
A robust optimisation approach to project scheduling and resource allocation

Elodie Adida* and Pradnya Joshi

Department of Mechanical and Industrial Engineering,
University of Illinois at Chicago,
Chicago, IL, USA

Email: elodie@uic.edu Email: pjoshi5@uic.edu

*Corresponding author

Abstract: In a rapidly changing and highly uncertain environment, Petri net-based project management systems can be used in the rescheduling (control reconfiguration) of projects, when unforeseen changes occur or new data estimates become available. However, having to reschedule the project after it has started can result in significant suboptimality due to increased cost. In this paper, we present a solution method which is, to a large extent, robust to data uncertainty and does not require rescheduling for a certain predefined level of uncertainty. We build the model using robust optimisation techniques which address data uncertainty in project parameters, by taking advantage of risk pooling and without knowledge of their probability distributions. We illustrate that the total cost when the robust solution is used is generally lower than the cost of reconfiguring the deterministic solution, or than a penalty cost due to overtime of the deterministic solution, for a high enough penalty cost per time unit. We find that the robust solution is better protected against constraint violations, including time and cost overruns.

Keywords: optimisation under uncertainty; robust optimisation; resource allocation; project management; Petri nets; control reconfiguration.

Reference to this paper should be made as follows: Adida, E. and Joshi, P. (2009) 'A robust optimisation approach to project scheduling and resource allocation', *Int. J. Services Operations and Informatics*, Vol. 4, No. 2, pp.169–193.

Biographical notes: Elodie Adida is an Assistant Professor in the Department of Mechanical and Industrial Engineering at the University of Illinois at Chicago. She received her PhD in Operations Research from the Massachusetts Institute of Technology in 2006, and a Diplôme d'Ingénieur from Ecole Centrale Paris, France, in 2002. Her research interests include mathematical programming, nonlinear dynamic optimisation, game theory, optimisation under uncertainty and applications of operations research to supply chain and health care.

Pradnya Joshi received her Master of Science Degree in Industrial Engineering from the University of Illinois at Chicago in 2008. She holds a Bachelor of Engineering in Mechanical Engineering from Osmania University, India.

1 Motivation, literature review and contribution

1.1 Motivation

Modelling data uncertainty is an important aspect of Operations Research. In project management systems, the results that are obtained assuming that the project parameters are deterministic are no longer valid when these parameters take realised values different than expected. Data may lack precision, rendering the computed optimal solution suboptimal, or even infeasible by leading to constraint violations. It is therefore important to ensure that the planning and resource allocation tools take this uncertainty into account and allow scheduling and resource allocation that is robust to data uncertainty, so that the schedule remains feasible and close to optimum even if data is changed slightly. For example, in the aftermath of a disaster, decision-makers sometimes have to make decisions based on somewhat inaccurate and constantly updated data. The efficiency of the rescue actions undertaken and the adequate allocation of the scarce resources available, greatly affect the number of casualties. It is crucial to develop appropriate planning tools so as to be prepared to respond in the most efficient fashion.

The scheduling and assigning of resources in project management systems is done effectively using Petri net-based supervisory control reconfiguration techniques (Haji and Darabi, 2007). A Petri net supervisory controller is developed that enforces the task precedence relationships and the resource constraints for a project. In the event of any change in the actual progress from the planned progress, the developed supervisory controller is reconfigured and the new solution is used to control the project. This allows fast rescheduling in response to new information, but may affect the cost of the project greatly. Our goal is to compute a solution that is close to optimal even as the data is perturbed, thus eliminating the need for reconfiguration to a certain extent.

Stochastic optimisation is the most traditional way to address data uncertainty (Dantzig, 1955). However, determining the probability distribution of the uncertain parameters is not always easy and may be sometimes unrealistic. Robust optimisation has been introduced in the last decade as an alternative to stochastic optimisation (Bertsimas et al., 2004). The motivation behind robust optimisation is, on the one hand, to not require a particular probability distribution on the data and on the other hand, to find an alternative to overly conservative worst-case reasoning, in which one assumes that the value of the data that is the least favourable occurs for all uncertain parameters at all times.

Introducing a *budget of uncertainty* on the data is an efficient way to measure the trade-off between conservativeness and performance (Bertsimas and Sim, 2004). The budget of uncertainty represents the overall cumulative amount of variation away from nominal values that must be shared among uncertain parameters. The budget of uncertainty is input in the model; the modeller can decide whether he/she wants to obtain a more conservative solution (by choosing a large budget of uncertainty) while sacrificing optimality, or a solution that performs well and is less immune to data uncertainty (by choosing a smaller budget of uncertainty). The motivation behind such a model is to take advantage of risk pooling and protecting the system only against the most likely outcomes.

1.2 Literature review

The Operations Research literature treats the presence of data uncertainty in optimisation problems in several ways. The problem is sometimes solved assuming all parameters are

deterministic; subsequently sensitivity analysis is performed to study the stability of the nominal solution with respect to small perturbations of the data. Stochastic programming is used when the probability distribution of the underlying uncertain parameters is known. In the last decade, some researchers have focused on treating uncertainty in an alternative way, known as robust optimisation. A robust optimisation formulation was first considered by Soyster (1973) in the case of a linear optimisation problem where the data were uncertain within a convex set. He addresses uncertainty by taking a worst-case approach and was criticised of being overly conservative. Ben-Tal and Nemirovski (1998, 1999, 2000) address the issue of over-conservativeness by considering data uncertainty sets that are ellipsoids. Bertsimas and Sim (2004) introduce the notion of budget of uncertainty to control the level of conservativeness. Bertsimas et al. (2004) use uncertainty sets described by an arbitrary norm. Bertsimas and Brown (2005) construct uncertainty sets for LPs by taking a coherent risk measure on primitive. Bertsimas and Sim (2006) propose a relaxed robust counterpart for general conic optimisation problems.

The robust optimisation methodology has been applied to a number of areas. Bertsimas and Sim (2003) use the robust optimisation approach for discrete optimisation and network flow problems. Bertsimas and Thiele (2006) apply robust optimisation principles to inventory theory and supply chain management. Ben-Tal and Nemirovski (1997) use robust optimisation for truss topology design. Goldfarb and Iyengar (2003) apply robust optimisation to portfolio selection problems. Ben-Tal et al. (2005) take a robust optimisation approach for multi-period stochastic operations management problems, and in particular the retailer–supplier flexible commitment problem with uncertain demand. Adida and Perakis (2006, 2009) apply the robust optimisation methodology to a problem of dynamic pricing and inventory control. They investigate the differences with a stochastic optimisation formulation of the problem in Adida and Perakis (2008).

Over the last few decades, researchers have developed several project management tools that can be used for resource allocation and scheduling of projects (Aytug et al., 2005; Deblaere et al., 2006). Project scheduling and project management problems are often solved using network diagrams such as the critical path method (Angus et al., 2000). Petri nets represent a good alternative to manage projects where schedules depend on resource allocation and have to be replanned over time (Petri, 1966). Murata (1989) describes the history, properties, analysis methods and application areas of Petri nets. A variety of papers describe how Petri nets can be used to model the project dynamics (see e.g. Magott, 1989; Kim et al., 1995; Kumar and Ganesh, 1998; Jeetendra et al., 2000; Reddy et al., 2001; Sampath, 2004; Gullo and Agostoni, 2007; Haji and Darabi, 2007). Kumar and Ganesh (1998) illustrate that Petri nets provide a powerful formalism for representing and analysing concurrent systems. They use Petri nets to facilitate resource allocation in projects under some conditions commonly encountered in practice. Reddy et al. (2001) address the use of Petri nets as a scheduling and modelling tool in multi-mode and multi-resource constrained project scheduling. In Kim et al. (1995), Petri nets are used to model construction project tasks but not for project monitoring. Sampath (2004) discusses Petri net-based supervisory control reconfiguration techniques. Haji and Darabi (2007) show the applications of Petri net-based supervisory control reconfiguration techniques in the control of projects when the project progress is known. They develop a supervisory controller which can be reconfigured to modify and revise the plan based on the actual progress.

1.3 Contribution

- 1 We consider the deterministic, the reconfiguration and the robust optimisation approach. We study the advantages of each approach and show the differences in the solutions they provide.
- 2 We provide an equivalent deterministic formulation to the robust model: the robust counterpart problem.
- 3 The model incorporates uncertainty not only in the cost coefficients but also in the duration of each task alternative.
- 4 We compare in a numerical study the complexity of the different approaches and their performance in terms of constraint violations and realised cost.

In Section 2, we introduce the original deterministic model used in the control and reconfiguration of a project management system. In Section 3, we introduce the robust formulation and develop the robust counterpart problem. In Section 4, we carry out a numerical study to compare the two models on a specific numerical example and interpret the results.

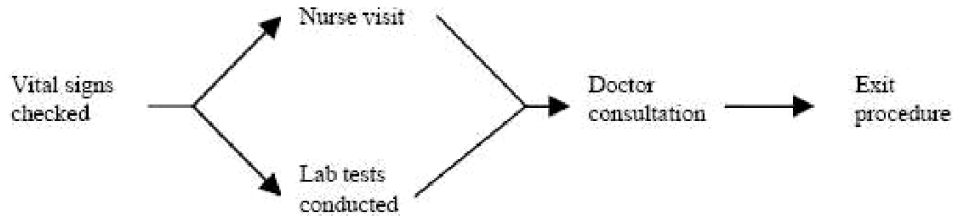
2 The model

In this section, we introduce the Petri net-based supervisory controller and give a brief description of the integer program used for the control and reconfiguration of the supervisory controller. This model was developed by Haji and Darabi (2007). The details can be found in Gullo and Agostoni (2007) and Haji and Darabi (2007).

