J Sched (2007) 10: 181–193
DOI 10.1007/s10951-007-0013-0

# Resource allocation in stochastic, finite-capacity, multi-project systems through the cross entropy methodology

**Izack Cohen · Boaz Golany · Avraham Shtub**

Published online: 11 May 2007


**Abstract** This paper addresses the problem of *resource allocation* in a *finite-capacity*, *stochastic (random)* and *dynamic multi-project system*. The system is modeled as a queuing network that is controlled by limiting the number of concurrent projects. We propose a Cross Entropy (CE) based approach to determine near-optimal resource allocations to the entities that execute the projects. The performance of the suggested approach is demonstrated through numerical experiments and compared to that of a heuristic, rough-cut based method.

**Keywords** Project management · Stochastic processes · Uncertainty · Resource allocation · Simulation · Cross entropy

## 1 Introduction

Traditionally, most of the research on multi-project management has focused on the *multi-project resource constrained scheduling problem* (*MPRCSP*) that assumes static, deterministic environments (e.g., Kurtulus and Davis 1982 and Speranza and Vercellis 1993). Papers addressing stochastic multi-project systems have started to appear only recently (e.g., Anavi-Isakow and Golany 2003; Cohen et al. 2005, Lee and Miler 2004). Still, a typical assumption in these papers is that the allocation of resources to the various work-centers, where the projects' tasks are performed, is given and constant. But, a common practice in many multi-project matrix organizations is to change the intermediate-term staffing assignments in order to improve the overall performance. Such a procedure may be referred to as tactical (intermediate-term) resource allocation (e.g., Hopp and Spearman 1996, p. 381). Thus, the problem of tactical resource allocation is an important one—ignoring it may lead to situations where some work-centers are heavily loaded while others may be partially idle. Tactical resource allocation is applicable to organizations in which the lower echelon management has the authority to determine the workforce mix within certain limits. In such organizations it makes sense to change the allocation of resources (for example, by hiring/firing or re-directing people from one resource group to another) to improve the organization's performance. Typically, such an organization starts with an initial assignment of resources that was done through prior planning. Updating the resource assignments is done on a periodical rather than continuous basis for the following two reasons: (1) the involved processes, such as hiring/firing, workforce training, discussions with the unions, etc., take time, and (2) frequent changes may cause organizational nervousness (Kropp and Carlson 1984) resulting, possibly, in reduced performance (e.g., throughput, work quality) and poor worker's morale.

The dynamic aspect of resource allocation adds considerable complexity to the problem and hence, the few papers that address such problems rely on either simulation or rough-cut approaches to solve it (e.g., Martien et al. 2002). Here, we propose a methodology aimed at finding nearly optimal resource allocation solutions within a finite-capacity, stochastic, dynamic multi-project system.

This paper treats a problem, which can be viewed as the "dual" of the problem that was treated in our previous work (Cohen et al. 2005), where we dealt with finite-capacity

I. Cohen · B. Golany (✉) · A. Shtub
Industrial Engineering and Management, Technion, Israel Institute of Technology, Haifa 32000, Israel
e-mail: golany@ie.technion.ac.il

loading by determining the number of projects of different types to be executed by a *given* system of resources. In this paper we treat the problem of determining the (tactical) allocation of resources for a *given* load of projects. We develop a Cross Entropy (CE) based approach (Rubinstein and Kroese 2004) to select near-optimal resource allocation solutions. We demonstrate the approach through CONPIP (Anavi-Isakow and Golany 2003), a methodology that controls the load in the system by limiting the maximal number of projects in process. Moreover, we illustrate through numerical examples the potential improvement in the overall system performance that might result from allocating resources with our proposed method as compared to situations in which resource allocations are fixed.

The paper starts with a brief discussion of the modeling approach followed by a development of a rough-cut based heuristic approximation procedure. Then we develop a CE algorithm. Next we explain the experiments and their outcomes. Finally, we offer some concluding remarks and directions for future research.

## 2 The modeling approach

### 2.1 A queuing theory based modeling approach

We consider an environment of multiple, concurrent, non-unique projects. The projects share common characteristics (for example, similar precedence relations among their activities), and they compete for the same set of finite resources. Such an environment is typical in organizations doing maintenance or retrofit projects in the aircraft industries (Gemmill and Edwards 1999). The realization of each project in the project portfolio is unique (e.g., due to its unexpected delays, different technical characteristics or, in the case of the aircraft industries, differences in aircraft structures). However, despite the differences among the projects, one can model such settings as a system that processes several classes of non-unique projects (e.g., different types of aircraft may be represented as different project classes). This approach of classifying projects into different classes has been found useful in past research (e.g., Adler et al. 1995, for product development projects in the chemical industry, Leung 2002, for software maintenance projects, and Griffin 2002, who suggested a general classification of projects).

Following Adler et al. (1995), we model the system as a stochastic *processing network*. The building blocks of the model are interdependent, identical resources that process project activities. At any given moment, each activity is receiving service from a resource; queuing up for access to a resource (in a *resource queue*); or waiting to join a predecessor activity that is being processed or delayed elsewhere (in a *synchronization queue*). The network model is stochastic (the duration of activities is governed by random variables) and dynamic (as time goes by, new projects randomly appear and existing projects are completed and leave the system). Randomness here captures unpredictable variability (both in the arrival pattern of new projects and in the required processing times) that is prevalent and significant in many multi-project environments. Such environments were also studied by Levy and Globerson (1997), who suggested a similar model to the one developed by Adler et al. (1995), and demonstrated its usefulness in different settings.

We start by assuming that each work-center is associated with a single resource type (e.g., mechanical work-center with mechanics, electrical work-center with electricians, etc.). Later, we extend the model to scenarios in which work-centers may consist of several resource types (see Sect. 5.5). Finally, we assume that specific allocations of resources to work-centers are not associated with any fixed costs. Thus, we may periodically change the workforce mix while keeping the total number of resource units fixed: for example, by enlarging the number of resource units in a specific resource group while reducing the number of resource units in another resource group.

### 2.2 The CONPIP model

The practice of beginning the work on an in-coming project immediately upon its arrival is equivalent to the *push* approach in production management. In contrast, a *pull* approach allows new projects to enter only when the system signals that it can accept them. We follow Anavi-Isakow and Golany (2003) who apply the pull principle in a dynamic, stochastic, multi-project system and show that it outperforms a push-based system (specifically, the average *total throughput time of a project*—from its arrival into the system until its departure—is lower).

The *COnstant Number of Projects In Process* (CONPIP) model, developed by Anavi-Isakow and Golany (2003), limits the number of projects in the system to a fixed number (*NPIP*). If a project arrives when the number of projects in the system equals *NPIP*, it will wait in a backlog queue. When a project is completed, the first project in the backlog queue enters the system. Alternatively, if there are fewer than *NPIP* projects in the system, the backlog queue is empty and incoming projects are immediately admitted into the system. Once started, a project is broken down to its individual activities. In order to be processed, an activity has to be ready (i.e., all its predecessor activities have been completed), and the relevant resources have to be available. If a project waits in a backlog queue, its total throughput time is composed of two components: (1) the time that was spent in the backlog queue, and (2) the time from leaving the backlog queue until the project's completion and departure from the system. For projects that do not wait in the backlog

queue, the value of the first component is zero. Thus, the total throughput time of a project is dependent upon the status of the system it meets upon its arrival.

