Ann Oper Res (2011) 186:429–442
DOI 10.1007/s10479-010-0733-y

Predetermined intervals for start times of activities in the stochastic project scheduling problem

Illana Bendavid · Boaz Golany

Published online: 20 March 2010

Abstract This paper proposes a new methodology to schedule activities in projects with stochastic activity durations. The main idea is to determine for each activity an *interval* in which the activity is allowed to start its processing. Deviations from these intervals result in penalty costs. We employ the Cross-Entropy methodology to set the intervals so as to minimize the sum of the expected penalty costs. The paper describes the implementation of the method, compares its results to other heuristic methods and provides some insights towards actual applications.

Keywords Stochastic project scheduling problem · Cross-entropy · Intervals · Gates · Flexible commitment

1 Introduction

Consider a project composed of activities whose durations are random variables with known distributions. The project has an external due-date and there is a tardiness penalty for each unit of time in which the project is delayed beyond that due-date. The activities of the project are processed by subcontractors, or alternatively they require that the resources needed for their processing must be booked well in advance to ensure that they are ready to carry out their tasks at predetermined times. The project manager (PM) seeks to determine for each activity an interval (g^1, g^2) in which it will be allowed to start its processing. The determination of these intervals induces the following cost structure:

- The subcontractor (or the resource needed) for the activity commits to be ready to start the activity at time g^1 and remains ready to start it throughout the interval (actual start time t will depend upon the completion times of its predecessors). In order to obtain this commitment, the PM has to pay the subcontractor an "interval cost" which grows linearly

I. Bendavid (✉) · B. Golany
Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, Haifa 32000, Israel
e-mail: illana@tx.technion.ac.il

with $g^2 - g^1$. This cost compensates the subcontractor for its willingness to make his resources available throughout the interval (g^1, g^2) (even if it means that these resources will sit idle, waiting for the call to launch the activity throughout the interval). This cost may be determined exogenously (according to "market rates") or it could be negotiated with the subcontractor.

- If the activity is ready to be processed at time $t < g^1$ (i.e., before its lower bound), the PM incurs a penalty cost which increases linearly with $g^1 - t$. This penalty is equivalent to a "holding cost" in the classical "newsvendor" inventory model as it reflects the fact that the PM has invested money to execute the preceding activities sooner than needed. Instead, the PM could have invested that money in secured bills and collect interest. The "holding cost" may also reflect indirect costs that are incurred in cases where the products of preceding activities deteriorate if not used immediately after they are finished.
- If the activity is ready to be processed at time $t > g^2$ (i.e., after its upper bound), the PM incurs a penalty cost which increases linearly with $t - g^2$. This penalty is equivalent to a "shortage cost" in the newsvendor model. It reflects the fact that the subcontractor must be compensated for having to wait idle longer than the interval which was agreed upon. It can also reflect an indirect cost in cases where the resources which were booked for the activity deteriorate if not used within the specific time interval which was defined for the start time of the activity.
- If the project is completed after the due date, the PM will pay a tardiness penalty (per unit time) which is larger than any of the shortage costs associated with starting individual activities later than their respective g^2's.

Given the cost structure explained above, there is a clear trade-off between penalty (holding and shortage) costs and interval costs: on one hand, the PM wants to enlarge the intervals of the activities since this will lead to lower penalty costs but, on the other hand, it will increase interval costs.

The goal of this paper is to develop a methodology that will determine the intervals described above so as to minimize the expected penalty and interval costs.

Scheduling of activities in projects with stochastic activity durations has been investigated extensively in the project management literature ever since the introduction of the Program Evaluation and Review Technique (PERT) (see Fazar 1959; Malcolm et al. 1959; Herroelen and Leus 2005) in the early 1960s. The models that were proposed to address this problem differ from each other by their assumptions on the characterization of the activities' durations, the presence of resource constraints and, most importantly, by their objective functions. Most of the existing models share the same objective—makespan minimization, which has been the most popular objective in the project scheduling domain.

In recent years, new methodologies have evolved that focus on another objective—maximizing the net present value (NPV) of projects. For the most part, these methods assume deterministic durations (see the reviews in Kimms 2001 and Herroelen et al. 1997). Maximization of NPV in stochastic settings was considered, to the best of our knowledge, only by Buss and Rosenblatt (1997) and by Sobel et al. (2009). An important observation that emerged from these articles was that when one maximizes NPV, starting activities as soon as possible (according to precedence constraints) is not always optimal.

In a recent paper (Bendavid and Golany 2009) we proposed a new approach to project scheduling with uncertain activity durations. This approach consists of determining for each activity a *gate*—a time before which the activity cannot begin. This approach was motivated by models proposed in Elmaghraby et al. (2000) and Elmaghraby (2001) in the context of scheduling manufacturing jobs on a single machine and models proposed by Trietsch (2005)

