards of *IIE Transactions* and the *International Journal of Production Research,* and is a past board member of the Israeli chapter of PMI. Professor Shtub was a faculty member of the Department of Industrial Engineering at Tel Aviv University from 1984 to 1998, where he also served as a chairman of the department (1993–1996), before joining the Technion in 1998. He has been a consultant to industry in the areas of project management, risk management, and the design of production-operation systems, and has served as a visiting Professor at the Sasin Graduate School of Management at Chulalongkorn University in Thailand (the Kellogg MBA program), Cyprus International Institute of Management, and at the Owen Graduate School of Management of Vanderbilt University.