2.1 The deterministic model

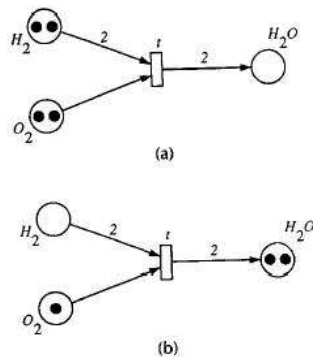
We consider a project involving n tasks a_1, a_2, \dots, a_n (Haji and Darabi, 2007). Each task uses some or all of m different types of resources for its execution. R_k represents the available number of resource type K . Based on the amount of resource utilised, each task realises a certain execution speed, out of the different duration alternatives. Different costs are associated with the different alternatives. For example, consider a medical support system wherein a patient must undergo certain medical tasks before he leaves the system (Figure 1). The workflow shows all the tasks that must be performed before the patient is discharged. It also gives the precedence relationships of these tasks. There are two types of resources in this system: doctors and nurses. All tasks do not use both the resources. For example, the task nurse visit does not need any doctors. Each task may be performed in different durations of time. The number of nurses and doctors required is different for different duration alternatives and hence different alternatives incur different costs. If the number of doctors and nurses available is not sufficient, additional resources can be hired at a certain cost. The scheduling of each task, the duration of each task and number of additional resources to hire may be optimised under some constraints on the system like an assigned finish time for the job.

Figure 1 Medical support system example



The modelling of such a workflow is done using Petri nets. A Petri net is defined by places, transitions and directed arcs. The input to a transition is a place from which an arc runs to a transition and an output of a transition is a place to which an arc runs from a transition. Places contain tokens, which determine if a transition can be fired, or not. Every arc has the weight associated with it. A transition is enabled or fired if the number of tokens in an input place is greater than the weight of the arc connecting the input place to the transition. The distribution of tokens over a place is called a marking. In Murata (1989), the chemical reaction producing water is illustrated. The transition t is enabled as two tokens in each input place, showing two units of H_2 and O_2 , are available. After firing t , the marking changes such that the transition can no longer be fired (Murata, 1989) (See Figure 2).

Figure 2 An illustration of transition firing: (a) marking before transition t is enabled and (b) marking after firing transition t , t is disabled (Murata, 1989)



I is the input state-transition matrix and represents the amount of tokens taken by each transition from its input places when the transition is fired (Haji and Darabi, 2007). O is the output transition matrix and represents the amount of tokens sent by each transition to its output places when the transition is fired. Incidence matrix of a Petri net is defined by $D^P = O - I$. The state equation $D^P \cdot \sigma = M - M_0$ involves the initial marking of the Petri net (M_0), the target marking to be reached (M) and the number of times each transition is be fired to reach the target (firing count vector σ). The Petri net showing the behaviour of the uncontrolled system is called a plant Petri net. The control specifications are imposed on the plant Petri net as a set of linear constraints on the marking of the plant Petri net places. The supervisory controller, used to restrict the behaviour of the plant Petri net so as to satisfy the constraints, is constructed by adding appropriate number of

control places to the plant Petri net and linking these places to the required existing transitions. During runtime, the coefficients in the constraints may change causing a change in the control specifications and a reconfiguration is required. To implement the reconfiguration, for every control transition, a unique transition, called the control transition, is defined. If the current state of the plant is infeasible, represented by negative markings in the control places based on the new specifications, we fire the new controls (or plant transitions) to turn the negative markings positive. There is a cost associated with firing the transitions. The objective of reconfiguration is to remove the negative tokens by executing the minimum cost firing count vector of transitions. This vector is found by solving an Integer Programming (IP) model.

The following IP model is used in control and reconfiguration of discrete event systems (Haji and Darabi, 2007). The model is an integer program corresponding to the system without explicit time dimension. This is done by associating two times with each transition, defining the static interval. The first time value denotes the minimal time elapsed from enabling the transition to firing it; the second denotes the maximum time the transition can wait before it fires. Thus one has to establish two time values for each transition. The model is tractable and its small size allows it to be solved in a short period of time.

$$\text{Minimise } \sum_{t \in H} c_t \sigma_t \quad (1)$$

$$\text{Subject to } \begin{bmatrix} D^p & 0 \\ D^c & I \end{bmatrix} \cdot \sigma = \begin{bmatrix} Y^p \\ Y^c \end{bmatrix} - \begin{bmatrix} M^p \\ M^c \end{bmatrix} \quad (2)$$

$$\sigma_{t_i} - \sum_{t_j \in {}^* p} \sigma_{t_j} \leq 0 \quad \text{for } t_i \in T^p, p \in {}^* t_i \quad (3)$$

$$S_j - F_i \geq 0 \quad \text{for } a_i, a_j \in A_{2,i} \quad (4)$$

$$F_i - S_i - \sum_{j=1}^{n_i} d_{ij} \sigma_{t-s-a_{ij}} \geq 0 \quad \text{for every } a_i \quad (5)$$

$$F_i - F \leq 0 \forall a_i \quad (6)$$

$$\sigma_{t-end} = 1 \quad \text{where } t-end \in p-term - a_i^* \quad (7)$$

$$F_i - S_j \geq 1 + M(x_{ij} - 1) \quad (8)$$

$$F_i - S_j \leq Mx_{ij} \quad (9)$$

$$F_j - S_i \geq 1 + M(x_{ji} - 1) \quad (10)$$

$$F_j - S_i \leq Mx_{ji} \quad (11)$$

$$(x_{ij} + x_{ji} + x_{ik} + x_{ki} + x_{jk} + x_{kj}) \geq 6 + M(x_{ijk} - 1) \quad (12)$$

$$(x_{ij} + x_{ji} + x_{ik} + x_{ki} + x_{jk} + x_{kj}) \leq 5 + Mx_{ijk} \quad (13)$$

$$\sum_{i=1}^n (x_{n-1})_i \geq n + M(x_n - 1) \quad (14)$$

$$\sum_{i=1}^n (x_{n-1})_i \leq n - 1 + Mx_n \quad (15)$$

$$\sum_i \sum_p^{n_p} r_{ipk} \cdot \sigma_{ip} \leq R_k + \sigma_k + M(2 - x_{ij} - x_{ji}); n = 2; k = 1, 2, 3, \dots, m; \quad (16)$$

$p = 1, 2, 3, \dots, n_p$ and $n =$ number of overlaps

$$\sum_p r_{ipk} \cdot \sigma_{ip} \leq R_k + \sigma_k; \quad k = 1, 2, \dots, m \quad (17)$$

$$\sum_i \sum_p^{n_p} (r_{ipk} \cdot \sigma_{ip}) \cdot x_n \leq R_k + \sigma_k; \quad k = 1, 2, \dots, m \quad (18)$$

$$\sigma_t = 0, 1 \quad \text{for plant transitions} \quad (19)$$

$$\sigma_t \geq 0 \text{ and integer for control transitions} \quad (20)$$

$$Y^p, Y^c, S_i, F_i \geq 0 \text{ and integer} \quad (21)$$

x_{ij}, x_{ji}, x_n binary and M a large integer positive number

In the above model, the objective (1) is to minimise the costs of firing transitions (c_t) in the Petri net. The variable σ_t represents the elements of the firing count vector and shows if the transition t is selected for firing. The total cost is obtained over all $t \in H$, where H is the set of all plant transitions T_p and all control transitions T_c .

Constraint (2) sets the current marking of the Petri net to a target marking and is a representation of the state equation. D_p is the incidence matrix of the plant Petri net, D_c the controller incidence matrix, Y_p the target marking of the plant places, Y_c the target marking of the control places, M_p the current marking of the plant Petri net and M_c is the current marking of the control Petri net.

Constraint (3) guarantees that a task is executed only if enough of the resources required are available, i.e. a transition is fired only if all of its input places have enough tokens. Here, *t_i is the set of input places of transition t_i and *p is the set of input transitions of place p .

Constraint (4) ensures that a task cannot begin until all the prerequisite tasks have been executed. $A_{2,i}$ is the set of all activities that a_i is a direct prerequisite for. S_j is the start time of activity j and F_i is the finish time of activity i .

Constraint (5) forces the end time of a task to be no less than its duration added to its starting time. d_{ij} is the duration of activity i with resource set up j and n_i is the number of resource-duration alternatives.

Constraint (6) enforces a given deadline (F) on the finish time of the project. Constraint (7) ensures that the project is executed, i.e. not executing the tasks is infeasible. As this is an integer program without time dimension, its solution may correspond to a resource allocation that is not feasible: due to the overlapping of tasks, multiple activities simultaneously use a particular kind of resource, requiring more resource than is available. Therefore, additional resources need to be hired for project

completion. Overlap constraints are introduced to know if the tasks are overlapping and based on that, to find out the maximum amount of resources the project needs at a given point.