Activities that experience long delays in resource queues are penalized by increasing their processing times through monotonically increasing functions. These functions reflect real life corrective or updating actions that are needed before the delayed activity is performed (for example, in aircraft maintenance projects, engineers usually need to update the data of a delayed activity or to repeat some inspections, etc.).

## 3 A heuristic approach for resource allocation

Rough-cut algorithms are commonly used to support decisions in stochastic, complex environments. Such algorithms may provide quick approximations when an exact solution is either too costly or computationally intractable. Hopp and Spearman (1996), give an example of a rough-cut capacity model for a production planning. Martien et al. (2002), use a rough-cut algorithm for resource allocation in a multi-project organization. To the best of our knowledge, no optimal solution method was developed to tackle this problem in multi-project organizations.

Hence, before presenting our CE-based methodology, we construct a heuristic algorithm whose solution would serve us as a benchmark to the resource allocation problem. Our heuristic combines a rough-cut approach and a local search, thus it finds better solutions than those that can be found by merely applying the rough-cut approach. In Sect. 5 we compare the performance of the heuristic with the proposed CE-based approach (that is developed in Sect. 4).

Consider the following resource allocation problem:

A stochastic and dynamic multi-project system is composed of $I$ work-centers (each may process different activity types) and a limited number $J$ of identical resource units. New projects are arriving randomly with a known average arrival rate. The long-term output rate should be equal to the input rate (call it *throughput*). The objective is to minimize the steady-state, mean projects' throughput time $S(x)$, where $x$ is a vector representing the allocation $x_1, \ldots, x_I$ that must satisfy the following constraints: $x_1 + \cdots + x_I = J$ and $x_i \in \{1, 2, 3, \ldots\}$ $\forall i = 1, \ldots, I$.

In this formulation, we need to determine the allocation of *all* the resources. In Sect. 5 we shall consider also the case where we have some flexibility to alter $J$. Note that for a fixed $J$, there are $\binom{J-1}{I-1}$ possible resource allocation combinations.

The heuristic algorithm relies on the utilization data to approximate the resource allocation. The procedure is summarized below:

*Algorithm RC*

1. Assume that a single unit of resource is assigned to each work-center $i$ and calculate its average utilization, $UT_i \in \Re^+$, according to the formula $UT_i = \lambda_i/\mu_i$, where $\lambda_i$ is the average arrival rate of activities to work-center $i$ and $\mu_i$ is the average service rate of a single resource-unit in work-center $i$.
2. Determine $\tilde{L}_i \in \Re^+$, the allocation of resource-units to work-center $i$, so as to balance the utilization load among the work centers. That is, $\tilde{L}_i = J \cdot (UT_i / \sum_{i=1}^{I} UT_i)$, $\forall i = 1, \ldots, I$.
3. Run a local search for a feasible (integer) resource allocation as follows:
   (a) Generate all[1] possible resource allocations by rounding all $\tilde{L}_i$'s except one, say $\tilde{L}_k$, to $x_i \in \{\lfloor \tilde{L}_i \rfloor, \lceil \tilde{L}_i \rceil\}$ and then determine the allocation to the $k$th work center so that the resource constraint $\sum_{i=1}^{I} x_i = J$ is satisfied.
   (b) Simulate the operations of the system, and choose the resource allocation solution that yields the minimal expected throughput time.

To demonstrate the algorithm, consider the *push*, multi-project system in Fig. 1. The network represents a single project type, arriving randomly according to a Poisson process with an arrival rate of $\lambda = 1/3.5$ projects per time unit. The *First Come First Served* (FCFS) priority rule is used to manage all queues. The resources can work in parallel at a processing rate of $\mu_i$ (processing times are taken from the exponential distribution $\sim \exp(\mu_i)$) for each resource engaged in activity type $i$. The start and finish activities are milestones—they have neither duration nor a resource requirement. Processing time parameters are given in Table 1. The objective is to find the best resource allocation for $J = 9$.

Assuming that a single resource-unit is assigned to each work-center, the first stage of the algorithm results with

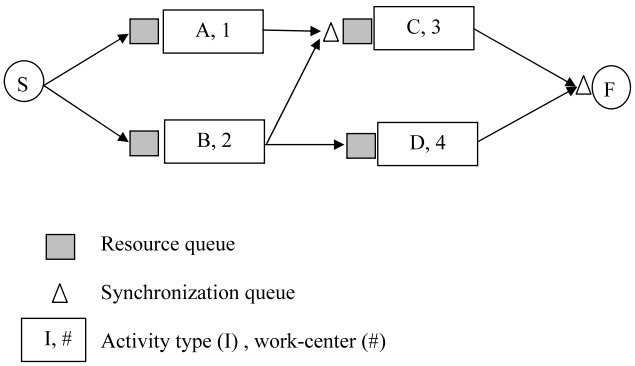


**Fig. 1** Stochastic processing network representing a multi-project system

[1] The number of possibilities is bounded by $I \cdot 2^{I-1}$.

**Table 1** Multi-project system characteristics - processing times and time distributions

| | Work-center | Time distribution |
|---|---|---|
| Activity A | 1 | $\sim \exp(\mu_A = 1/6)$ |
| Activity B | 2 | $\sim \exp(\mu_B = 1/5)$ |
| Activity C | 3 | $\sim \exp(\mu_C = 1/4)$ |
| Activity D | 4 | $\sim \exp(\mu_D = 1/3)$ |

the utilization vector: $(UT_1, UT_2, UT_3, UT_4) = (1.71, 1.43, 1.14, 0.86)$. We notice that a minimum of 2 resource-units should be allocated to work-centers 1–3 to prevent the system from "exploding". In the second stage we allocate the 9 resources proportionally to the load, resulting in the resource allocation vector (3.0, 2.5, 2.0, 1.5). The third stage needs to draw a candidate for rounding from work-centers 2 and 4—say work-center 2. We round its resource allocation value to a nearby integer (2 or 3 in our case)—suppose we round the value to 3. The possible number of resource units to allocate to work-center 4 is now reduced to 1 in order to satisfy the overall resource limit. Thus, we end up with the following resource allocation (3, 3, 2, 1). The second possibility is to round the number of resource units assigned to work-center 2 to 2. Then we get a resource allocation of (3, 2, 2, 2). Simulating the model with these allocations, we find (3, 2, 2, 2) to yield the lower expected projects' throughput time.

## 4 The CE algorithm for resource allocation

We begin with a brief background on the fundamentals of CE and then adapt it to our problem. For a comprehensive review of the theory and application of CE, refer to Rubinstein and Kroese (2004), or to the on-line tutorial that is available on the CE homepage (http://iew3.technion.ac.il/CE/).

### 4.1 The CE methodology

The CE method is a versatile new technique for rare event simulation and combinatorial optimization. This heuristic method was proven capable of solving a large variety of estimation and optimization problems, especially NP-hard combinatorial, deterministic and stochastic (noisy) problems (Rubinstein and Kroese 2004 provide CE solutions for various problem classes such as: routing, partition, packing and scheduling). The CE is unique compared to most other heuristic techniques, as it defines a precise mathematical framework for deriving fast updating/learning rules based on advanced simulation theory.

To apply the CE method one must first translate the underlying optimization problem into an associated stochastic problem. Next, the CE algorithm is applied in 2 iterative phases: (1) generation of a sample of random data (trajectories, vectors, etc.) according to a specified random mechanism, and simultaneous calculation of the objective function; (2) updating the parameters of the random mechanism (on the basis of the data collected) in order to produce a "better" sample in the next iteration.