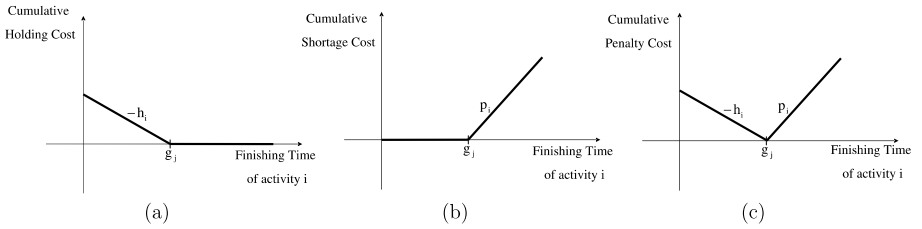

Fig. 1 Cumulative penalty cost according to the gate of activity j

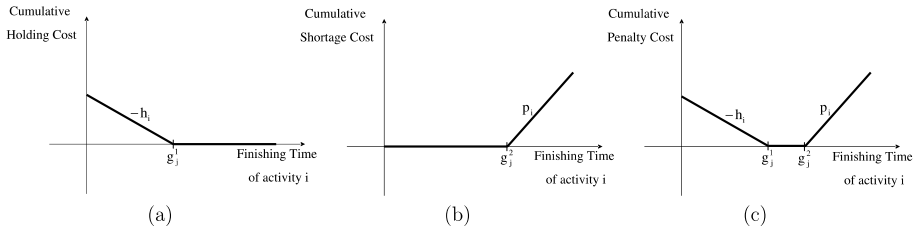

Fig. 2 Cumulative penalty cost according to the gate of activity j with the new cost structure

and (2006) in the context of scheduling project activities. In both environments, the idea was to determine for each operation a start time that would maximize total profit.

Analogous to the newsvendor model, the costs in Bendavid and Golany (2009) were defined as linear functions where the parameters h_i and p_i represent the holding and shortage costs per unit time, respectively, for activity i and g_i is a decision variable that sets the gate for activity i. When the activity durations are characterized as continuous random variables, the probability that an activity will finish exactly at the gate of one of its successors is null. Consequently, for each value of the gate and for each realization of the durations, we will incur some penalty (holding or shortage) cost, as is shown in Fig. 1 where activity j is a successor of activity i.

In the current paper, we employ the concept of Interval Goal Programming models (see Ignizio 1982 and Romero 1991) to extend the earlier work by enabling more flexible contracts with the subcontractors which would now be expected to start their respective activities within a certain time interval rather than exactly at a certain gate. Thus, instead of requiring a subcontractor to start processing activity j, $j \in S_i$ (where S_i is the set of successors of activity i), exactly at time g_j, we now require that activity j would begin within the interval: $[g_j^1, g_j^2]$. The contract will further stipulate that: (1) if activity i finishes its processing before time g_j^1 the PM will incur a holding cost. (2) If activity i finishes its processing after time g_j^2 the PM will incur a shortage cost. (3) Finally, if activity i finishes its processing within the time interval $[g_j^1, g_j^2]$, no penalty cost will be incurred. These three situations are shown in Fig. 2. In Figs. 1 and 2, we illustrate the case of continuous distribution. We note that when the activity durations are characterized as discrete random variables, the probability that an activity will finish exactly at the gate of one of its successors can be positive (but, in most realistic cases, small). Thus, for each value of the gate and for each realization of the durations, there will still be a large probability to incur some penalty (holding or shortage) cost. From now on, the rest of the analysis will deal with the case of discrete distributions.

Introducing flexibility into contracts with the subcontractors has recently become quite common in supply chain management. For example, there is a growing literature on contracts with "flexible commitments"—where retailers commit to purchase quantities within pre-defined intervals so as to smooth fluctuations of inventory replenishment orders between suppliers and retailers (see, for example, Ben-Tal et al. 2005; Bassok and Anupindi 2008).

Introducing time intervals aimed at smoothing the fluctuations caused by the randomness in activity durations is also not new—it stands at the core of the Critical Chain /Buffer Management (CC/BM) method (see Goldratt 1997 and Rand 2000). However, in CC/BM, these intervals are buffers added to the critical chain and the paths leading into it. In contrast, the intervals defined here are associated with the release time of individual activities rather than paths. Also, while the buffers in CC/BM are determined in an ad-hoc manner, here we propose a method that determines the intervals so as to optimize a certain objective function.

The rest of this paper is organized as follows. In the next section we describe the problem environment. In Sect. 3, we present the Cross-Entropy (CE) approach and its implementation to our particular problem. In Sect. 4, we present the comparison of the CE results to two heuristic methods and to the former approach of setting gates. Finally, in Sect. 5, we provide concluding remarks and some insights towards actual applications.

2 Problem description

First, we define the following notation:

n	the number of activities in the project
P_j	the set of predecessors of activity j
S_j	the set of successors of activity j
Y_j	the random variable that describes the duration of activity j
y_j	the realization of the random variable Y_j
F_j	the cumulative distribution function of the random variable Y_j
a_j	the minimal value of the random variable Y_j
b_j	the maximal value of the random variable Y_j
$\mathbf{G}^1 = (g_1^1, \ldots, g_n^1)$	the vector of the lower bounds where g_j^1 is the lower bound of the interval for activity j
$\mathbf{G}^2 = (g_1^2, \ldots, g_n^2)$	the vector of the upper bounds where g_j^2 is the upper bound of the interval for activity j
g_j^1	the lower bound of the interval of activity j (a decision variable)
g_j^2	the upper bound of the interval of activity j (a decision variable)
T_j	the random variable that describes the start time of activity j
t_j	the realization of the start time of activity j
h_j	the holding cost per time unit for activity j
p_j	the shortage cost per time unit for activity j
c_j	the interval cost per time unit for activity j
d	the due date for the entire project

We consider the following environment: a project is composed of n activities. We define P_j and S_j to be the set of predecessors and successors of activity j, respectively. We consider the case where the duration of each activity is a random variable. Each activity j has a duration Y_j, which is a discrete random variable with a cumulative distribution function F_j and realization y_j. It is assumed that the distribution of Y_j is bounded in the interval $\{a_j, a_j + 1, \ldots, b_j\}$ $(0 \le a_j \le b_j)$. The gates for each activity are the decision variables. Once they are

determined, the processing of an activity cannot start before the beginning of the interval but it may start later (if its predecessors are delayed). The recursion that determines the starting time t_j of activity j whose interval was $[g_j^1, g_j^2]$ and whose predecessors $l \in P_j$ have started at time t_l is given by:

$$t_j = \max_{l \in P_j} \{g_j^1, t_l + y_l\}$$

where we define $t_1 = g_1^1$.

