A Distributionally Robust Analysis of the Program Evaluation and Review Technique

Ernst Roos*, Dick den Hertog
Department of Econometrics and Operations Research, Tilburg Univeristy

May 7, 2020

Abstract

Traditionally, stochastic project planning problems are modeled using the Program Evaluation and Review Technique (PERT). PERT is an attractive technique that is commonly used in practice as it requires specification of only a few characteristics of the activities' duration. Moreover, its computational burden is extremely low. Over the years, four main disadvantages of PERT have been voiced and much research has been devoted to analyzing them. The effect of the beta distribution and corresponding variance PERT assumes is investigated in numerous studies, through analyzing the results for a variety of other distributions. In this paper, we propose a more general method of analyzing PERT's sensitivity to its assumptions regarding the beta distribution. In particular, we do not assume a singular distribution for the activity duration, but instead assume this distribution to only be partially specified by its support, mean and possibly its mean absolute deviation. The exact worst- and best-case expected project duration over this set of distributions can be calculated through results from distributionally robust optimization on the worst- and best case distributions themselves. A numerical study of project planning instances from PSPLIB shows that the effect of PERT's assumption regarding an underlying beta distribution is limited. Furthermore, we find that the added value of knowing the exact mean absolute deviation is also modest.

Keywords: project planning, distributionally robust optimization, PERT

1 Introduction

The efficient planning of projects has been studied extensively for decades. Optimization of decisions regarding the planning of projects under various circumstances has been investigated in for example (Demassey et al., 2005; Zhu et al., 2006). Even when the optimization aspect of project planning is disregarded, one is left with computationally challenging problems. In particular, if uncertainty is present in the duration of activities, determining the project length's distribution or even its expectation is usually a difficult computational problem. In this light, a common assumption in PERT, the Program Evaluation and Review Technique first proposed

*Corresponding author: e.j.roos@tilburguniversity.edu

by Malcolm et al. (1959), is that activity durations follow a beta distribution. To properly define this beta distribution, three values for each activity, a pessimistic, optimistic and most likely duration are assumed to be known. Subsequently, it is assumed that the standard deviation is one sixth of the range of the distribution and that a linear approximation for the mean is acceptable (Littlefield Jr and Randolph, 1987). Given this beta distribution, the critical set of activities is determined based on the mean activity duration that follows from this assumed beta distribution, and the expected project length is calculated by summing the mean activity durations of all critical activities.

PERT is particularly attractive because of its extremely low computational burden. Moreover, it only requires the support and most likely value of the activity durations to be known and/or estimated from historical data. Because of these advantages, PERT is widely used in practice and implemented in software packages such as MS Project.

On the other hand, PERT also has some disadvantages. In general, four main points of criticism exist (Hajdu, 2013):

1. An additional assumption on the variance is needed to fully specify the beta distribution;
2. The beta distribution might not appropriately model reality;
3. Activity durations are assumed to be independent;
4. The expected project duration found is usually too optimistic.

A great deal of research has been devoted to addressing these disadvantages. The assumption that activity durations are independent has been relaxed by Klein Haneveld (1986), among others. Klein Haneveld (1986) discusses the worst-case project duration distribution given (partly specified) marginal distributions.

Moreover, many alternatives to the beta distribution have been proposed, hoping to alleviate both issue 2 and 4. Hahn (2008), for example, discusses an extension of the beta distribution that allows a user to specify a different variability for different activities. Other suggested distributions include, but are not limited to, the doubly truncated normal distribution (Kotiah and Wallace, 1973) and the triangular distribution (Johnson, 1997). While many of these alternatives have specific advantages, the question remains whether these distributions do appropriately model reality.

In this paper, we analyze PERT's assumptions regarding the beta distribution and its variance, not by analyzing a specific alternative to the beta distribution, but by considering all distributions with a specified support, mean and possibly mean absolute deviation. We use techniques from Distributionally Robust Optimization to derive tight upper and lower bounds for the expected project duration given this distributional information.

Distributionally Robust Optimization (DRO) is a recent stream of research that focuses on optimization under uncertainty. It assumes there is only limited information on the distribution of the uncertain parameters. This limited information is captured in an ambiguity set: a set

that contains all probability distributions under consideration. In the context of PERT we are mostly interested in results from DRO concerning expectations, i.e., methods to compute upper and lower bounds on an expectation of a nonlinear function for a given ambiguity set. Such results are presented in Wiesemann et al. (2014) for ambiguity sets with moment conditions and confidence sets, and Postek et al. (2018) for ambiguity sets with a specified support, mean and mean absolute deviation, for example. For a comprehensive overview of DRO we refer to Rahimian and Mehrotra (2019).