Constraints (8)–(11) represent the overlap restriction for two activities. For the two tasks i, j that may be overlapping, two different variables x_{ij} and x_{ji} are introduced. There is an overlap between two tasks if the value of both variables is equal to one.

Constraints (12) and (13) show the overlap restriction for a set of three activities. Constraints (14) and (15) generalise to an overlap of n activities.

Constraint (16) forces the model to hire extra resources if there are not enough available when two tasks overlap. Here, r_{ipk} is the amount of resource k needed for the activity i using the transition p of the plant Petri net, σ_{ip} the element of the firing vector for a transition p , σ_k the control transition of the k th resource and R_k is the marking of the control place which gives the availability of resources. Constraints (17) and (18) generalise to an overlap of n tasks.

Constraint (19) gives binary condition for plant transitions and integer conditions for the number of resources to hire or fire. The details of the model can be found in Gullo and Agostoni (2007) and Haji and Darabi (2007).

3 The robust formulation

In computing a solution to the above model, the data is assumed to be precisely known. But the cost coefficients c_i and the duration coefficients d_{ij} are subject to uncertainty. Due to uncertainty in the cost parameters, the total cost of the project may be higher than what would have been expected in the deterministic model. Uncertainty in the duration coefficients may result in finish time constraint violations for each task (constraint 5) and violation of project deadline constraint (constraint 6). In some applications, violating the project finish time may be permitted but would incur a penalty per time unit. In other applications, if the finish time must be met a reconfiguration is necessary. The cost may be increased significantly in either of these two cases.

Since we have a variety of numerical parameters subject to uncertainty (task durations and task costs), we introduce a budget of uncertainty for each category: a task duration budget for each task (to be shared by the multiple resource-duration alternatives for that task) and a task cost budget (to be shared by all the cost coefficients). The use of risk pooling in this model illustrates that for each task it is unlikely that all duration alternatives estimates are wrong. However, task durations do not share the budget with durations of another task or with cost parameters, as these uncertainties come from radically different sources and do not necessarily balance each other out.

We derive a robust model by considering the following uncertainty framework (Bertsimas and Sim, 2003, 2004). We assume that the realised value of the cost coefficient \tilde{c}_i takes values in $[c_i, c_i + \hat{c}_i]$ (where c_i is the nominal value and \hat{c}_i is the length of the range of uncertainty) and that every realised value of the duration coefficient \tilde{d}_{ij} takes values in $[d_{ij} - \hat{d}_{ij}, d_{ij} + \hat{d}_{ij}]$ (where d_{ij} is the nominal value and \hat{d}_{ij} is the half-length of the range of uncertainty). Let J_i be the set of indices of duration coefficients d_{ij} that are subject to uncertainty i.e. $J_i = \{j \mid \hat{d}_{ij} > 0\}$. Similarly, J_0 represents the set of indices of duration coefficients c_i that are subject to uncertainty, i.e.

$J_0 = \{t \mid \hat{c}_t > 0\}$. We introduce budgets of uncertainty Γ_i for the duration coefficients of duration alternatives of task i and Γ_0 for the cost coefficients, where $\Gamma_i \in [0, |J_i|]$ and $\Gamma_0 \in [0, |J_0|]$. The budget of uncertainty for task i means that at most $\lfloor \Gamma_i \rfloor$ of the coefficients are allowed to realise within their entire range and one-coefficient d_{it} is allowed to deviate by at most $(\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{d}_{it}$ (Bertsimas and Sim, 2003, 2004).

These range of realisation constraints and budget of uncertainty constraints define the uncertainty set of the uncertain data. Our goal is to find a solution that is close to optimum and feasible for any realisation of the data within this set.

Let Δ be the set of all realisations $\{\tilde{d}_{ij}, i = 1, 2, \dots, n, j \in J_i\}$ in the uncertainty set, i.e. satisfying the following constraints:

$$\begin{aligned} \tilde{d}_{ij} &\in [d_{ij} - \hat{d}_{ij}, d_{ij} + \hat{d}_{ij}] \\ \sum_{j \in J_i} \left| \frac{\tilde{d}_{ij} - d_{ij}}{\hat{d}_{ij}} \right| &\leq \Gamma_i \forall_i \end{aligned}$$

We want constraint (5) to be satisfied for any realisation of the data within the uncertainty set Δ . Thus the robust formulation of constraint (5) is given by:

$$F_i - S_i - \sum_{j \in J_i} \tilde{d}_{ij} \sigma_{t-s-a_{ij}} \geq 0, \quad \forall a_i, \forall \{\tilde{d}_{ij}, i = 1, \dots, n, j \in J_i\} \in \Delta \quad (22)$$

This represents an infinite number of constraints, as the inequality has to be satisfied for any realisation of the duration parameters within the uncertainty set, which includes an infinite number of points. We obtain a deterministic counterpart of this constraint by determining the realisation that represents the worst case (within the uncertainty set) for satisfying this inequality, i.e. by solving a subproblem that searches for the minimum of the left-hand side above:

$$\min_{\{\tilde{d}_{ij}\} \in \Delta} F_i - S_i - \sum_{j \in J_i} \tilde{d}_{ij} \sigma_{t-s-a_{ij}}$$

Notice that this worst realisation depends on the choice of decision variable σ of the original problem. This transformed constraint is then plugged back into the original problem in the place of constraint (22). This manipulation ensures that the original constraint involving uncertain durations will be feasible for any realisation within the uncertainty set defined by the range of uncertainty constraints and the budget of uncertainty constraint. We proceed in a similar way with uncertainties in the cost coefficients. We obtain that the robust counterpart of the Petri net integer model is:

$$\text{Minimise } \sum_{t \in H} c_t \sigma_t + z_0 \Gamma_0 + \sum_{t \in J_0} q_{0t} \quad (23)$$

$$\text{Subject to } \begin{bmatrix} D^p & 0 \\ D^c & I \end{bmatrix} \cdot \sigma = \begin{bmatrix} Y^p \\ Y^c \end{bmatrix} - \begin{bmatrix} M^p \\ M^c \end{bmatrix} \quad (24)$$

$$\sigma_{t_i} - \sum_{t_i \in_* p} \sigma_{t_i} \leq 0, \quad \text{for } t_i \in T^p, p \in_* t_i \quad (25)$$

$$S_j - F_i \geq 0, \quad \text{for } a_i, a_j \in A_{2,i} \quad (26)$$

$$F_i - S_i - \sum_{j=1}^{n_i} d_{ij} \sigma_{t-s-a_{ij}} \geq z_i \Gamma_i + \sum_{j \in J_i} q_{ij}, \quad \text{for every } a_i \quad (27)$$

$$z_0 + q_{0t} \geq \hat{c}y_t, \quad \forall t \in J_0 \quad (28)$$

$$z_i + q_{ij} \geq \hat{d}_{ij} y_j, \quad \forall i, \forall j \in J_i \quad (29)$$

$$-y_j \leq \sigma_{t-s-a_{ij}} \leq y_j, \quad \forall j \quad (30)$$

$$F_i - F \leq 0, \quad \forall a_i \quad (31)$$

$$\sigma_{t-end} = 1 \quad \text{where } t-end \in p-term - a_i^* \quad (32)$$

$$F_i - S_j \geq 1 + M(x_{ij} - 1) \quad (33)$$

$$F_i - S_j \leq Mx_{ij} \quad (34)$$

$$F_j - S_i \geq 1 + M(x_{ji} - 1) \quad (35)$$

$$F_j - S_i \leq Mx_{ji} \quad (36)$$

$$(x_{ij} + x_{ji} + x_{ik} + x_{ki} + x_{jk} + x_{kj}) \geq 6 + M(x_{ijk} - 1) \quad (37)$$

$$(x_{ij} + x_{ji} + x_{ik} + x_{ki} + x_{jk} + x_{kj}) \leq 5 + Mx_{ijk} \quad (38)$$

$$\sum_{i=1}^n (x_{n-1})_i \geq n + M(x_n - 1) \quad (39)$$

$$\sum_{i=1}^n (x_{n-1})_i \leq n - 1 + Mx_n \quad (40)$$

$$\sum_i^n \sum_p^{n_p} r_{ipk} \cdot \sigma_{ip} \leq R_k + \sigma_k + M(2 - x_{ij} - x_{ji}); n = 2; k = 1, 2, 3, \dots, m; \quad (41)$$

$p = 1, 2, 3, \dots, n_p$ and $n =$ number of overlaps

$$\sum_p^{n_p} r_{ipk} \cdot \sigma_{ip} \leq R_k + \sigma_k, \quad k = 1, 2, \dots, m \quad (42)$$

$$\sum_i^n \sum_p^{n_p} (r_{ipk} \cdot \sigma_{ip}) \cdot x_n \leq R_k + \sigma_k, \quad k = 1, 2, \dots, m \quad (43)$$

$$\sigma_t = 0, 1 \text{ for plant transitions} \quad (44)$$

$$\sigma_t \geq 0 \text{ and integer for control transitions} \quad (45)$$

$$Y^p, Y^c, S_i, F_i \geq 0 \text{ and integer} \quad (46)$$

$$y_j, z_i, q_{ij}, q_{0t} \geq 0 \quad \forall i, \forall j \in J_i, \forall t \in J_0 \quad (47)$$

x_{ij}, x_{ji}, x_n binary and M a large integer positive number

The robust counterpart is similar to the original problem in the sense that it is still a linear program. But the following differences can be observed:

- The number of constraints and the number of variables has increased.
- We introduce new variables y_j , z_i , q_{ij} , q_{0t} and z_0 .
- There is an additional linear term in the objective function (23) and the task finish time constraint is made tighter (27). The additional term $z_i\Gamma_i + \sum_{j \in J_i} q_{ij}$ in the duration constraint ensures protection by providing a margin, to allow for changes in the duration coefficients so that when the next task is scheduled to begin, the previous one is more likely to have ended and thus no reconfiguration is necessary. Similarly, the linear term in the objective protects against changes in the cost coefficients.