We define h_j and p_j as the holding and shortage costs per unit time, respectively, for activity j. An ideal situation would be to finish processing all the predecessors of activity i within the interval $[g_j^1, g_j^2]$ in which case we will not incur any cost. The entire project has a due date d (generally imposed by external agents). The shortage penalty for the last activity is assumed to be significantly larger than the penalty of the other activities since it represents the penalty for not finishing the project on the due date.

The objective function is to minimize the expected penalty and interval cost by determining the lower and upper bounds of the intervals:

$$\min_{\mathbf{G}^1, \mathbf{G}^2} E\left[\sum_{j=1}^{n}\left(\sum_{s \in S_j}\left[h_j(g_s^1 - T_j - Y_j)^+ + p_j(T_j + Y_j - g_s^2)^+\right] + c_j(g_j^2 - g_j^1)\right)\right] \quad (1)$$

where we define: $x^+ = \max(0, x)$, and $g_{n+1}^1 = g_{n+1}^2 = d$.

3 Methodology

3.1 Description of the Cross Entropy (CE) method

The CE method was developed by Rubinstein and Kroese (2004). It is a general heuristic method for solving estimation and optimization problems. This method has been applied to solve a variety of problems and was proven to be quite effective for "hard" deterministic (combinatorial) and stochastic problems. For example, Alon et al. (2005) apply the CE method to allocate buffer spaces amongst machines in a serial production line so as to optimize some performance measure such as the steady-state throughput. The CE method has also been applied in the field of project management. Cohen et al. (2005) use the method to determine optimal loading of a stochastic and dynamic multi-project environment by keeping a constant number of projects performed concurrently in the system so as to minimize the average makespan of the projects. Bendavid and Golany (2009) use the CE method to set gates for activities as explained earlier.

For optimization problems, a general outline of the CE method requires the translation of the underlying optimization problem into a meaningful estimation problem called the *associated stochastic problem*. Thus, when solving a minimization problem, the estimation problem may be the expected number of times the objective function yields a value which is lower than a pre-specified threshold value. Next, the CE algorithm involves the following two phases: (1) generation of a sample of random data (demands, durations, trajectories, etc.) according to a specified random mechanism, and simultaneous calculation of the objective function values; (2) updating parameters of the random mechanism (on the basis of the data collected) in order to produce an improved sample in the next iteration, that is, a sample that will improve the value of the objective function.

In the next section, we explain how we apply the CE method to the specific scheduling problem addressed in this paper.

3.2 Employing CE method to the problem of setting intervals

The application of the CE method to our problem involves the following two phases: (1) generating the decision variables (the vectors of intervals) from an initial distribution and simultaneously calculating the total cost of the project for each vector of intervals generated; (2) updating the parameters of the generation distribution on the basis of the outcomes observed in the first phase.

To apply the CE method to our scheduling problem, we choose to use a continuous distribution function for the generation of the vector of intervals instead of a discrete distribution function. The reason is that the choice for the sample size N depends, according to Rubinstein and Kroese (2004), on the number of parameters we have to estimate. As explained in Bendavid and Golany (2009), if we use the Normal distribution function, we would have to estimate for each activity and for each one of the end-points of its interval just two parameters—the mean and the standard deviation. In contrast, if we use a discrete distribution function, we would have to estimate for each activity and for each one of the end-points of its interval the probability of all the values these variables can take.

We define the following additional notation:

N	the number of vectors of lower and upper bounds of the intervals generated at each iteration (the sample size in the CE algorithm)
N'	the number of vectors of activity durations generated for each vector of gate intervals
$\mathbf{G}_i^k = (g_{i1}^k, \ldots, g_{in}^k)$	the ith $\mathbf{G}^k$ vector generated ($k = 1, 2$)
$\hat{\mu}_t^k = (\hat{\mu}_{t,1}^k, \ldots, \hat{\mu}_{t,n}^k)$	the vector of means used in iteration $t+1$ where $\hat{\mu}_{t,j}^k$ is the mean of the normal distribution used to generate the bound g_j^k ($k = 1, 2$) of the interval for activity j in iteration $t+1$
$\hat{\sigma}_t^k = (\hat{\sigma}_{t,1}^k, \ldots, \hat{\sigma}_{t,n}^k)$	the vector of standard variations used in iteration $t+1$ where $\hat{\sigma}_{t,j}^k$ is the standard deviation of the normal distribution used to generate the bound g_j^k ($k = 1, 2$) of the interval for activity j in iteration $t+1$
ρ	the percentage of best results we want to use in each iteration
$C(\mathbf{G}_i^1, \mathbf{G}_i^2)$	the performance of the ith vector of intervals generated
$\mathcal{N}^{\text{elite}}$	the set of indexes of the $\lceil \rho N \rceil$ vectors that give the best performances among the N vectors generated
C_t^*	the best performance (lower cost) obtained in iteration t
N_{LIM}	the maximum number of variance injections to perform in the algorithm

The adaptation of the CE procedure to our problem, using a continuous density function is given below:

Algorithm 3.2: CE algorithm for setting intervals

Step 1 Set $t = 1$, choose $\hat{\mu}_0^1 = (\hat{\mu}_{0,1}^1, \ldots, \hat{\mu}_{0,n}^1)$, $\hat{\mu}_0^2 = (\hat{\mu}_{0,1}^2, \ldots, \hat{\mu}_{0,n}^2)$, $(\hat{\sigma}_0^1)^2 = ((\hat{\sigma}_{0,1}^1)^2, \ldots, (\hat{\sigma}_{0,n}^1)^2)$ and $(\hat{\sigma}_0^2)^2 = ((\hat{\sigma}_{0,1}^2)^2, \ldots, (\hat{\sigma}_{0,n}^2)^2)$ where $\hat{\mu}_{t,j}^k$ is the mean of the normal distribution in iteration $t+1$ used for generating the value g_j^k and $\hat{\sigma}_{t,j}^k$ is the standard deviation of the normal distribution used in iteration $t+1$ used for generating the value g_j^k, $k = 1, 2$.

Step 2 Generate a random sample $\mathbf{G}_1^1, \ldots, \mathbf{G}_N^1$ from the Normal distribution $N(\hat{\mu}_{t-1}^1, (\hat{\sigma}_{t-1}^1)^2)$, a random sample $\mathbf{G}_1^2, \ldots, \mathbf{G}_N^2$ from the Normal distribution

$N(\hat{\mu}_{t-1}^2, (\hat{\sigma}_{t-1}^2)^2)$. For each vector i of intervals generated, check if: $\mathbf{G}_i^1 \leq \mathbf{G}_i^2$. Calculate the performance (the total cost of the project) for each vector i of intervals generated—$C(\mathbf{G}_i^1, \mathbf{G}_i^2)$ in the following way: generate N' vectors of activity durations, each activity j according to the probability function of Y_j (these vectors are generated once and used for all the vectors of gates in every iteration). Calculate for each realization of durations the cost of the project. For a specific realization of activity durations $y = (y_1, \ldots, y_n)$, the cost of the project can be calculated using the following equation that derives from the objective function (1):

$$\sum_{j=1}^{n}\left[\sum_{s\in S_j}\left[h_j(g_s^1 - t_j - y_j)^+ + p_j(t_j + y_j - g_s^2)^+\right] + c_j(g_j^2 - g_j^1)\right] \tag{2}$$

where $x^+ = \max(0, x)$, t_j is the start time of activity j as defined above, $t_1 = g_1^1$, $t_j = \max_{l\in P_j}\{g_j^1, t_l + y_l\}$ and $g_{n+1}^1 = g_{n+1}^2 = d$. Finally, we calculate the average of these N' costs. This average is the estimator of the expected cost of the project for the specific vector of intervals $(\mathbf{G}_i^1, \mathbf{G}_i^2)$. Since we use an estimation of the cost function, Algorithm 3.2 implements in fact a noisy CE optimization procedure. We now have to order these performances from largest to smallest. Let $\mathcal{N}^{\text{elite}}$ be the set of indices of the $\lceil \rho N \rceil$ vectors that give the best performances among the N vectors generated.

Step 3 Update: for all $j = 1, \ldots, n$, $k = 1, 2$

$$\hat{\mu}_{tj}^k := \sum_{i\in\mathcal{N}^{\text{elite}}} g_{ij}^k / |\mathcal{N}^{\text{elite}}| \tag{3}$$

and

$$(\hat{\sigma}_{tj}^k)^2 := \sum_{i\in\mathcal{N}^{\text{elite}}} (g_{ij}^k - \hat{\mu}_{tj}^k)^2 / |\mathcal{N}^{\text{elite}}|. \tag{4}$$

Step 4 Smooth:

$$\begin{aligned} \hat{\mu}_t^k &:= \alpha\hat{\mu}_t^k + (1-\alpha)\hat{\mu}_{t-1}^k, \\ (\hat{\sigma}_t^k)^2 &:= \alpha(\hat{\sigma}_t^k)^2 + (1-\alpha)(\hat{\sigma}_{t-1}^k)^2. \end{aligned} \tag{5}$$

If $\max_{j=1,\ldots,n} \hat{\sigma}_{tj}^2 \leq \varepsilon$ ($\varepsilon = 0.01$), perform a "variance injection" according to:

$$\hat{\sigma}_t^2 := \hat{\sigma}_t^2 + |C_t^* - C_{t-1}^*|K \tag{6}$$

where the injection parameter K was set to 2.

Step 5 If the stopping criterion is met, that is, the number of "variance injections" performed using (6) exceeds a specific limit N_{LIM} (we use $N_{\text{LIM}} = 5$), then stop; otherwise set $t = t + 1$ and reiterate from Step 2.

In the first step, we choose the initial mean and variance. In our case, we chose $\hat{\mu}_0^1$ to be the vector of the early start times based on minimal processing times and $\hat{\mu}_0^2$ to be the vector of the early start times based on maximal processing times. For the initial variance, we chose: $(\hat{\sigma}_0^1)^2 = (\hat{\sigma}_0^2)^2 = (\frac{d^2}{9}, \ldots, \frac{d^2}{9})$ (that is, $\hat{\sigma}_{0,j}^k = d/3$ for all $j = 1, \ldots, n$). This initial value for the variance ensures that when the procedure starts it is possible to generate values for each activity from all the interval $[0, d]$. Indeed, in a Normal distribution, 99.7% of the values lie within three standard deviations of either side of the mean. With our choice of $\hat{\sigma}_{0,j}^k = d/3$

for all $j = 1, \ldots, n$, 99.7% of the values generated will be between the mean plus d and the mean minus d.