The method can be illustrated with the following discrete minimization problem. Let $\chi$ represent a finite set of states (for example, all the feasible resource allocation combinations), where $x \in \chi$ is a particular vector (for example, a vector of resource allocation values, one for each work-center) and $S$ is a real function on $\chi$ denoted as the objective function (for example, the average throughput time of a project in the system). Find the minimal $S$ and its corresponding state(s). For the sake of simplicity we assume that there is only one such state vector $x^*$ and formulate the problem as:

$$S(x^*) = \gamma^* = \min_{x \in \chi} S(x), \tag{1}$$

where $\gamma^*$ is the optimal objective function value.

The first step in implementing CE is to translate (1) into an associated stochastic problem. To do so, we define on $\chi$ a set of indicator functions $I_{\{S(x) \le \gamma\}}$ for various threshold values $\gamma \in \Re^+$ and a set of discrete *probability density functions* (pdfs) with a parameter (vector) $u$, $\{f(\cdot, u)\}$. Now we associate with (1) the following stochastic equation:

$$\begin{aligned} \ell(\gamma) &= \mathbf{P}_f\big(S(X) \le \gamma\big) = \sum_x I_{\{S(x) \le \gamma\}} f(x, u) \\ &= \mathbf{E}_f I_{\{S(X) \le \gamma\}}, \end{aligned} \tag{2}$$

where for a certain $u$, $\mathbf{P}_f$ is the probability for which a random state $X$ has a pdf $f(\cdot, u)$, and $\mathbf{E}_f$ represents the corresponding expectation. In order to demonstrate how (2) is linked to (1) we shall assume, for example, that $\gamma$ equals $\gamma^*$. In this case $\ell(\gamma^*) = f(x^*, u^*)$ which is usually a very small number, thus estimating it via crude Monte Carlo simulations is expected to be highly inefficient. Nevertheless, we can estimate $\ell(\gamma^*)$ using *Importance Sampling* (IS). In IS we take a random sample $X_1, \ldots, X_N$ from another pdf $g$. We can formulate (3) using the pdf $g$ as:

$$\ell(\gamma) = \mathbf{E}_g I_{\{S(X) \le \gamma\}} \frac{f(X, u)}{g(X)}. \tag{3}$$

An unbiased estimator for $\ell(\gamma)$ is (Rubinstein and Melamed 1998; Rubinstein 1997):

$$\hat{\ell}(\gamma) = \frac{1}{N} \sum_{n=1}^{N} I_{\{S(X_n) \le \gamma\}} \frac{f(X_n, u)}{g(X_n)}. \tag{4}$$

In general, any pdf $g$ can be assigned in (4). However, we would like to select $g$ so that its entire probability mass will

reside at, or near, the optimal state vector $x^*$. When $\gamma = \gamma^*$, the optimal density function for $g$ is the one for which all the density is in state $x^*$ and, therefore, the estimator's variance is zero. Similarly, if $\gamma$ is close to $\gamma^*$, it is likely that most of the probability density is given by the optimal pdf $g$ to state(s) close to $x^*$, and the objective's value is close to (1). The CE method was developed especially for fast estimation in order to find the optimal IS density (i.e., the optimal $g$). Of course, once we find the optimal $g$, we actually know what the optimal state is, and then it is trivial to find the optimal objective function in (1). Rubinstein (1997) and Rubinstein and Melamed (1998), show that the optimal density function for the change of measure from $f$ to $g$ is:

$$g^*(x) = \frac{I_{\{S(x)\le\gamma\}} f(x,u)}{\ell(\gamma^*)}. \tag{5}$$

However, $g^*(x)$ is difficult to find because it depends on the unknown value $\ell(\gamma^*)$. The idea is to select, instead of $g^*(x)$, a different pdf $f$ with parameter $\tilde{u}$ that is "optimal" in the sense that its "distance" from $g^*(x)$ is minimal. The CE method uses the *Kullback–Leibler distance* for this minimization. The Kullback–Leibler distance (Kapur and Kesavan 1992) is defined as:

$$\begin{aligned} K(g_1, g_2) &= \mathbf{E}_{g_1} \ln \frac{g_1(x)}{g_2(x)} \\ &= \sum_x g_1(x) \ln g_1(x) - \sum_x g_1(x) \ln g_2(x). \end{aligned} \tag{6}$$

In our case,

$$g_1(x) \equiv \phi \cdot I_{\{S(X)\le\gamma\}} \cdot f(x,u),$$

where $\phi$ is a normalization constant;

$$g_2(x) \equiv f(x,v),$$

where $v$ is the parameter of the pdf $g_2$.

Finally, by minimizing (6) we formulate the stochastic problem that is associated with (1):

$$v^* = \underset{v}{\operatorname{argmax}}\, \mathbf{E}_u I_{\{S(X)\le\gamma\}} \ln f(X,v), \tag{7}$$

where $\mathbf{E}_u$ denotes the expectation on $X$ under pdf $f(X,u)$.

For random discrete vectors $X$ we can estimate the components of a $v^*$ vector by:

$$\hat{v} = \underset{v}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} I_{\{S(X_n)\le\gamma\}} \ln f(X_n, v), \tag{8}$$

where $X_n$ are generated from the pdf $f(\cdot,u)$. This estimator is practical only when $I_{\{S(X_n)\le\gamma\}} = 1$ for a sufficient number of samples. This means, for example, that when $\gamma$ is close to $\gamma^*$, $u$ should be such that the value of $\mathbf{P}_u(S(X) \le \gamma)$ is not too low. Consequently, on the one hand we wish to select $\gamma$ as close as possible to $\gamma^*$ and estimate $v^*$ (but then we know that in such a case $\mathbf{P}_u(S(X) \le \gamma)$ is usually very small). On the other hand, we wish to keep the value of $\mathbf{P}_u(S(X) \le \gamma)$ relatively high in order to find a reliable estimator in (8). To deal with these contradicting needs, CE takes an iterative approach while setting threshold levels $\hat{\gamma}_1, \hat{\gamma}_2, \ldots, \gamma_T$ and their corresponding vectors $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_T$ so that $\hat{\gamma}_T$ is close to the optimal $\gamma^*$, and $\hat{v}_T$ gives its probability density to the states with the best objective function values, for a constant $\hat{\gamma}_T$ (note that at the $t$-iteration $X_n$ are generated from the pdf $f(\cdot, v_{t-1})$).

Instead of directly updating the parameter $\hat{v}_{t-1}$ to $\hat{v}_t$, where $t$ denotes the iteration number, the standard procedure is to use a *smoothing* mechanism as follows:

$$\hat{v}_t = \alpha \hat{w}_t + (1-\alpha)\hat{v}_{t-1}, \tag{9}$$

where $\hat{w}_t$ represents the vector $\hat{v}$ that was calculated using (8) at iteration $t$. This smoothing procedure may prevent converging to a local optimum. Values of $0.7 \le \alpha \le 0.9$ have been found empirically to give the best results with (9).

For "noisy" (stochastic) problems $S(x)$ in problem (1) is replaced by the estimator $\hat{S}(x)$, $\hat{S}(x) = S(x) + \varsigma(x)$, where $\varsigma(x)$ is the noise factor. As stated in Law and Kelton (1991), in complex systems the common approach is to estimate expected performance through simulation experiments. In our case we shall estimate the expected performance through the steady state average of projects' throughput time after truncation of a warm-up period.