However, the overall structure is similar and the complexity of solving the problem has not increased. Therefore, the running time does not significantly increase when adapting the deterministic model to a robust model that protects the solution against data uncertainty.

4 Numerical study

In this section, we describe a numerical example (Gullo and Agostoni, 2007; Haji and Darabi, 2007) to compare the deterministic model without reconfiguration, the deterministic model with reconfiguration and the robust model. We illustrate the following:

- 1 The robust solution provides better protection against constraint violations when compared to the deterministic solution without reconfiguration, when data is perturbed. It is therefore more likely to allow to finish the project on time.
- 2 This gain in protection is achieved at a small increase in cost.
- 3 The constraint violation in the deterministic model may incur cost increase larger than cost increase due to robust if a penalty is incurred for each time unit of overtime.
- 4 The robust solution yields lower cost than the deterministic with reconfiguration if the project finish time must be met and the deterministic solution must be reconfigured as soon as it will not meet the finish time according to the current schedule.

4.1 Example

We consider the Petri Net example involving six activities a_1 , a_2 , a_3 , a_4 , a_5 and a_6 . The workflow uses two types of resources N and D. The network diagram of the project with its resource-duration alternatives is shown in Figure 3 (Gullo and Agostoni, 2007; Haji and Darabi, 2007). In this example:

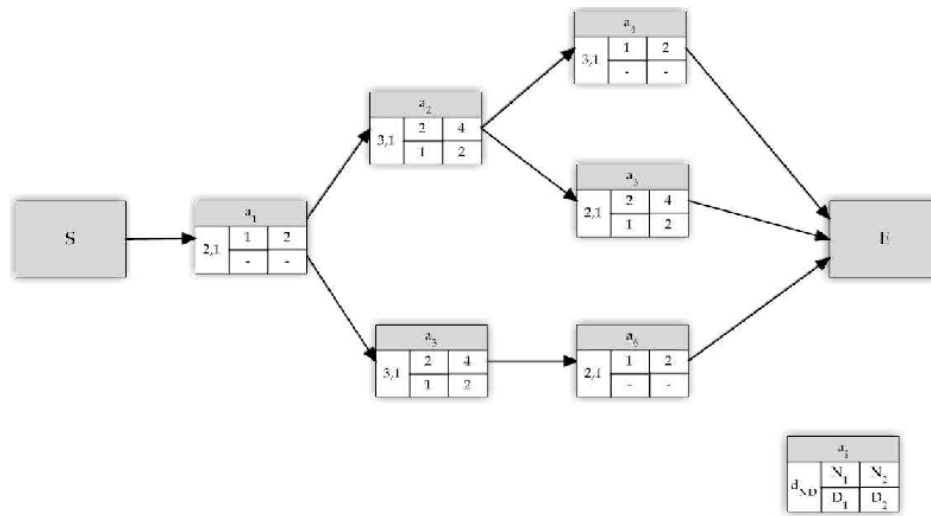
- 1 The number of resources N, D available when the project starts is 2 each: The resource availability, that is the marking of the control places, is $R_k = 2$.
- 2 The project finish time is 8: $F = 8$.

- 3 The cost of hiring resource N (firing the associated control transition) is 100 and that of hiring resource D is 200.
- 4 The cost of choosing one duration alternative over the other is shown in Table 1.

Table 1 Costs for the two duration alternatives

	a_1	a_2	a_3	a_4	a_5	a_6
Alternative 1	5	25	25	5	25	5
Alternative 2	10	50	50	10	50	10

Figure 3 Network diagram of the example (see online version for colours)



4.2 Robust solution vs. deterministic solution

We evaluate and compare the performance of the robust and the deterministic solutions when the uncertain data is selected randomly. We solve the robust model for different budgets of uncertainty. We then generate 1000 random realisations for all the cost and duration coefficients according to a normal distribution on the one hand and a uniform distribution on the other hand. In the case of the normal distribution, we generate the values for the uncertain coefficient \tilde{d}_{ij} from a distribution with mean d_{ij} and standard deviation $0.5\hat{d}_{ij}$. In the case of the uniform distribution, the support is $[d_{ij} - \hat{d}_{ij}, d_{ij} + \hat{d}_{ij}]$. Similarly, \tilde{c}_i is generated either from a normal distribution with mean c_i and standard deviation $0.5\hat{c}_i$, or from a uniform distribution on $[c_i, c_i + \hat{c}_i]$. We measure the performance of the robust and the deterministic solutions in terms of the average realised cost and the probability and amplitude of the finish time constraint violations over the 1000 generated realisations. The probability of the finish time constraint violations is found as the number of times the finish time constraint is negative out of the 1000 realisations. The amplitude is found as the average of the absolute value of the finish time constraint violation, every time it is violated. The inputs we choose are:

$$\Gamma_1 = \Gamma_2 = \Gamma_3 = \Gamma_4 = \Gamma_5 = \Gamma_6 \in [0, 2], \text{ while } \Gamma_0 = 0$$

$$\Gamma_0 \in [0, 14], \text{ while } \Gamma_1 = \Gamma_2 = \Gamma_3 = \Gamma_4 = \Gamma_5 = \Gamma_6 = 0$$

where Γ_0 is the budget of uncertainty of the cost coefficients and $\Gamma_i = \Gamma_1 = \Gamma_2 = \Gamma_3 = \Gamma_4 = \Gamma_5 = \Gamma_6$ are the budgets of uncertainty for the duration coefficients of each task. We carry out the numerical study for $\hat{d}_{ij} = 0.1d_{ij}$, $\hat{c}_i = 0.1c_i$ and $\hat{d}_{ij} = 0.5d_{ij}$, $\hat{c}_i = 0.5c_i$.

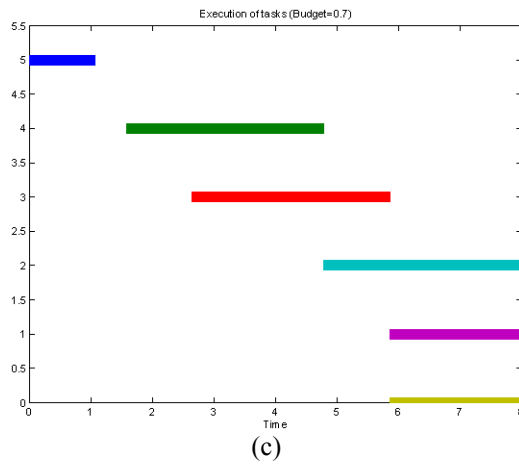
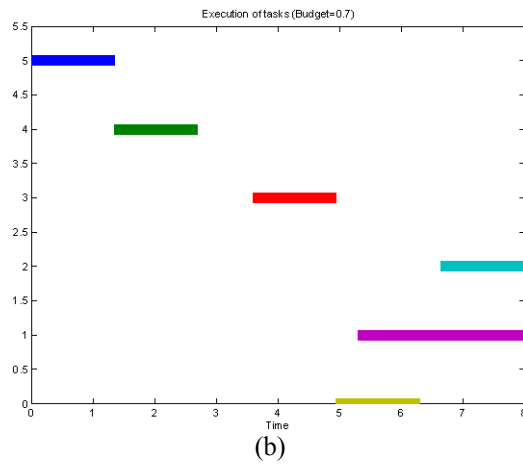
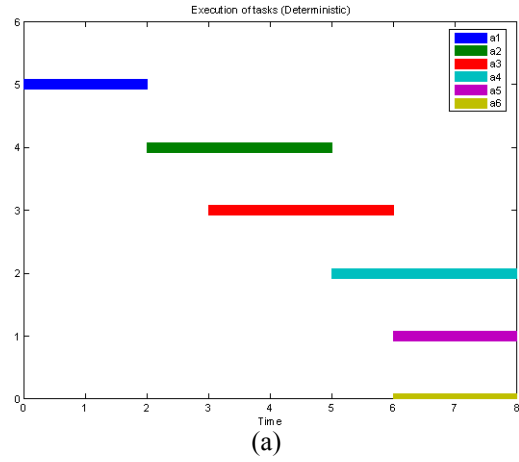
We observe no difference in the performance of the robust solution and the deterministic solution when uncertainty is introduced in only the cost coefficients (Table 2). The cost associated with alternative 2 is twice as that of alternative 1, for every task. For a range of uncertainty of $\hat{c}_i = 0.1c_i$ or $\hat{c}_i = 0.5c_i$, the maximum cost for alternative 1 is still much lower than the cost of the alternative 2. Hence, the resource allocations of the tasks and scheduling of tasks does not change for cost uncertainty in the robust model for any budget of uncertainty Γ_0 .