In the second step, we generate two samples from the Normal distributions with the initial means and standard deviations and calculate the cost of the project for each vector of intervals generated in the following way: we generate N' vectors of activity durations, each activity j according to the probability function of Y_j, then calculate for each realization of durations the cost of the project according to (2). Finally, we calculate the average of these N' costs. This average is the estimator of the expected cost of the project for a specific vector of intervals. We then choose the $\lceil \rho N \rceil$ vectors that give the best performances and keep the indexes of these vectors in the set $\mathcal{N}^{\text{elite}}$, where the cardinality of the set $\mathcal{N}^{\text{elite}}$ is equal to $\lceil \rho N \rceil$: $|\mathcal{N}^{\text{elite}}| = \lceil \rho N \rceil$. For example, for $\rho = 10\%$ and for $N = 1{,}000$, the set $\mathcal{N}^{\text{elite}}$ will be composed of indexes of the 100 vectors of intervals that gave lower costs among the $N = 1{,}000$ vectors generated.

In the next step we update for each activity four parameters: its mean and standard deviation for the two values of the interval. In the next iteration, the mean $\hat{\mu}_{tj}^k$ of the Normal distribution will be the average of all the k-values that gave the best performances (3) and the standard deviation $\hat{\sigma}_{tj}^k$ will be the standard deviation of all the k-values that gave the best performances (4).

In the fourth step, we use a smoothing procedure. For the smoothing of $\hat{\mu}_t^k$ and $(\hat{\sigma}_t^k)^2$ we use $\alpha = 0.7$. With this simple smoothing, the convergence to a degenerate distribution may happen too quickly and may lead to a sub-optimal solution. One possibility to overcome this problem is to follow the suggestions of Kroese et al. (2006) and use dynamic smoothing for the smoothing of $(\hat{\sigma}_t^k)^2$. For this purpose, (5) is replaced by the following one: $(\hat{\sigma}_t^k)^2 := \beta_t(\hat{\sigma}_t^k)^2 + (1-\beta_t)(\hat{\sigma}_{t-1}^k)^2$ where: $\beta_t = \beta - \beta(1-\frac{1}{t})^q$, q is an integer typically between 5 and 10 and β is a smoothing constant typically between 0.8 and 0.99. The problem in this method is that now the algorithm has a very slow convergence which makes it more difficult to choose a good stopping criterion. In this paper, we chose the second possibility which was suggested of Botev and Kroese (2004) and use the technique of "variance injection": if at iteration t the maximal variance in the vector $(\hat{\sigma}_t^k)^2$ is lower than ε (we use $\varepsilon = 0.01$ according to some runs of the algorithm that help us to figure out what value can be considered as a small variance), add to all the variances of the vector the value $|C_t^* - C_{t-1}^*|K$ for some injection parameter K between 0.1 and 10 (we use $K = 2$ according to previous experimentations explained in Botev and Kroese 2004).

In step 5, we have to stop if the stopping criterion is met. Again, we chose to adopt the stopping criterion suggested by Botev and Kroese (2004): stop when the number of variance injections exceeds a specific limit N_{LIM} (we use $N_{\text{LIM}} = 5$).

Of course, the gates obtained from applying this algorithm are continuous values, whereas our solution is restricted to discrete values. Rather than using arbitrary rounding, we apply the CE algorithm using a discrete distribution developed in Bendavid and Golany (2009). We generate for each activity the bounds of the interval according to the discrete Uniform distributions: $DU[\mathbf{G}^{F,1}, \mathbf{G}^{C,1}]$ for the lower bound of the interval and $DU[\mathbf{G}^{F,2}, \mathbf{G}^{C,2}]$ for the upper bound of the interval, where $\mathbf{G}^{F,k}$ and $\mathbf{G}^{C,k}$ $(k = 1, 2)$ are the floor and ceiling values, respectively, of the continuous vector of bounds. For example, assume that for a two-activities project, we obtained the following continuous intervals: [3.7, 5.2] and [6.8, 9.1]. We then feed these results into a CE algorithm based on discrete distributions as explained in Bendavid and Golany (2009) (Sect. 3) in the following way: the initial distribution for the lower bound of the first activity will be the discrete uniform distribution $DU[3, 4]$, the initial distribution for the upper bound of the first activity will be $DU[5, 6]$, the initial distribution for the lower bound of the second activity will be $DU[6, 7]$, and the initial distribution for

the upper bound of the second activity will be $DU[9, 10]$. The update of the distribution parameters will also be done using discrete values, as explained in Bendavid and Golany (2009).

4 Computational study

4.1 Comparison of the CE algorithm to other methods for the case of "zero interval costs"

Setting intervals for start times of activities is a new concept in the context of project management. Hence, there are no other techniques that can be considered as straightforward candidates for comparison with our method. Therefore, we decided to study the performance of Algorithm 3.2 first in the case where the interval costs c_j are equal to zero and compare it to the following heuristic methods:

- ES-LS: this is a simple heuristic method that consists of setting for each activity an interval where the lower bound of the interval is the expected early start time of the activity and the upper bound of the interval is the expected late start time of the activity.
- Simulated Annealing (SA): SA is a general purpose heuristic that has been used to solve various combinatorial problems. A detailed description of SA can be found in Heragu (1997) or Golany et al. (1999). The SA heuristic starts with an initial feasible solution and an initial "temperature". Then, it randomly generates a neighbor solution. If the new solution has a better performance, it is accepted. Otherwise, it can still be accepted with a probability that depends on the degradation in the objective function and on the temperature of the process at that stage. This routine is repeated until some termination conditions are met. Then, the temperature is decreased and the same steps are repeated. The detailed SA algorithm (Algorithm A.3) which was adapted and used in the present paper is presented in Appendix A.