In conclusion, to apply the CE method to a specific problem the following steps are needed: (1) develop a suitable sampling algorithm, (2) develop the updating rules for the parameters through the minimization of the Kullback–Leibler distance, and (3) define the stopping rule. Having done that, CE has the ability to converge independently to a solution. It is important to note that Rubinstein and Kroese (2004), prove asymptotic convergence of CE and their numerical results for NP-Hard problems suggest complexity of order $O(n^2)$—see, e.g., ibid. Sect. 4.9.4, p. 168.

### 4.2 The main algorithm

We follow Rubinstein and Kroese (2004), who show that the components of the random discrete vectors, $X$, can be estimated at the $t > 0$ iteration according to,

$$\hat{v}_{ij}^t = \frac{\sum_{n=1}^{N} I_{\{\hat{S}(X_n \le \gamma_t)\}} I_{X_{n,i}=j}}{\sum_{n=1}^{N} I_{\{\hat{S}(X_n \le \gamma_t)\}}}, \tag{10}$$

where $\hat{v}_{ij}^t$ is the $t$-iteration's estimator for the probability that the optimal resource allocation value for work-center $i$ is $j$. The threshold levels $\gamma_t$ are heuristically determined

by arranging the sample performance in an increasing order: $\hat{S}_1 \leq \cdots \leq \hat{S}_{(N)}$ and fixing $\gamma_t = \hat{S}_{(\lceil \rho N \rceil)}$, where $\rho$ is a predetermined sample quantile, and $N$ is the sample size. For each iteration, the probability estimators, $\hat{v}_{ij}^t$'s, may be arranged into a $(I) \cdot (J - I + 1)$ matrix, $V^t$, where $I$ and $J$ denote the number of work-centers and the total number of resource units, respectively.

Equation (10) has a simple intuitive interpretation: to update the probability we simply take the fraction of the times that the resource allocation value for work-center $i$ is $j$, taking into account only the cases that have an objective value less than or equal to $\gamma$.

The proposed CE algorithm stops when the following two conditions are satisfied:

$$\varphi_i^t(j) = \varphi_i^{t-1}(j) = \cdots = \varphi_i^{t-c}(j), \qquad \max_j \hat{v}_{ij}^t \geq p', \quad \forall i = 1, \ldots, I, \tag{11}$$

where $c$ is some integer, $\varphi_i^t(j)$ denotes the index of the maximal element in the $i$th row of matrix $V^t$ (that corresponds to work-center $i$), and $p'$ is a probability criterion, for which the purpose is to assure the convergence to a single combination of resource allocation. The sample size $N$, needed to estimate $I \cdot (J - I + 1)$ components of matrices $V^t$, is of the order $I \cdot (J - I + 1)$ replications (Rubinstein and Kroese 2004).

Based on the above discussion, the main CE algorithm for the Resource Allocation (RA) problem is:

*Algorithm RA*

0. Determine CE parameters: the sample size $N$, $\rho$ the sample quantile of performance, the smoothing parameter and the stopping rule parameters.
1. Generate an initial matrix of probabilities $V^0$ from a discrete uniform distribution (in the absence of preliminary information). This matrix represents the initial distribution for all the work-centers. Set $t = 1$ (level counter).
2. Call Algorithm SG (see Sect. 4.3) to generate a sample of independent, random vectors for resource allocation combinations $X_1, \ldots, X_N$ from the pdf $f(\cdot, V^{t-1})$. Use simulation to estimate the mean projects' throughput time $\{\hat{S}(X_n), \forall n = 1, \ldots, N\}$ and arrange them in an increasing order: $\hat{S}_{(1)} \leq \ldots \leq \hat{S}_N$. Let $\gamma_t$ be the $\rho$ sample quantile of the throughput times: $\gamma_t = \hat{S}_{(\lceil \rho N \rceil)}$.
3. Use the same sample to update $V^t$ via (9) and (10).
4. Stop if convergence is reached (let $T$ denote the final iteration). Otherwise, increase $t$ by 1 and repeat from step 2.

### 4.3 Sample generation algorithm

A simple and efficient method to generate a random resource allocation is given by the following algorithm. The probability vectors for drawing the resource allocations are obtained from the main algorithm.

*Algorithm SG (sample generation)*

0. Randomly generate priorities for the allocation of resources to all $I$ work-centers. The priorities determine the sequence of the resource allocation (that is: resources are allocated first to work-centers with higher priorities). Let $X_{(i)}$ denote the number of resource units allocated to a work-center with priority $i$ and $J$ as the total number of available resource units.
1. Generate $X_{(1)}$ for the work-center with the highest priority, from a probability vector $(v_{11}, \ldots, v_{1(J-(I-1))})$. For any probability $v_{gy}$—$g$ and $y$ represent the work-center and the number of allocated resource units, respectively.
2. Given $X_{(1)} = r$, generate $X_{(2)}$ from the truncated distribution $(v_{21}, \ldots, v_{2(J-(I-2)-r)}, 0, \ldots, 0)/(v_{21} + \cdots + v_{2(J-(I-2)-r)})$.
3. Given $X_{(1)} + X_{(2)} = k$, generate $X_{(3)}$ from the truncated distribution $(v_{31}, \ldots, v_{3(J-(I-3)-k)}, 0, \ldots, 0)/(v_{31} + \cdots + v_{3(J-(I-3)-k)})$.
4. Continue according to the above procedure for all work-centers. The result is a vector $X$ that represents the resource allocation for all the work-centers.

The algorithm SG is repeated in every CE iteration. But the matrix $V^t$ is different each iteration (it is updated by the CE updating rules) until the algorithm converges to a single resource allocation.

## 5 The experiments

We conducted a series of experiments to test the proposed CE-based approach in stochastic, dynamic multi-project systems. In these tests the solutions of the Algorithm RC served as a benchmark. The CPU-time to find these benchmark solutions was relatively short—a few seconds for each solution. Thus, computational time to find RC solutions was not an issue. For the experiments, we used multi-project systems that appeared in previous research (Cohen et al. 2004, 2005), so their characteristics and performance measures are known.

### 5.1 Description of the experiments

We conducted 3 sets of experiments. In the first two sets we considered a resource-constrained organization, where $J$ resource units have to be allocated to $I$ work-centers. We assumed that assigning a specific allocation of resources to work-centers is not associated with a specific cost (e.g., when workforce is hired through a manpower agency that charges only variable costs and changes in the workforce

mix can be done without paying the agency any fixed cost). In the third set projects' activities were processed, simultaneously, by up to two different resource types. The overall amount of each resource type was given. Thus, the resource allocation problem amounted to allocating all the available resource units, of both resource types, to the work-centers in order to minimize the expected projects' throughput time. In the first set of experiments we determined the resource allocation for a stochastic, dynamic system that processed a single project type. In the second set of experiments the system processed 3 different project types and was managed by CONPIP. In fact, it was the same system introduced in Fig. 4 of Cohen et al. (2005). This experiment demonstrated that our CE algorithm improved a well-managed system by allocating its resources in a better way. The purpose of the third set of experiments was to demonstrate that our approach could easily be extended to multiple resource types' scenarios.