Table 2 Average realised costs for only cost uncertainty

	<i>Uniform distribution</i>	<i>Normal distribution</i>
Deterministic ($\hat{c}_i = 0.1c_i$)	304.594	305.216
Robust ($\hat{c}_i = 0.1c_i$)	304.594	305.216
Deterministic ($\hat{c}_i = 0.5c_i$)	362.97	366.078
Robust ($\hat{c}_i = 0.5c_i$)	362.97	366.078

When uncertainty is introduced in the duration coefficients, the robust task scheduling is found to be different from the deterministic solution for any budget of uncertainty. The robust solution is computed in a way that for any value of the task durations within the uncertainty set, changing the solution (reconfiguration) to avoid constraint violations will not be required. The tasks are scheduled in a way that provides the necessary buffer in each finish time constraint so that the start time of a task is greater than or equal to the finish task of the previous task even if the durations vary within the uncertainty set (Figures 4a–4c) and the project can be completed in 8 units of time. In Figures 4b and 4c, we observe that in the robust solution, task a_1 is completed in about 1 unit of time less than in the deterministic solution (Figure 4a). This change in the duration of task a_1 calls for a shorter duration alternative requiring more resources and thus increases the cost (Table 3). However, it allows to create buffer time necessary to avoid the need of reconfiguration. When the range of uncertainty in the coefficients is increased, the buffer time needed will also increase. In Table 4, for $\hat{d}_{ij} = 0.5d_{ij}$, a budget as low as 0.4 requires shorter duration alternatives for tasks a_1 and a_6 in the robust allocation when compared to the deterministic allocation, whereas, for $\hat{d}_{ij} = 0.1d_{ij}$ and any Γ_i only task a_1 requires a shorter duration alternative. Also, as the budget of uncertainty increases, the solution has to provide protection against bigger changes in a larger number of coefficients and more tasks require shorter duration alternatives (Table 4). This is because as the budget of uncertainty increases, the allowed cumulative variation of the uncertain coefficient increases. As the resource alternatives change, the cost also increases (Tables 5 and 6). In this example, even though the number of resources required for the tasks changed, the total number of N and D resources required in the system at any point of time remained at a maximum of 2 each, so no additional resources are hired.

Figure 4 Time diagrams showing the execution of the tasks: (a) deterministic, (b) robust, $\hat{d}_{ij} = 0.1d_{ij}, \Gamma_i = 0.7$ and (c) robust, $\hat{d}_{ij} = 0.5d_{ij}, \Gamma_i = 0.7$ (see online version for colours)



In Figures 5–8, we observe that the robust solution is better protected against constraint violations than the deterministic solution as the robust approach provides a buffer time to allow for the variation in the coefficients, whereas the deterministic solution is tight and a delay in a task will cause the project to end late. We observe that when the range of uncertainty is increased from $\hat{d}_{ij} = 0.1d_{ij}$ to $\hat{d}_{ij} = 0.5d_{ij}$, the amplitude of violation increases under both probabilistic distributions as the variance of the realised values is increased. However, the probability of constraint violation is similar. Therefore when the range increases, constraint violations occur as often, but when they do occur they are larger. We find that the probability and amplitude of constraint violations decreases with an increase in the budget illustrating that the robust model is better protected against constraint violations as Γ_i increases. This is because as the budget of uncertainty for the duration coefficients Γ_i increases, the solution will be robust to wider changes and in a larger number of coefficients.

Table 3 Resource requirements for $\hat{d}_{ij} = 0.1d_{ij}$

Tasks	a_1	a_2	a_3	a_4	a_5	a_6
(N,D)-Deterministic	(1,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)
(N,D)-Robust	(2,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)

Table 4 Resource requirements for $\hat{d}_{ij} = 0.5d_{ij}$

Tasks	a_1	a_2	a_3	a_4	a_5	a_6
(N,D)-Deterministic	(1,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)
(N,D)-Robust (Budget [0,0.25])	(2,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)
(N,D)-Robust (Budget [0.35,0.65])	(2,0)	(2,1)	(2,1)	(2,0)	(2,1)	(2,0)
(N,D)-(Budget [0.7,2])	(2,0)	(4,2)	(4,2)	(2,0)	(2,1)	(2,0)

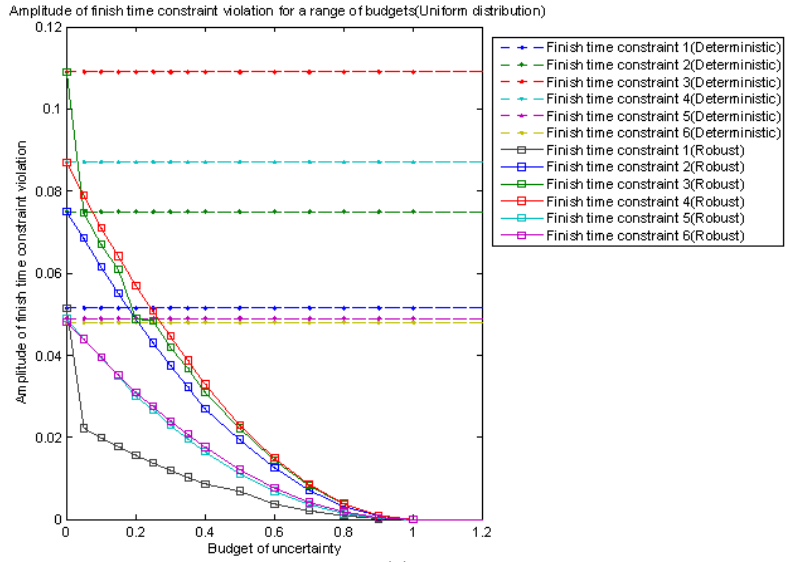
Table 5 Average realised costs for $\hat{d}_{ij} = 0.1d_{ij}, \hat{c}_i = 0.1c_i, \Gamma_0 = 0$

	Uniform distribution	Normal distribution
Deterministic solution	304.594	305.216
Robust solution	309.849	310.457

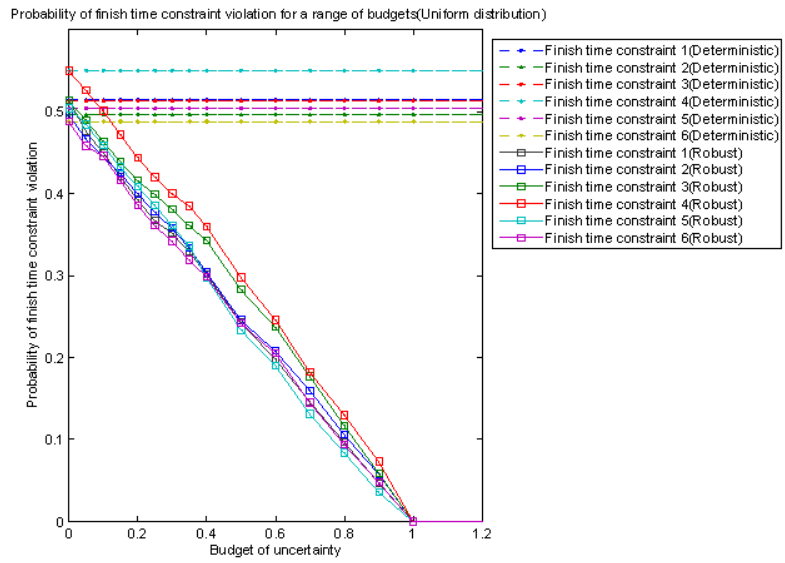
Table 6 Average realised costs for $\hat{d}_{ij} = 0.5d_{ij}, \hat{c}_i = 0.5c_i, \Gamma_0 = 0$

	Normal distribution	Uniform distribution
Deterministic solution	366.078	362.97
Robust ($\Gamma_i \in [0, 0.25]$)	372.314	369.247
Robust ($\Gamma_i \in [0.35, 0.65]$)	384.738	381.671
Robust ($\Gamma_i \in [0.7, 2]$)	448.219	443.905

Figure 5 Constraint violations over a range of budgets for $\pm 10\%$ range of uncertainty ($\hat{d}_{ij} = 0.1d_{ij}$), uniform distribution: (a) amplitude of violation and (b) probability of violation (see online version for colours)

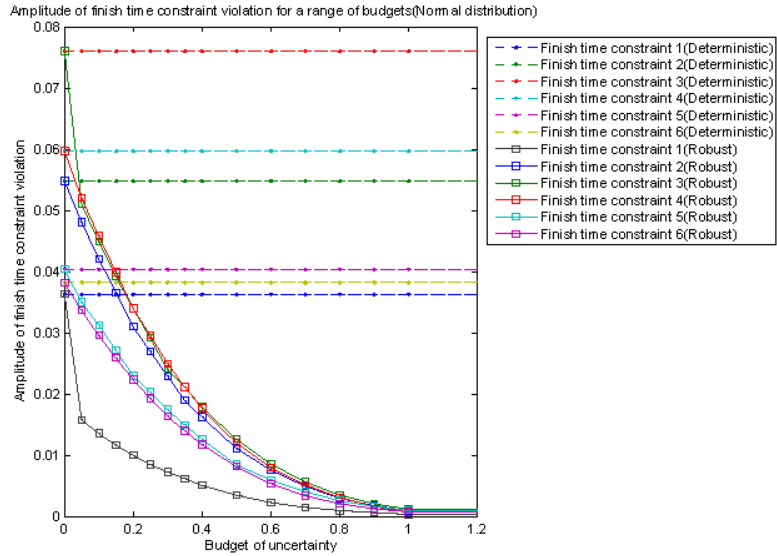


(a)