We also compare the interval cross-entropy method (I-CE) developed in the present paper to the cross-entropy method for determining gates (G-CE) which was developed in Bendavid and Golany (2009). To carry out these comparisons, we first need a tool capable of generating project networks. This was done by employing the Progen software (see Kolish et al. 1995) that gave us a random number of activities and a random network structure represented by a matrix of successors. In addition, for each activity i, we generated the parameters: a_i, b_i, h_i, p_i and d for the entire project. These parameters were uniformly generated in an interval between some minimal and maximal values. One possible configuration for small projects (up to 20 activities) could be: $a_i \sim DU[2, 10]$, $b_i \sim DU[8, 20]$, $h_i \sim DU[1, 5]$, $p_i \sim DU[3, 7]$. The penalty p_n for the last activity should be significantly larger than the penalty of the other activities since it represents the penalty for not finishing the project on the due date. We set $p_n = 4p_{max}$ where p_{max} is the maximal value of the interval wherein the penalty p_i was generated. For large projects, we added a parameter called the type of the activity. For each activity we randomly set a "time type" that determines if it is a long, medium or short activity. Then, for each type we generate the parameters a_i and b_i from different intervals. In the same way, for each activity we randomly set a "cost type" that determines if it is an expensive, medium or inexpensive activity and then, for each type we generate the penalty costs h_i and p_i from different intervals. This allows many possible configurations for a project that could be composed of short or long activities, cheap or expensive penalties. In addition, for the entire project we need to generate a due date. To do so, we calculate what we called d_{min}, which is the minimal completion time of the project.

Table 1 Average cost for different sizes of projects

Project size	Average I-CE	G-CE	ES-LS	SA
5–20	665.8225	1215.59	1932.7586	1104.9624
20–30	1497.4908	4377.3274	8438.5898	4302.1696
30–40	2390.4512	6312.9342	14098.6123	6968.2156
40–50	2630.7695	8168.7927	16197.2604	9922.5847

In the same way, we calculate d_{max}, which is the maximal completion time of the project. Then, we generate d uniformly in the interval $[d_{min}, d_{max}]$.

We programmed Algorithm 3.2 in Matlab and applied it to ten random projects with a random number of activities between 5 to 20, ten random projects with a random number of activities between 20 to 30, ten random projects with a random number of activities between 30 to 40 and ten random projects with a random number of activities between 40 to 50. We compared the performances of the I-CE algorithm to the ES-LS heuristic, the SA heuristic and the G-CE method. The results that were obtained are summarized in Table 1. The data and results of a detailed project can be found in Appendix B.

4.2 The influence of the length of the interval on the cost

It is clear from the previous results that the approach of determining for each activity an interval confers flexibility to the contract and leads to reduced costs in comparison with the gating approach. In Sect. 4.1, we could reduce considerably the cost of a project since for each activity we choose the better interval, with no cost imposed on its length. In reality, the length of the intervals are negotiated in the contract with the subcontractor. In some types of contracts we need to pay a cost for each time unit of the interval. Of course we will be ready to pay such costs if the savings obtained by using intervals are greater than the costs. In this section, we check how the lengths of the intervals influence the cost of a specific project. For this purpose, we applied several times Algorithm 3.2 using the following algorithm:

Algorithm 4.2: the influence of the length of the interval on the cost

1. Apply on the project Algorithm 3.2 without any constraint on the intervals length.
2. Compute the maximal interval length obtained in step 1.
3. Repeat step 4 until the interval length is less than τ (we choose $\tau = 5$).
4. Decrease the interval length by τ and apply Algorithm 3.2 with the new interval length as a constraint on the intervals of all activities.
5. Apply Algorithm 3.2 with the interval length set to zero to all activities, which is equivalent to the gating approach (developed in Bendavid and Golany 2009).

We checked the influence of the interval lengths on a project network with seven activities. The results are presented in Fig. 3.

We can see from Fig. 3 that larger intervals lead to lower costs and that the minimal cost is obtained for unconstrained interval lengths. We can also see in Fig. 3 that the diminution of the cost is larger for small interval lengths and then it stabilizes. Thus, for example, it is more beneficial to pay an extra cost in order to raise the interval length from 10 to 15 than to raise it from 50 to 55.

With the insight that in reality intervals are constrained and with some changes to the basic algorithm we can handle different types of negotiations with subcontractors:

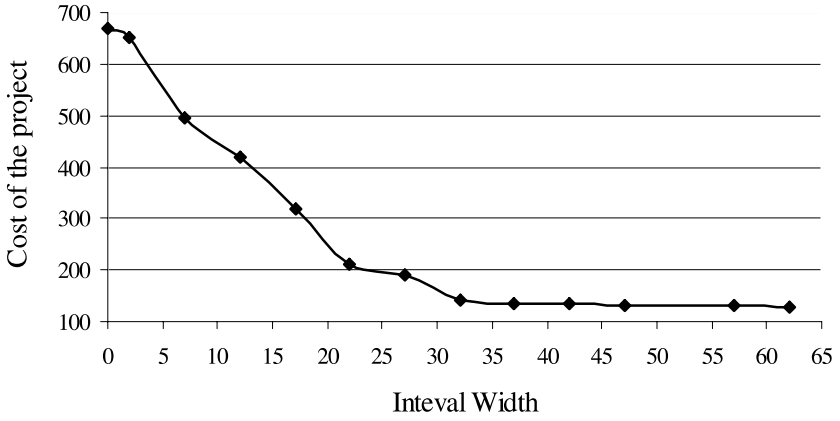

Fig. 3 The cost of a project for different intervals length

- If the cost per time unit is given by the subcontractor, we can find the "best" interval for each activity. This can be done using Algorithm 3.2.
- If the length of the intervals are given/constrained by the subcontractor, we can find the lower and upper bounds of the activities such that they respect this constraint. This can be done by replacing the second step of Algorithm 3.2 by the following one:

  Step 2 Generate a random sample $\mathbf{G}_1^1, \ldots, \mathbf{G}_N^1$ from the Normal distribution $N(\hat{\mu}_{t-1}^1, (\hat{\sigma}_{t-1}^1)^2)$, a random sample $\mathbf{G}_1^2, \ldots, \mathbf{G}_N^2$ from the Normal distribution $N(\hat{\mu}_{t-1}^2, (\hat{\sigma}_{t-1}^2)^2)$. Check for each generated vector i of intervals if $\mathbf{G}_i^2 - \mathbf{G}_i^1 \leq \mathbf{L}$ where $\mathbf{L} = (L_1, \ldots, L_n)$ and L_k is the constrained length of activity k. Otherwise generate again vector i until the constraint is respected. Calculate the performance (the total cost of the project) for each vector of intervals: $C(\mathbf{G}_i^1, \mathbf{G}_i^2)$. Order these performances from largest to smallest and let $\mathcal{N}^{\text{elite}}$ be the set of indices of the $\lceil \rho N \rceil$ vectors that give the best performances among the N vectors generated.

- Finally, if the length of the intervals are given/constrained by the subcontractor, but the cost is set by negotiations with the subcontractor, we can find how much we are ready to pay per time unit for a specific activity in the following way:

  1. Calculate the cost of the project according to the G-CE method, i.e. without intervals.
  2. Calculate the cost of the project according to the I-CE method developed in Sect. 3.2 while adding the constraint of the length for the specific activity as explained in the previous bullet.
  3. Calculate the difference between these two costs: this is the potential saving for adding an interval for this activity. This is also the maximum we are willing to pay to the subcontractor for adding flexibility in the contract by allowing intervals.

5 Concluding remarks

This paper presents a new approach for the stochastic project scheduling problem. Rather than trying to determine an exact release gate for each activity, we attempt to introduce some flexibility into the schedule by determining intervals in which activities are expected to start. The underlying problem is too complex to be solved exactly and therefore we rely on the CE

heuristic which has already proven its effectiveness in other settings including the context of projects with stochastic activity durations.

The proposed method may help the PM carry out different types of negotiations with subcontractors. If the cost per time unit for the interval is given by the subcontractor, the PM can use the method to find the best interval for each activity. If the length of the intervals are constrained by the subcontractor, the PM can use the method to find the lower and upper bounds of the activities that will adhere to these constraints. Finally, if the length of the intervals are constrained by the subcontractor, but the interval costs can be negotiated with the subcontractor, the PM can use the method to find how much he is ready to pay per time unit for the intervals associated with each specific activity.

Extensions of the method we presented here might include the scheduling of activities in projects where certain activities are restricted to take place within pre-defined time-windows (i.e., where the constraints are on both the start and the finish time of these activities). Another interesting direction for future research may involve a simultaneous determination of the interval lengths and the mode (time-cost trade-off) in which each activity will be carried out.

Appendix A: Specification of the SA algorithm

A.1 Notation

The additional notation used in this section is as follows:

ρ	the percentage by which the gate of an activity changes to create a new solution
T_{Limit}	the number of times the temperature is decreased
T_r	the temperature used in step r, $r = 1, \ldots, T_{\text{Limit}}$
N_{max}	maximum number of solutions evaluated at each temperature
δ	the difference between the cost of the candidate solution and the cost of the current solution
CR	the cooling rate by which the temperature is decreased

A.2 Generation of neighboring solutions

1. At iteration r, randomly select a number i between 1 to n.
2. Randomly select sign1 to be equal to 1 or -1.
3. Set: $g_i^1(r) = g_i^1(r-1) + \text{sign1}(\rho g_i^1(r-1))$ where ρ was set to 0.1.
4. Randomly select sign2 to be equal to 1 or -1.
5. If sign2 $= 1$ we will enlarge the interval. Therefore, randomly select $g_i^2(r)$ in the interval: $[g_i^2(r-1) + \text{sign1}(\rho g_i^1(r-1)), d]$. Otherwise we will tighten the interval. Therefore, randomly select $g_i^2(r)$ in the interval: $[g_i^1(r), g_i^2(r-1) + \text{sign1}(\rho g_i^1(r-1))]$.
6. Check feasibility: if $g_i^1(r) < 0$, $g_i^2(r) < 0$, $g_i^1 > d$ or $g_i^2 > d$ goto 1, otherwise stop.

A.3 Simulated annealing algorithm

Algorithm A.3: SA algorithm for setting intervals

1. Set $r = 1$, $T_r = 0.1$.
2. Select as an initial solution for the gates the vector composed of the early start times of the activities, and compute the cost of the initial solution.

3. Repeat steps 4–5 N_{max} times, where N_{max} was set to 1,000.
4. Select a neighbor solution according to Sect. A.2 and compute its cost.
5. If the cost of the candidate solution is lower than the current cost, accept the candidate solution with probability 1. Otherwise, accept it with probability $e^{-\delta/T_r}$ where δ is the difference between the cost of the candidate solution and the cost of the current solution ($\delta > 0$).
6. Set $r = r + 1$. If $r \leq T_{\text{Limit}}$ (T_{Limit} was set to 10), set $T_r = T_{r-1}\text{CR}^{r-1}$, where CR was set to 0.9 and goto 3. Otherwise, stop.