### 5.2 Discussion of the experimental environment

For each of the experiments we ran Algorithm RA: first, we determined the CE parameters and then we generated an initial uniform matrix of probabilities, from which we drew our sample (according to Algorithm SG). Our experimental tool to estimate the objective function value (Step 2 of Algorithm RA) was simulation, written in Visual Basic. Each simulation run started with a warm-up period (processing of several thousands of projects). This transient phase was discarded from the analysis. The system was then simulated for a predefined time interval that was large enough to process an additional several thousands of projects. After the CE algorithm converged to a solution (a certain resource allocation), we estimated the objective function value using 100 replications.

For each of the experiments, we generated 10 independent solutions of Algorithm RA. Then we calculated an *alpha-level* (we used alpha = 0.05) *confidence interval* (CI), given by:

$$\bar{S}(x^*) \pm t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{\sigma}}{\sqrt{n}},$$

where $\pm t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{\sigma}}{\sqrt{n}}$ is the *confidence interval variation* (CIV), $\bar{S}(x^*)$ is the estimator for the performance function value (based on the average of 10 independent solutions) and $\hat{\sigma}^2$ is the estimator for the variance,

$$\hat{\sigma}^2 = \frac{\sum_{j=1}^{n} (\hat{S}_j(x) - \hat{S}(x))^2}{n-1}.$$

The other performance measures that we used to evaluate the model performance were: $\bar{T}$ the mean number of iterations until convergence, the coefficient of variation in percent ($CV = \frac{\hat{\sigma}}{\bar{S}} \cdot 100$), the worst and best values for the absolute relative error, $|\frac{\hat{S}_j(x) - \bar{S}(x)}{\bar{S}(x)} \times 100|$, amongst the 10 independent solutions in percentages ($\varepsilon_*$ and $\varepsilon^*$, respectively) and the mean CPU time required for convergence (in seconds). All the experiments were conducted on a PC platform with an Intel, Pentium 4, 1.8 GHz processor.

### 5.3 Single project type

We applied the proposed model to the (push) stochastic processing network that was earlier described in Fig. 1 and Table 1 and for which we applied Algorithm RC. This network is identical to the one presented in Cohen et al. (2004)—but here it serves the purpose of demonstrating the resource allocation model.

#### *5.3.1 The experimental design*

The set of parameters used in the experiments:

The available number of resource units $J$ was set to 9. That is:

$$\sum_{i=1}^{4} X_{(i)} = 9 \quad \text{and} \quad X_{(i)} \in \{1, 2, \ldots, 6\} \quad \forall i = 1, \ldots, 4.$$

We selected $c = 3$ and $p' = 0.99$ for the stopping rule. We took $\rho = 0.1$ (in Rubinstein and Kroese 2004, an interval of $0.01 \leq \rho \leq 0.2$ is suggested for a buffer allocation problem in a stochastic and dynamic network) and the value of the smoothing parameter, $\alpha$ was set to 0.8 (which is within the recommended interval, $0.7 \leq \alpha \leq 0.9$). We fixed the sample size at $N = 120$ (using the formula $(I) \cdot (J - I + 1)$ with a factor of 5 to represent an average order of magnitude). We generated 10 independent solutions.

#### *5.3.2 Results*

All the solutions converged to $x^* = (3, 2, 2, 2)$. Convergence was achieved in 4 to 5 iterations where the mean number of iterations was 4.3. The mean CPU time until convergence was 939 seconds. Figure 2 shows the dynamics of convergence for a typical solution (see Table 3 for the summary of results).

Next, we decreased the sample size $N$ to 48 using a factor of 2 to represent an average order of magnitude. The same solution was reached with 4.6 as the mean number of iterations before convergence (instead of 4.3) but with a mean CPU time of 375 seconds. Namely, a 60% decrease of CPU time, with approximately the same number of iterations.

A system, designed by the proposed model, exhibits a much better performance compared to an identical system managed without the resource allocation model. For example, in the "No Control" case that appeared in Cohen et al.

$$V^0 = \begin{pmatrix} 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \end{pmatrix}$$

$$V^1 = \begin{pmatrix} 0.03 & 0.30 & 0.57 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.30 & 0.43 & 0.17 & 0.03 & 0.03 \\ 0.03 & 0.63 & 0.23 & 0.03 & 0.03 & 0.03 \\ 0.63 & 0.23 & 0.03 & 0.03 & 0.03 & 0.03 \end{pmatrix}$$

$$V^2 = \begin{pmatrix} 0.01 & 0.06 & 0.91 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.85 & 0.09 & 0.03 & 0.01 & 0.01 \\ 0.01 & 0.93 & 0.05 & 0.01 & 0.01 & 0.01 \\ 0.13 & 0.85 & 0.01 & 0.01 & 0.01 & 0.01 \end{pmatrix}$$

$$\vdots$$

$$V^5 = \begin{pmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

**Fig. 2** Dynamics of matrices $V^t$ for a system with a single project type

(2004), an arbitrary resource allocation of $(3, 2, 3, 1)$ was set. The expected projects' throughput time in the model we propose here is smaller than the original one by 39% (19.93 vs. 32.44). This was achieved without changing the throughput rate. We note also that in this case the resource allocation solution found by Algorithm RA is equal to the one that was found by Algorithm RC. However, this is not always the case, as we discuss later on. In Sect. 5.4 we elaborate on the heuristic, rough-cut based approach and compare it to the CE algorithm.

We conducted a sensitivity analysis to investigate the effect of relaxing the overall resource constraint on the performance. We changed $J$ values in the interval $\{7, 8, \ldots, 15\}$ (below 7 resource units the system "explodes") and for each value we solved the resource allocation problem using Algorithm RA. The results, which are presented in Fig. 3, construct an efficiency frontier of the expected throughput time and the overall number of resource units. We notice that the performance improvement decreases with larger values of $J$, so that even without exact knowledge on the costs that are involved, we see that in our example there is not much sense in increasing $J$ above 10 or 11 resource-units. Using simulation, we estimated the expected projects' throughput time for a system with an unlimited number of resource units as 13.04—which may serve also as a lower bound on the performance of the resource constrained systems. For a given $J$, management needs to consider if it is worthwhile to change the number of resource units. Furthermore, if both the economic cost of an additional resource unit and the benefit

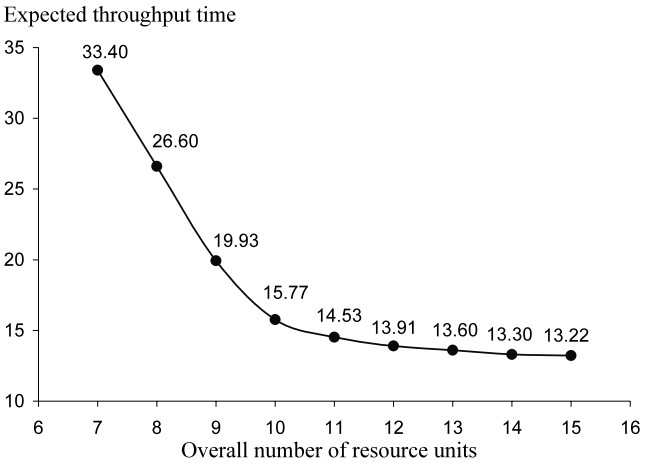


**Fig. 3** Expected projects' throughput time vs. overall number of resource units that were allocated by Algorithm RA

from shortening the expected projects' throughput time by a time unit are known, then management can decide upon the optimal number of resource units $J$. We do not pursue this subject further as it is not in the main focus of the paper and worth an independent research.