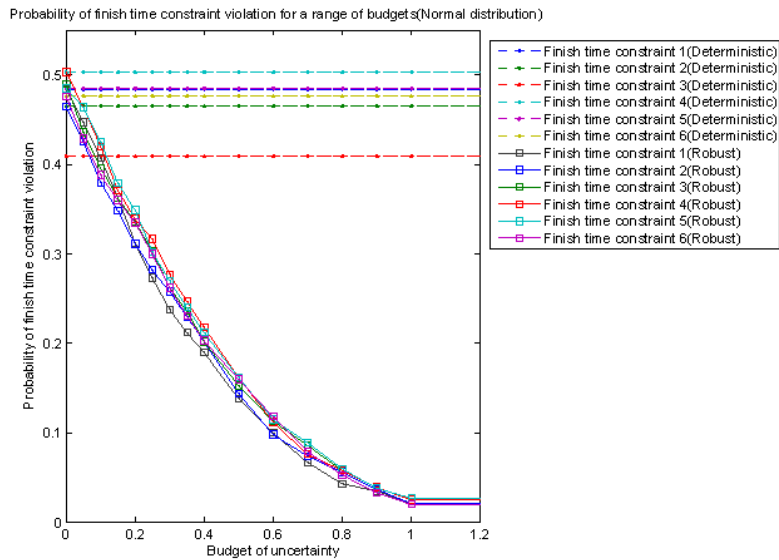


(b)

Figure 6 Constraint violations over a range of budgets for $\hat{d}_{ij} = 0.1d_{ij}$, normal distribution: (a) amplitude of violation and (b) probability of violation (see online version for colours)

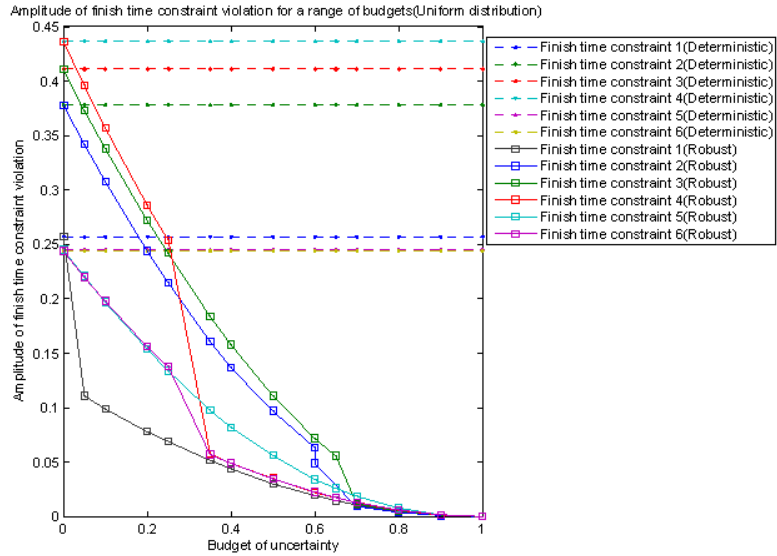


(a)

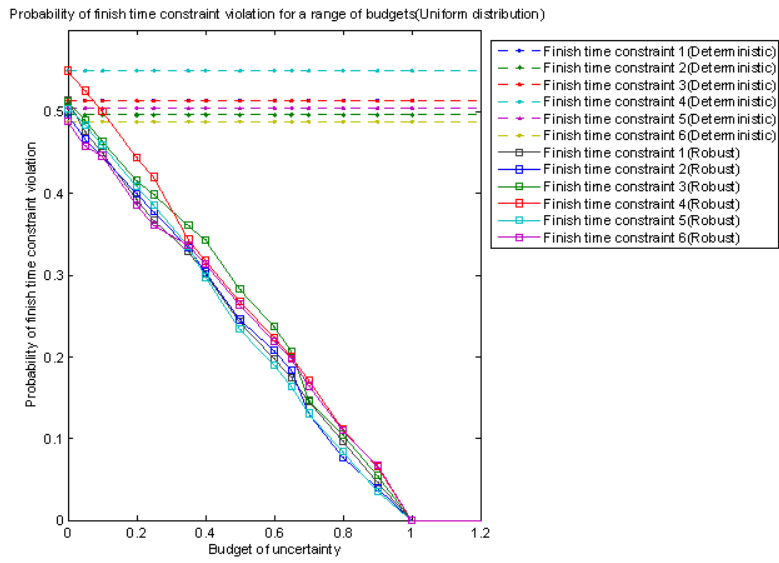


(b)

Figure 7 Constraint violations over a range of budgets for $\hat{d}_{ij} = 0.5d_{ij}$, uniform distribution: (a) amplitude of violation and (b) probability of violation (see online version for colours)

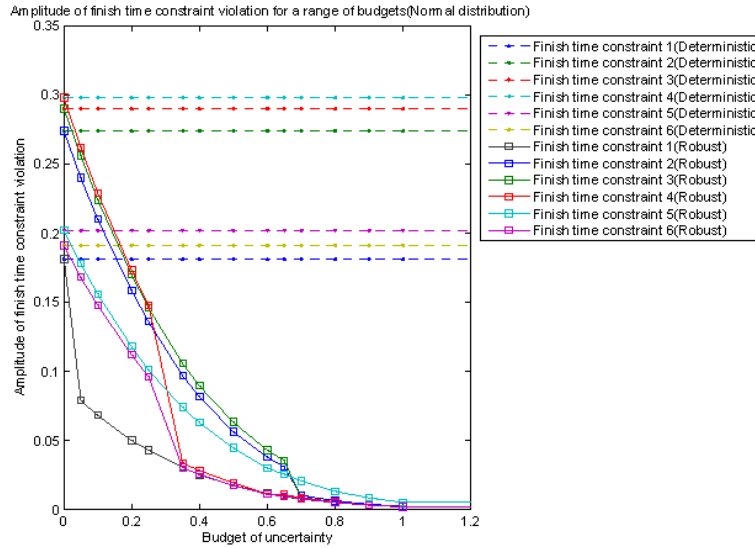


(a)

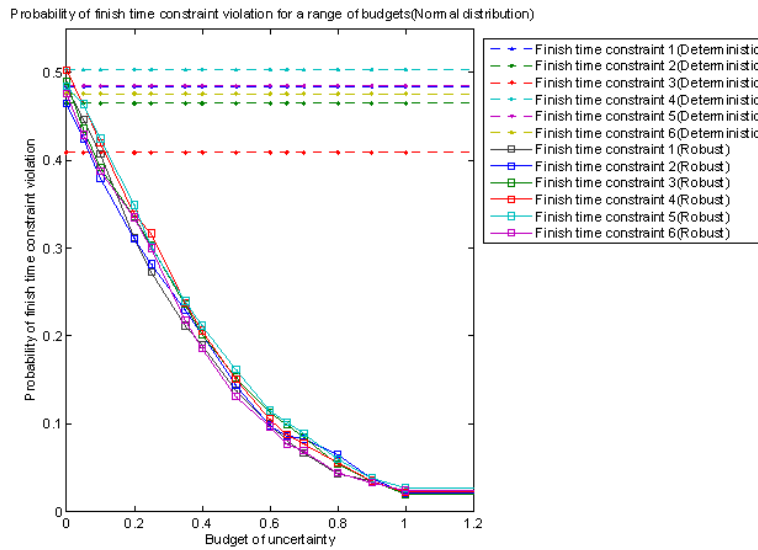


(b)

Figure 8 Constraint violations over a range of budgets for $\hat{d}_{ij} = 0.5d_{ij}$, normal distribution:
 (a) amplitude of violation (b) probability of violation (see online version for colours)



(a)



(b)

As seen above, uncertainty in the durations will result in task finish time constraint violations (constraint 5). This constraint states that the finish time of a task has to be greater than or equal to its start time plus its duration. If the realised duration is longer than what it would be based on nominal values, the next task will be delayed. As a result the project may not be able to meet the given deadline (F) and violates constraint (6). We consider the following two scenarios for comparing the performance of the robust and the deterministic solutions:

- the finish time deadline can be violated, incurring a penalty cost per time unit
- the finish time deadline must be met and reconfiguration is necessary if the current schedule does not allow to finish the project on time.

4.3 *Robust solution vs. deterministic solution without reconfiguration*

We compare the performance of the two solutions in the scenario where the total finish time of the project may go beyond the deadline of 8 units of time at an additional cost. We introduce a penalty cost per time unit beyond the finish time deadline. We find the new total cost of the project for the robust approach, and the deterministic approach without reconfiguration for different penalty costs per unit time and a range of budgets.