Appendix B: Data and results of a detailed project

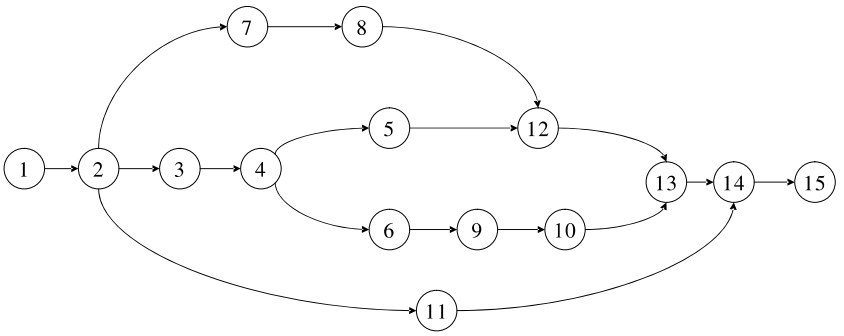

Fig. 4 Network for a project with 15 activities and $d = 190$

Table 2 Data for a project with 15 activities and $d = 190$

Activity i	a_i	b_i	h_i	p_i
1	24	38	14	14
2	14	24	13	14
3	10	24	10	16
4	15	29	10	12
5	25	38	2	4
6	4	14	5	10
7	3	15	3	5
8	3	11	12	16
9	15	30	1	3
10	14	30	3	5
11	21	38	3	7
12	5	13	5	6
13	13	23	3	4
14	23	32	1	4
15	23	33	11	48

Table 3 Results for a project with 15 activities and $d = 190$.

	I-CE	G-CE	ES-LS	SA
Cost	1268.4761	2599.633	4583.693	1950.1048

References

Alon, G., Kroese, D. P., Raviv, T., & Rubinstein, R. Y. (2005). Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, *134*, 137–151.

Bassok, Y., & Anupindi, R. (2008). Analysis of supply contracts with commitments and flexibility. *Naval Research Logistics*, *55*(5), 377–491.

Ben-Tal, A., Golany, B., Nemirovski, A., & Vial, J. P. (2005). Retailer-supplier flexible commitments contracts: a robust optimization approach. *Manufacturing & Service Operations Management*, *7*(3), 248–271.

Bendavid, I., & Golany, B. (2009, accepted). Setting gates for activities in the stochastic project scheduling problem through the cross entropy methodology. *Annals of Operations Research*.

Botev, Z., & Kroese, D. P. (2004). Global likelihood optimization via the cross-entropy method with an application to mixture models. In Ingalls, R. G., Rossetti, M. D., Smith, J. S., & Peters, B. A. (Eds.), *Proceedings of the 2004 winter simulation conference* (pp. 529–535). Washington: IEEE.

Buss, A. H., & Rosenblatt, M. J. (1997). Activity delay in stochastic project networks. *Operations Research*, *45*(1), 126–139.

Cohen, I., Golany, B., & Shtub, A. (2005). Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology. *Annals of Operations Research*, *134*, 189–199.

Elmaghraby, S. E. (2001). On the optimal release time of jobs with random processing times, with extensions to other criteria. *International Journal of Production Economics*, *74*, 103–113.

Elmaghraby, S. E., Ferreira, A. A., & Tavares, L. V. (2000). Optimal start times under stochastic activity durations. *International Journal of Production Economics*, *64*, 153–164.

Fazar, W. (1959). Program evaluation and review technique. *American Statistician*, *13*(2), 10.

Golany, B., Dar-El, E. M., & Zeev, N. (1999). Controlling shop floor operations in a multi-family, multi-cell manufacturing environment through constant work-in-process. *IIE Transactions*, *31*, 771–781.

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

Heragu, S. (1997). *Facilities design*. Boston: PWS Publishing Company.

Herroelen, W. S., & Leus, R. (2005). Project scheduling under uncertainty: survey and research potentials. *European Journal of Operational Research*, *165*, 289–306.

Herroelen, W. S., Van Dommelen, P., & Demeulemeester, E. L. (1997). Project network models with discounted cash flows: a guided tour through recent developments. *European Journal of Operational Research*, *100*, 97–121.

Ignizio, J. P. (1982). *Linear programming in single- & multiple-objective systems*. Englewood Cliffs: Prentice-Hall.

Kimms, A. (2001). *Mathematical programming and financial objectives for scheduling projects*. Massachusetts: Kluwer.

Kolish, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, *41*(10), 1693–1703.

Kroese, D. P., Rubinstein, R. Y., & Porotsky, S. (2006). The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, *8*, 383–407.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a technique for research and development of program evaluation. *Operations Research*, *7*, 646–669.

Rand, G. K. (2000). Critical chain: the theory of constraints applied to project management. *International Journal of Project Management*, *18*, 173–177.

Romero, C. (1991). *Handbook of critical issues in goal programming*. Headington Hill Hall: Pergamon.

Rubinstein, R. Y., & Kroese, D. P. (2004). *The cross-entropy method—a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. New York: Springer.

Sobel, M. J., Szmerekovsky, J. G., & Tilson, V. (2009). Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, *198*(3), 697–705.

Trietsch, D. (2005). Economically balanced criticalities for robust project scheduling and control. Working paper, ISOM Department, University of Auckland, New Zealand.

Trietsch, D. (2006). Optimal feeding buffers for projects or batch supply chains by an exact generalization of the newsvendor result. *International Journal of Production Research*, *44*(4), 627–637.