### 5.4 Three project types in a CONPIP managed system

We applied Algorithm RA to the stochastic network described in Fig. 4. The network represents 3 project types being processed by 4 work-centers. The network characteristics are identical to the network presented in Cohen et al. (2005), but with the use of the resource allocation model. Each project type arrives randomly according to a Poisson process with mean inter-arrival rates, $\lambda_1 = 1/5.5$, $\lambda_2 = 1/9.2$ and $\lambda_3 = 1/13.8$ projects per time unit. The processing time parameters are identical to the parameters used in the single project case. Following Anavi-Isakow and Golany (2003) long waiting times for resources are penalized with a two-step penalty function. For waiting times longer than 10 times the average processing time, the penalty is 0.5 times the average processing time; and for waiting times longer than 15 times the average processing time, the penalty is 0.8 times the average processing time. The *NPIP* combination was set to the combination that was found in Cohen et al. (2005), that is: to the maximum allowed number of 6 projects of type I, 4 projects of type II and 3 projects of type III.

#### *5.4.1 The experimental design*

We generated 10 independent solutions under the following CE parameters: $c = 3$ and $p' = 0.99$ for our stopping rule. We took $\rho = 0.1$. The value of the smoothing parameter $\alpha$ was set to 0.8. The sample size was fixed at $N = 120 = 5 \cdot (I) \cdot (J - I + 1) = 5 \cdot 4 \cdot 6$.

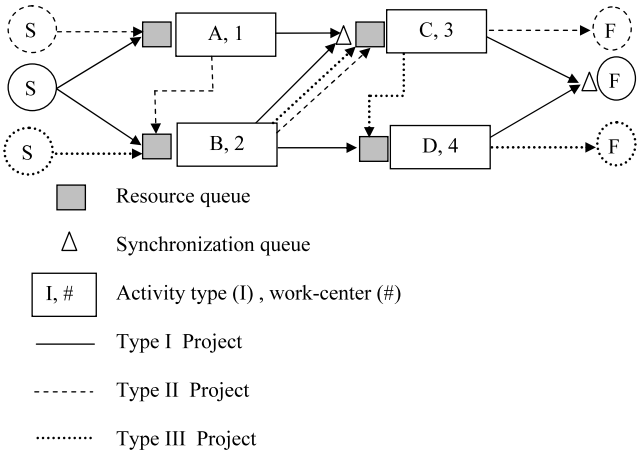


**Fig. 4** Stochastic processing network representing a multi-project system with 3 project types

### 5.4.2 Results

All the solutions converged to $x^* = (3, 3, 2, 1)$. Figure 5 shows the dynamics of the $V^t$ convergence for a typical solution. Convergence was achieved in 4 to 5 iterations with a mean of 4.2 iterations, and a mean CPU time of 1097 seconds.

We compared the performance of the system to the performance found in Cohen et al. (2005), where the resource allocation was arbitrary set to $(3, 2, 3, 1)$. As in the single project's system, the proposed model improves the performance significantly. The mean projects' throughput time was decreased by 51% (28.23 for the proposed system compared to 57.11 for the system that was optimized only with respect to the amount of work in process) while maintaining the same throughput rate. Based on *Little's Law* we may say that amount of work in process within the optimized system has been reduced. Thus, if management is willing to accept the previous resource utilization, it may then increase the throughput rate of projects resulting in higher profits.

We conducted a sensitivity analysis to investigate the stability of the solution. We increased the average processing time of the projects in work-center 3 by 5% increments. Work-center 3 was selected because all 3 project types are processed in it, thus increasing their processing times is expected to have the greatest impact on the solution stability. The results indicate that the chosen resource allocation remained unchanged up to a 25% increase in the average processing time. At a 25% increase (denoted as 1.25 on the $x$-axis of Fig. 6) the CE algorithm found that the (near-optimal) resource allocation changed to $x* = (3, 2, 3, 1)$, that is: a single resource unit was moved from work-center 2 to work-center 3. Figure 6 demonstrates that the resource allocation $(3, 2, 3, 1)$ is more stable than the resource allocation$(3, 3, 2, 1)$, with respect to an increase in the processing time of work-center 3.

$$V^0 = \begin{pmatrix} 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \end{pmatrix}$$

$$V^1 = \begin{pmatrix} 0.03 & 0.33 & 0.20 & 0.37 & 0.03 & 0.03 \\ 0.03 & 0.45 & 0.41 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.75 & 0.12 & 0.03 & 0.03 & 0.03 \\ 0.54 & 0.33 & 0.03 & 0.03 & 0.03 & 0.03 \end{pmatrix}$$

$$V^2 = \begin{pmatrix} 0.01 & 0.07 & 0.84 & 0.07 & 0.01 & 0.01 \\ 0.01 & 0.09 & 0.88 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.95 & 0.02 & 0.01 & 0.01 & 0.01 \\ 0.91 & 0.07 & 0.01 & 0.01 & 0.01 & 0.01 \end{pmatrix}$$

$$\vdots$$

$$V^5 = \begin{pmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

**Fig. 5** Dynamics of the matrices $V^t$ convergence for a system with 9 resources and 3 project types

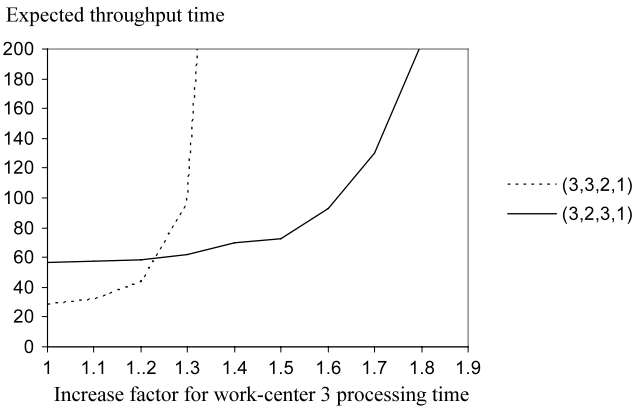


**Fig. 6** Expected projects' throughput time versus work-center 3 processing time for a system with 3 project types and 9 resources

In order to experiment with a larger system we doubled the following parameters of the system: the number of resource units $J$ was set to 18, the *NPIP* was set to $(12, 8, 6)$ and the throughput rate of projects was doubled. The sample size was set to $5 \cdot 4 \cdot (18 - 4 + 1) = 300$. The solutions converged to a resource allocation vector $x^* = (5, 5, 5, 3)$. Figure 7 shows the dynamics of the $V^t$ convergence for a typical solution. The mean number of iterations and CPU time before stopping were 6.8 and 7921 seconds, respectively. The mean project's throughput time for the modified system was 16.16–43% lower than the throughput time of the original system (28.23). This demonstrates the pooling

$$V^0 = \begin{pmatrix} 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \end{pmatrix}$$

$$V^1 = \begin{pmatrix} 0.02 & 0.02 & 0.18 & 0.11 & 0.18 & 0.09 & 0.09 & 0.14 & 0.04 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.35 & 0.14 & 0.14 & 0.09 & 0.09 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.25 & 0.23 & 0.16 & 0.09 & 0.04 & 0.04 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.25 & 0.30 & 0.09 & 0.09 & 0.07 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \end{pmatrix}$$

$$\vdots$$

$$V^3 = \begin{pmatrix} 0.00 & 0.00 & 0.02 & 0.01 & 0.90 & 0.03 & 0.01 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.11 & 0.37 & 0.45 & 0.03 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.02 & 0.48 & 0.44 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.08 & 0.83 & 0.06 & 0.01 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

$$\vdots$$

$$V^7 = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

**Fig. 7** Dynamics of the matrix $V^t$ convergence for a system with 3 project types and 18 resources

effect and the advantage of a large organization with many resource units ready to perform various projects' activities.