In Figure 9, we see that for no penalty cost (penalty cost = \$0), the cost of using the deterministic approach is always lower than the cost of using the robust approach as there is no penalty for violating the project deadline constraint and the robust solution corresponds to a resource allocation with higher total cost. However, when a penalty cost is incurred for violating the finish time deadline, the cost of using the robust approach can be lower than the cost of using the deterministic approach. This is because the penalty cost incurred due to constraint violation is lower for the robust approach than the deterministic approach since the deterministic solution may incur project deadline constraint violation and thus a penalty. In the case where $\hat{d}_{ij} = 0.1d_{ij}$, for penalty costs of \$60 and above the total cost of using the robust solution is always lower than the cost of using the deterministic solution (see Figures 9a and 9b). However, for penalty costs of \$50 or less, the robust cost is higher than the deterministic cost at lower budgets. This is because at low budgets the realised coefficients vary moderately from the nominal values and thus the penalty amount due to constraint violation is small and does not compensate the increase in the cost due to the change in the resource allocation corresponding to the robust solution. As the budget of uncertainty varies, the resource allocation changes once (for $\Gamma = 0.1$). For budgets greater than 0.1 the cost of the robust solution is nondecreasing with the budget of uncertainty as a higher budget implies less constraint violation and thus less penalty cost. In the case when $\hat{d}_{ij} = 0.5d_{ij}$, as the budget of uncertainty varies, the resource allocation changes three times ($\Gamma = 0.1, 0.35$ and 0.7). For a given resource allocation, the robust cost decreases with an increase in the budget of uncertainty (see Figures 9c and 9d), but each change in the resource allocation incurs an increase in cost due to the selection of a new duration alternative with higher cost to create more buffer time and allow more uncertainty in the task durations, providing better protection against constraint violation.

4.4 *Robust solution vs. deterministic solution with reconfiguration*

In applications where the finish time deadline must be met and violating the project finish time by incurring a penalty cost is not an option, reconfiguration becomes necessary. We illustrate that the cost for the robust approach is lower than the cost incurred in using the deterministic solution and reconfiguring the project when the actual progress is different from the planned progress due to uncertainty in duration coefficients.

Figure 9 Total cost over a range of budgets and various penalty costs: (a) uniform distribution, $\hat{d}_{ij} = 0.1d_{ij}$, (b) normal distribution, $\hat{d}_{ij} = 0.1d_{ij}$, (c) uniform distribution, $\hat{d}_{ij} = 0.5d_{ij}$ and (d) normal distribution, $\hat{d}_{ij} = 0.5d_{ij}$ (see online version for colours)

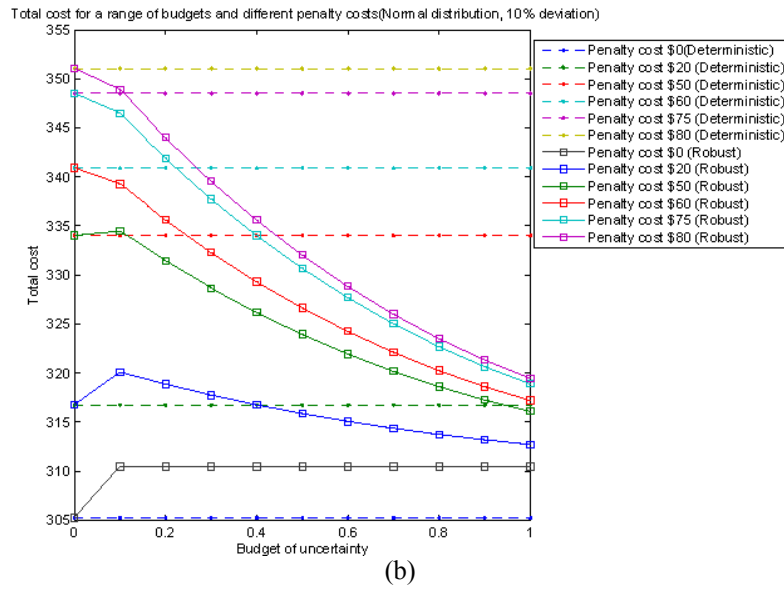
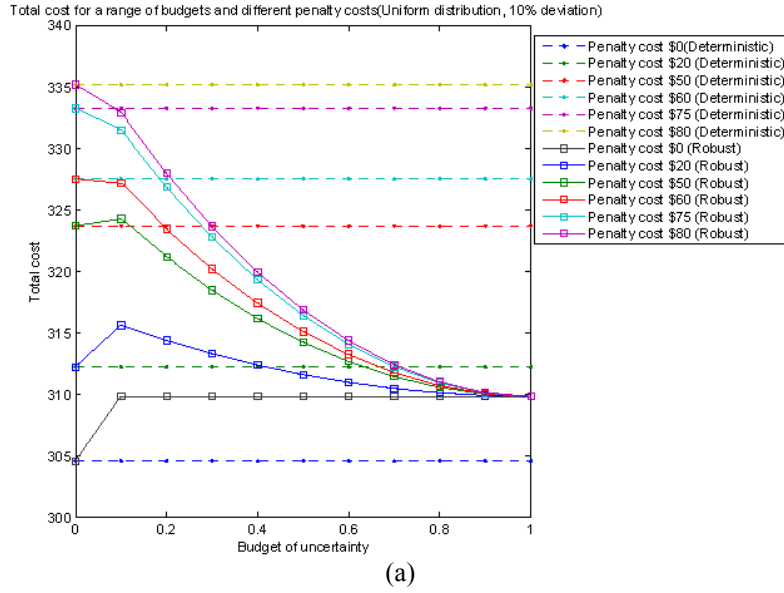
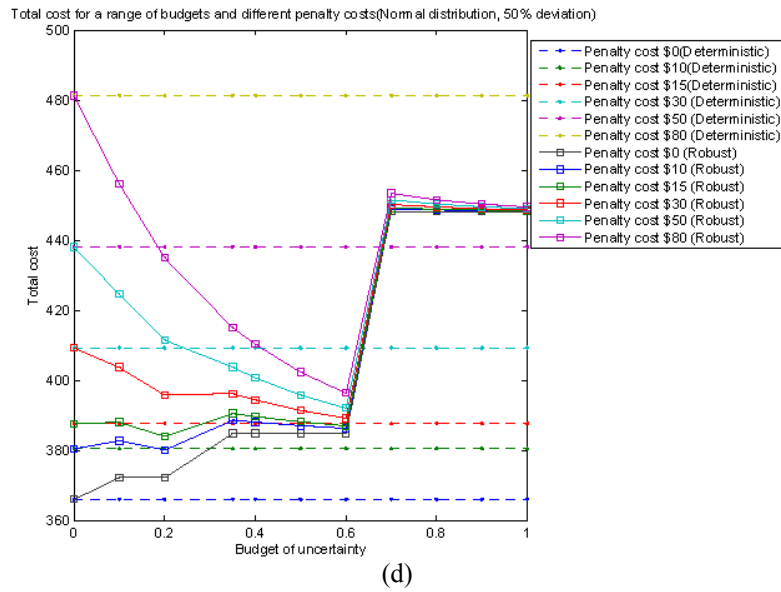
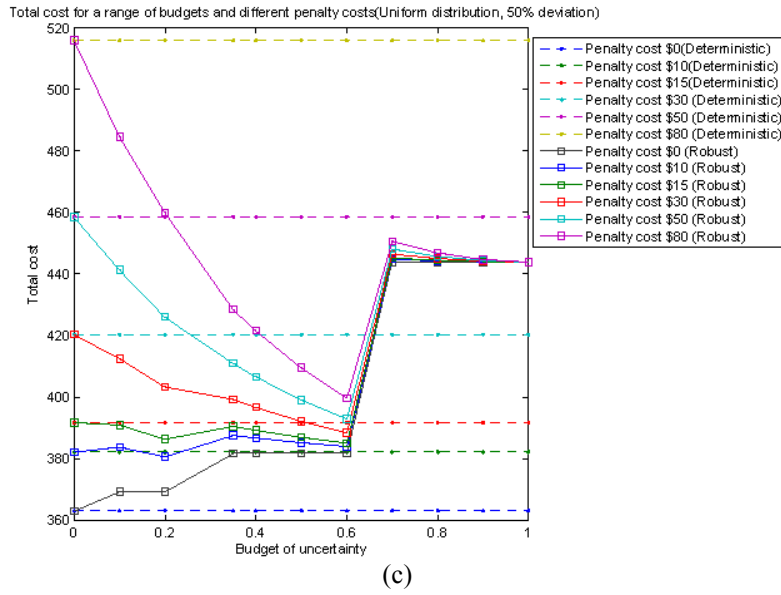


Figure 9 Total cost over a range of budgets and various penalty costs: (a) uniform distribution, $\hat{d}_{ij} = 0.1d_{ij}$, (b) normal distribution, $\hat{d}_{ij} = 0.1d_{ij}$, (c) uniform distribution, $\hat{d}_{ij} = 0.5d_{ij}$ and (d) normal distribution, $\hat{d}_{ij} = 0.5d_{ij}$ (see online version for colours) (continued)



We consider all possible scenarios in which the observed finish time of a various tasks is different from the expected finish time. When it is observed that the finish time deadline (F) cannot be attained based on the current scheduling and resource allocation, we reconfigure to find a new schedule for the project. As the deterministic solution is in general tight, in the sense that there is no buffer time, the project must be reconfigured as soon as the task ending late forces the next task to start late. Reconfiguration implies selecting a shorter duration alternative for at least one of the tasks left at that time. For example, if task a_1 under the resource alternative (1,0) (cost = \$5, nominal duration $d_{1j} = 2$) is delayed by $\hat{d}_{1j} = 0.5d_{1j} = 1$, then the computed deterministic solution, which gives a finish time of exactly 8 units, is no longer valid as the project finish time will go beyond 8 units of time due to the delay in task a_1 of one time unit. Thus, reconfiguration becomes necessary when task a_1 has ended late. Shorter duration alternatives for at least one of the remaining tasks are required to meet the deadline. In this example, reconfiguration leads to assigning a shorter duration alternative to tasks a_4 and a_6 (assigning two resources instead of one for both tasks, reducing the nominal durations to 1 unit of time for each task). This results in a cost increase of \$10. If the robust solution would have been used, there would be no need for reconfiguration, as the robust solution is well prepared to handle this change in the duration of task a_1 . There would thus be no further increase in cost (Table 7).