We investigated Algorithm RC with the modified system: the second stage of the algorithm resulted in the non-integer vector of a resource allocation $(5.4, 5.6, 4.5, 2.4)$ and the search ended with a resource allocation equal to the one found by the CE method, that is $(5, 5, 5, 3)$. Despite these results, using Algorithm RC may prove unfavorable in more complex systems, when penalties are different for projects in different work-centers. If, for example, the penalties on projects that are processed in work-center 3 are rather large (5 times larger than for the other work-centers) we reach a resource allocation $(5, 4, 7, 2)$ with an expected projects' throughput time of 19.45. Algorithm RC did not find this solution. All the resource allocations that were found by the RC algorithm caused the system to explode. The slowest explosion rate was found for a resource allocation $(5, 5, 6, 2)$. Figure 8 demonstrates the increase in the expected throughput rate with increasing simulation lengths for 3 vectors found by Algorithm RC versus the stability of the resource allocation that was found using CE.

This is expected, since the RC algorithm doesn't consider the delay times due to stochastic variations and their associated effects (for example, penalties, set-up times, etc.).

We conducted a sensitivity analysis to investigate the stability of the CE algorithm solution. We selected again work-center 3 and increased the average processing time

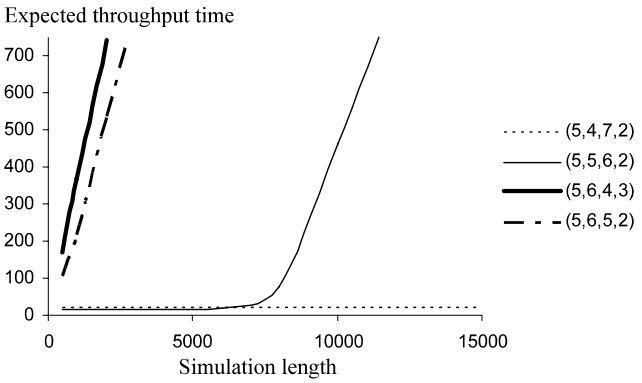


**Fig. 8** Expected projects' throughput time versus simulation length for a system with 3 project types and 18 resources

of the projects in work-center 3 by 5% increments. Results indicated that the resource allocation remained unchanged ($x^* = (5, 5, 5, 3)$) up to 35% increase in the average processing time. At a 35% increase, the CE algorithm found that the resource allocation has changed to $x^* = (5, 5, 6, 2)$. Figure 9 demonstrates that the resource allocation $(5, 5, 6, 2)$ is more stable with respect to an increase in the processing time of work-center 3. Comparing Fig. 9 to Fig. 6 we notice that the larger the system the more stable the performance, and the differences between the 2 best resource allocations are smaller. This however does not indicate that it is less important to find the optimal resource allocation for larger sys-

**Table 2** Multi-project system characteristics—processing times, activities' resource demands and time distributions

| | Work-center | Resource demands | | Time distribution |
|---|---|---|---|---|
| | | Type-1 | Type-2 | |
| Activity A | 1 | 2 | 2 | $\sim\exp(\mu_A = 1/6)$ |
| Activity B | 2 | 3 | 1 | $\sim\exp(\mu_B = 1/5)$ |
| Activity C | 3 | 1 | 0 | $\sim\exp(\mu_C = 1/4)$ |
| Activity D | 4 | 1 | 3 | $\sim\exp(\mu_D = 1/3)$ |

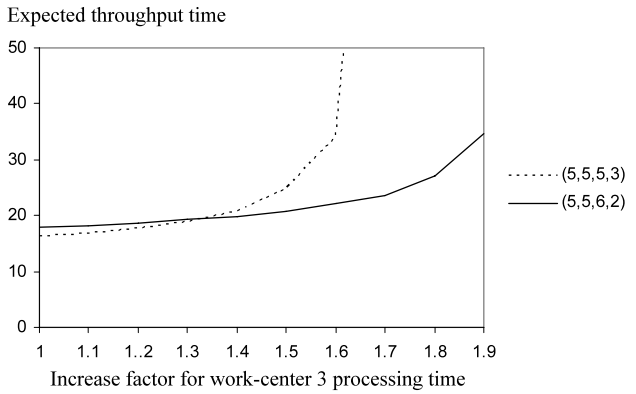


**Fig. 9** Expected projects' throughput time versus work-center 3's processing time for a system with 3 project types and 18 resources

tems, as the amount of the processed projects and the costs at stake are higher.

Further enlargement of the system, by doubling again the number of resource units to 36, setting the *NPIP* vector to (24, 16, 12) and doubling the throughput rate of projects, necessitates the use of a larger sample size of $5 \cdot 4 \cdot (36 - 4 + 1) = 660$. All 10 experiments converged to a resource allocation vector $x^* = (11, 11, 9, 5)$ within a mean number of 8.5 iterations and a mean CPU time of 23 545 seconds. The mean project's throughput time for the modified system was 14.04, 13% lower than the throughput time of the earlier system (16.16). The improvement in the performance that was caused by doubling the system size for the second time (13%), was significantly lower than the improvement in performance that was caused by doubling its size the first time (42%). The explanation is that as we are getting closer to the deterministic lower bound on the throughput time (11.6), we need to add more and more resources: in larger systems the delays in queues are expected to be smaller. Table 3 includes the summary of the results.

## 5.5 Three project types and 2 resource types

We applied Algorithm RA to the stochastic network described in Fig. 4 and Table 2. The network represents 3 project types being processed by 4 work-centers. The network characteristics are identical to the network presented in the previous section, but here the projects' activities are processed by 2 resource types whose capacities are 14 and 9 resource units for type-1 and type-2 resources, respectively. For the sake of simplicity, we modeled the system as a push system (otherwise, for a CONPIP managed system we would have needed to conduct a preliminary experiment using the approach suggested by Cohen et al. (2005) in order to determine the NPIP vector).

### 5.5.1 *The experimental design*

We generated 10 independent solutions under the following CE parameters: $c = 3$ and $p' = 0.99$ for our stopping rule. We took $\rho = 0.05$. The value of the smoothing parameter $\alpha$ was set to 0.7. It is important to note that in this case the CE iterations involve 2 probability matrices, one for each resource type. The sample size was fixed at $N = 560$.

### 5.5.2 *Results*

All the solutions converged to $x_1^* = (4, 6, 3, 1)$, and $x_2^* = (4, 2, 0, 3)$ for the first and second resource types, respectively. Figure 10 shows the dynamics of the $V^t$ convergence for a typical solution. A preliminary computation was conducted to eliminate infeasible solutions (e.g., the resource allocation to a work-center that processes activities that demand 2 units of type-1 resource must be greater than or equal to 2 units of that resource). This was done by fixing the corresponding probabilities in the initial probability matrices $V^0$ to zero.

Convergence was achieved in 7 to 16 iterations with a mean of 12 iterations, and a mean CPU time of 5316 seconds. The mean projects' throughput time was 35.49 for the proposed system (see Table 3 for the summary of the results).

$$V^0 = \begin{pmatrix} 0.00 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 \\ 0.00 & 0.00 & 0.14 & 0.14 & 0.14 & 0.14 & 0.14 & 0.14 & 0.14 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \end{pmatrix} \begin{pmatrix} 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.33 & 0.33 & 0.33 \end{pmatrix}$$

$$V^1 = \begin{pmatrix} 0.00 & 0.10 & 0.23 & 0.42 & 0.10 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.00 & 0.00 & 0.04 & 0.04 & 0.04 & 0.36 & 0.11 & 0.30 & 0.11 \\ 0.29 & 0.48 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.42 & 0.22 & 0.16 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{pmatrix} \begin{pmatrix} 0.00 & 0.33 & 0.39 & 0.20 & 0.08 \\ 0.06 & 0.38 & 0.25 & 0.25 & 0.06 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.61 & 0.23 & 0.16 \end{pmatrix}$$

$$V^2 = \begin{pmatrix} 0.00 & 0.03 & 0.07 & 0.76 & 0.09 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.00 & 0.00 & 0.01 & 0.01 & 0.01 & 0.74 & 0.10 & 0.09 & 0.03 \\ 0.09 & 0.76 & 0.09 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.25 & 0.64 & 0.05 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \end{pmatrix} \begin{pmatrix} 0.00 & 0.10 & 0.12 & 0.76 & 0.02 \\ 0.02 & 0.81 & 0.08 & 0.08 & 0.02 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.88 & 0.07 & 0.05 \end{pmatrix}$$

$$\vdots \qquad\qquad \vdots$$

$$V^7 = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.93 & 0.07 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.99 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.46 & 0.54 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.61 & 0.39 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix} \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \end{pmatrix}$$

$$\vdots \qquad\qquad \vdots$$

$$V^{11} = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.07 & 0.93 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.93 & 0.07 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix} \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \end{pmatrix}$$

$$\vdots \qquad\qquad \vdots$$

$$V^{13} = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.99 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.99 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix} \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \end{pmatrix}$$

**Fig. 10** Dynamics of the matrices $V^t$ convergence for a system with 3 project types and 2 resource types where there are: 14 units of type-1 resource, and 9 units of type-2 resource

**Table 3** Performance of Algorithm RA for: a system with a single project type (case a), CONPIP managed systems with 3 project types and 9 resource units (case b), 18 resource units (case c), 36 resource units (case d) and a push system with 3 project types and 2 resource types (case e)

| Case | $\bar{T}$ | $x^*$ | $\bar{S}(x^*)$ | CIV | CV | $\varepsilon_*$ | $\varepsilon^*$ | CPU |
|---|---|---|---|---|---|---|---|---|
| (a) Single project type | 4.3 | (3, 2, 2, 2) | 19.93 | ±0.04 | 0.29 | 0.43 | 0.02 | 939 |
| (b) 3 project types—9 resources | 4.2 | (3, 3, 2, 1) | 28.23 | ±0.35 | 1.73 | 2.99 | 0.44 | 1097 |
| (c) 3 project types—18 resources | 6.8 | (5, 5, 5, 3) | 16.16 | ±0.03 | 0.22 | 0.40 | 0.09 | 7921 |
| (d) 3 project types—36 resources | 8.5 | (11, 11, 9, 5) | 14.04 | ±0.00 | 0.05 | 0.11 | 0.04 | 23 545 |
| (e) 3 project types—2 resource types | 12 | (4, 6, 3, 1) | 35.49 | ±0.10 | 0.38 | 0.59 | 0.11 | 5316 |
| | | (4, 2, 0, 3) | | | | | | |

## 6 Conclusions

Little attention has been given to the problem of resource allocation in stochastic, multi-project organizations. Based on a management methodology, in which the system is controlled through keeping a constant number of projects concurrently in the system, a practical CE-based procedure for resource allocation was developed. The approach finds the resource allocation that heuristically minimizes the projects' average throughput time.

We showed that our enhanced rough-cut algorithm ("enhanced" relates to an additional local search for the optimal solution, within the neighborhood of the rough-cut solution) performs well for simple and small systems. However, its use may be unsuitable for larger and more complicated systems for the following two reasons: (1) it may promote choosing unfavorable resource allocations thus it is unsuitable for generic use. However, for the same examples CE did converge to the optimal (near-optimal) resource allocation; and (2) the extent of the local search may be impractical for large systems. Cross Entropy, on the other hand, is known to converge asymptotically, and numerical results for NP-Hard problems suggests complexity of order $O(n^2)$ (e.g., Rubinstein and Kroese 2004, p. 168).

We demonstrate via numerical examples the ability of CE based resource allocation to improve the performance of multi-project systems.

The suggested approach is flexible, allowing the allocation of all resources (for example, the decision about the resource allocations for an entirely new division); a partial allocation (for example, improving the performance of an existing organization where only some of the resources in the different work-centers are interchangeable); or determining the allocation of several resource types to the different work-centers (for example, when some or all the projects' activities are being processed by several resource types simultaneously).

Future research should focus on stochastic, dynamic multi-project systems in which a variety of resource types are used. A parallel computing approach may be applicable in such cases to speed up processing time for real life problems. The suggested model may also be applied to stochastic, *CONWIP* based environments (that the *CONPIP* methodology draws from), for example to determine the optimal product mix or the work-force allocation.

## 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.

Cohen, I., Mandelbaum, A., & Shtub, A. (2004). Multi-project scheduling and control: A process-based comparative study of the critical chain methodology and some alternatives. *Project Management Journal*, *35*(2), 39–50.

Cohen, I., Golany, B., & Shtub, A. (2005). Managing stochastic, finite-capacity, multi-project systems through the Cross Entropy methodology. *Annals of Operations Research*, *134*, 183–199.

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

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

Hopp, W. J., & Spearman, M. L. (1996). *Factory physics—foundations of manufacturing management*. Boston: Irwin.

Kapur, J. N., & Kesavan, H. K. (1992). *Entropy optimization principles with applications*. New York: Academic.

Kropp, D. H., & Carlson, R. C. (1984). A lot-sizing algorithm for reducing nervousness in MRP systems. *Management Science*, *30*, 240–244.

Kurtulus, I., & Davis, E. W. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, *28*(2), 161–172.

Law, A. M., & Kelton, W. D. (1991). *Simulation modeling and analysis*. New York: McGraw–Hill.

Lee, B., & Miler, J. (2004). Multi-project management in software engineering using simulation modeling. *Software Quality Journal*, *12*(1), 59–82.

Levy, N., & Globerson, S. (1997). Improving multiproject management by using a queuing theory approach. *Project Management Journal*, *28*(4), 40–46.

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

Martien, H. A., Hendricks, B. V. & Leon, H. K. (2002). Human resource allocation in a multiproject research and development environment. In: J. S. Pennypacker, L. D. Dye (Eds.), *Managing multiple projects* (pp. 249–262). New York: Marcel Dekker.

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operations Research*, *99*, 89–112.

Rubinstein, R. Y., & Kroese, D. P. (2004). *The cross entropy method: A unified approach to Monte Carlo simulation, randomized optimization and machine learning*. Berlin: Springer.

Rubinstein, R., & Melamed, B. (1998). *Modern simulation and modeling*. Berlin: Wiley.

Speranza, M. G., & Vercellis, C. (1993). Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, *64*, 312–325.