Table 7 Resource requirements, total cost for the deterministic solution with reconfiguration and the robust solution for $\hat{d}_{ij} = 0.5d_{ij}$

Tasks	a_1	a_2	a_3	a_4	a_5	a_6	Total cost
(N,D)-Deterministic	(1,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)	\$290
(N,D)-Reconfiguration	(1,0)	(2,1)	(2,1)	(2,0)	(2,1)	(2,0)	\$300
(N,D)-Robust	(2,0)	(2,1)	(2,1)	(1,0)	(2,1)	(1,0)	\$295

The change in duration alternatives due to reconfiguration may not be the same as the robust solution and the cost may be higher. In the same way, for a delay in tasks a_2 or a_3 the total cost after reconfiguration is also found to be \$300 as the resource allocations of tasks a_4 and a_6 also have to be changed. A delay in task a_4 while implementing the deterministic solution will not allow reconfiguration: the project cannot meet its finish time deadline because task a_4 is the last task of the project (see Figure 4a), whereas the robust solution can handle this delay. Delays in tasks a_5 or a_6 do not require reconfiguration of the deterministic solution since these tasks are scheduled to occur from time 5 to time 7, no task is scheduled to start after their completion, and the finish time deadline is 8, allowing a delay of up to 1 unit of time without creating a constraint violation. Table 8 summarises the results.

Table 8 Total cost for the deterministic solution with reconfiguration for different scenarios

Scenario	Total cost after reconfiguration	Total cost robust
Delay in a_1	\$300	\$295
Delay in a_2	\$300	\$295
Delay in a_3	\$300	\$295
Delay in a_4	No reconfiguration possible, project fails	\$295
Delay in a_5	\$290	\$295
Delay in a_6	\$290	\$295

Thus for a delay less than 1 unit of time in any of the tasks, the average cost of reconfiguration is found to be \$296 (disregarding the scenario when reconfiguration is not possible), which is higher than the cost of using the robust solution.

5 Conclusions

In this paper, we presented a robust optimisation model for project scheduling and resource allocation in highly uncertain environments. The original deterministic model is a Petri net-based project management system that can be reconfigured to obtain new schedules if the finish time deadline of the project cannot be met due to uncertainty in the project parameters. We compared the performance of the deterministic model and the formulated robust model. We showed that the robust model does not generally need rescheduling for a predefined level of uncertainty. We illustrated that the robust model provides better protection against constraint violations in the event of uncertainty when compared to the deterministic model at the expense of a small increase in cost. We showed that the robust cost is lower than the deterministic cost when violation of the finish time deadline of a project incurs a high enough penalty cost per time unit of overtime. We illustrated that the total cost of using the robust approach may be lower than the cost of using the deterministic approach with reconfiguration when exceeding the finish time deadline is not an option.

References

- Adida, E. and Perakis, G. (2006) 'A robust optimization approach to dynamic pricing and inventory control with no backorders', *Mathematical Programming*, Vol. 107, Nos. 1–2, pp.97–129.
- Adida, E. and Perakis, G. (2009) 'Dynamic pricing and inventory control: robust vs. stochastic uncertainty models', *working paper*.
- Adida, E. and Perakis, G. (2009) 'Dynamic pricing and inventory control: uncertainty and competition', *Operations Research* (forthcoming).
- Angus, R.B., Gundersen, N.A. and Cullinane, T.P. (2000) 'Planning, performing and controlling projects: principles and applications', 2nd ed., Prentice Hall, Upper Saddle River, NJ.
- Aytug, H., Lawley, M., McKay, K., Mohan, S. and Uzsoy, R. (2005) 'Executing production schedules in the face of uncertainties: a review and some future directions', *European Journal of Operational Research*, Vol. 161, No. 1, pp.86–110.
- Ben-Tal, A., Golany, B., Nemirovski, A. and Vial, J.P. (2005) 'Retailer-supplier flexible commitments contracts: a robust optimization approach', *Manufacturing and Service Operations Management*, Vol. 7, No. 3, pp.248–271.
- Ben-Tal, A. and Nemirovski, A. (1997) Robust truss topology design via semidefinite programming, *SIAM Journal on Optimization*, Vol. 7, No. 4, pp.991–1016.
- Ben-Tal, A. and Nemirovski, A. (1998) 'Robust convex optimization', *Mathematics of Operations Research*, Vol. 23, pp.769–805.
- Ben-Tal, A. and Nemirovski, A. (1999) 'Robust solutions to uncertain linear programs', *Operations Research Letters*, Vol. 25, pp.1–13.
- Ben-Tal, A. and Nemirovski, A. (2000) 'Robust solutions of linear programming problems contaminated with uncertain data', *Mathematical Programming*, Vol. 88, pp.411–424.

- Bertsimas, D. and Brown, D. (2005) *Robust Linear Optimization And Coherent Risk Measures*, Working Paper, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.
- Bertsimas, D., Pachamanova, D. and Sim, M. (2004) 'Robust linear optimization under general norms', *Operations Research Letters*, Vol. 32, pp.510–516.
- Bertsimas, D. and Sim, M. (2003) 'Robust discrete optimization and network flows', *Mathematical Programming*, Vol. 98, pp.49–71.
- Bertsimas, D. and Sim, M. (2004) 'The price of robustness', *Operations Research*, Vol. 52, No. 1, pp.35–53.
- Bertsimas, D. and Sim, M. (2006) 'Tractable approximations to robust conic optimization problems', *Mathematical Programming*, Vol. 107, Nos. 1–2, pp.5–36.
- Bertsimas, D. and Thiele, A. (2006) 'A robust optimization approach to supply chain management', *Operations Research*, Vol. 54, No. 1, pp.150–168.
- Dantzig, G.B. (1955) 'Linear programming under uncertainty', *Management Science*, Vol. 1, pp.197–206.
- Deblaere, F., Demeulemeester, E., Herroelen, W. and Van de Vonder, S. (2006) *Proactive Resource Allocation Heuristics for Robust Project Scheduling*, Research Report 0608, Department of decision sciences and information management, Katholieke Universiteit Leuven.
- Goldfarb, D. and Iyengar, G. (2003) 'Robust portfolio selection problems', *Mathematics of Operations Research*, Vol. 28, No. 1, pp.1–38.
- Gullo, P. and Agostoni, L. (2007) *A Petri Nets based Approach to Resource Constrained Project Scheduling Problems*, Masters of Science Thesis, Politecnico di Milano.
- Haji, M. and Darabi, H. (2007) 'Petri net based supervisory control reconfiguration of project management systems', *Proceedings of the 3rd Annual IEEE Conference*.
- Jeetendra, V.A., Krishnaiah Chetty, O.V. and Reddy, J.P. (2000) 'Petri nets for project management and resource levelling', *The International Journal of Advanced Manufacturing Technology*, Vol. 16, pp.516–520.
- Kim, J., Desrochers, A.A. and Sanderson, A.C. (1995) 'Task planning and project management using Petri nets', *IEEE International Symposium on Assembly and Task Planning*, pp.265–271.
- Kumar, V.K.A. and Ganesh, L.S. (1998) 'Use of Petri nets for resource allocation in projects', *IEEE Transactions on Engineering Management*, Vol. 45, No. 1, pp.49–56.
- Magott, J. (1989) 'Combined generalized stochastic Petri nets and PERT networks for the performance evaluation of concurrent processes', *Proceedings of the 3rd International Workshop on Petri Nets and Performance Models*, pp.249–256.
- Murata, T. (1989) 'Petri nets: properties, analysis and applications', *Proceedings of the IEEE*, Vol. 77, No. 4, pp.541–580.
- Petri, C.A. (1966) 'Communication with automata', Griffiss Air Force Base, New York, Tech. Rep. RADC-TR-65-377, Vol. 1, No. 1.
- Reddy, J.P., Kumanan, S. and Krishnaiah Chetty, O.V. (2001) 'Application of Petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling', *The International Journal of Advanced Manufacturing Technology*, Vol. 17, pp.305–314.
- Sampath, R. (2004) *Control Reconfiguration of Discrete event Systems with Dynamic Constraints*, Master of Science Thesis, University of Illinois at Chicago.
- Soyster, A. (1973) 'Convex programming with set-inclusive constraints and applications to inexact linear programming', *Operations Research*, Vol. 21, pp.1154–1157.