In this paper, we focus on the distributional results from Postek et al. (2018), because they provide a closed-form expression for tight upper and lower bounds on the expectation of a convex function, expanding upon results from Ben-Tal and Hochman (1972). This allows us to consider a set of potential distributions, instead of assuming a beta distribution with a specified variance, and hence we can analyze PERT's disadvantages with respect to those assumptions. Moreover, by considering both the best- and worst-case distribution in this set, we are guaranteed not to obtain an optimistic expected project duration.

These best- and worst-case distributions are shown to be a two and three point distribution, respectively. By comparing the expected project duration for the best- and worst-case distributions, we can comment on the importance of actually knowing the true distribution. We remark that although these two- and three-point distributions might not be realistic distributions for the activity duration, we do not consider them as such; they are simply used as a tool to obtain tight upper and lower bounds on the expected project duration.

Birge and Maddox (1995) also present an approach to bounding the expected project duration given limited information on the activity duration distribution, which is somewhat similar to ours. Because they use the variance instead of the mean absolute deviation, however, the bounds they propose are not tight. Therefore, they cannot necessarily draw definitive conclusions with respect to PERT's core assumptions.

It is important to note that PERT generally only considers a single scenario for the whole project, in which all activities are at their mean value. This implies that PERT only considers a single critical path: the path that is critical when all activities are at their mean duration. Due to the nature of the three-point distribution that attains the worst-case expected project duration, our method considers every possible combination of the minimum, mean and maximum duration for each activity. We thus consider every possible critical path, thereby avoiding one of the main causes of PERT's optimistic result. Because we consider all combinations of the possible values for each activity, however, computing the best- and worst-case expected project duration requires the project duration to be computed for an exponential amount of scenarios. In this paper, we also present an algorithm that severely diminishes the (exponential) number of scenarios that have to be considered. Moreover, we present techniques to obtain bounds on the worst- and best-case project duration, one of which is particularly effective, such that we can obtain slightly weaker bounds with much less computational effort. These bounds can be employed whenever the project at hand contains a prohibitively high number of activities. Employing these techniques, we can effectively bound the expected project duration for projects

with up to 120 activities.

The contributions of this paper can be summarized as follows:

1. We propose a new method of analysis that yields tight upper and lower bounds for the expected project duration when PERT's assumptions on the beta distribution and its variance are relaxed.

2. We analyze a large set of projects from PSPLIB with this method and show that the gap between the worst-case expected project duration and PERT's result is below 1% on average and always below 3% when activity duration can deviate up to 15%. We thus find that PERT's assumptions on distribution and variance are not particularly detrimental.

The remainder of the paper is organized as follows. Section 2 elaborates on the mathematical background of PERT. Section 3 introduces the distributionally robust techniques we apply to this framework. Section 4 elaborates on the algorithm we develop to more efficienltly compute the resulting expressions. Section 5 discusses relaxations to obtain upper and lower bounds for both the best- and worst-case expected project length. In Section 6 we present the results of the numerical experiments for both the algorithm as well as the discussed bounds for instances from PSPLIB. Section 7 concludes the paper.

2 The Basics of PERT

Projects as modeled by PERT (Malcolm et al., 1959) are represented by a weighted directed graph $G = (V, A)$. Here, $V = \{1, \ldots, n\}$, where 1 represents the start of the project and n represents the completion. Activities of the project are represented by arcs $a \in A$, with a duration $d_a \in \mathbb{R}_+$, and we denote by m the number of activities $|A|$. Precedence constraints between activities are modeled by the nodes $2, \ldots, n-1$, in the sense that activities originating from a node can only be started when all activities ending in that node are completed. The minimum amount of time in which the project can be completed is found by finding the longest (also called critical) path in G from 1 to n.

Given a vector of activity durations $\boldsymbol{d}$, we will denote the length of the longest path in G from node 1 to node i by $f_i(\boldsymbol{d})$ for all $i \in V$. Because any graph constructed by the logic above cannot contain cycles, we can always order the nodes in such a way that $i > j$ implies $(i, j) \notin A$ and hence $f_n(\boldsymbol{d})$, the project duration, can be computed by recursively computing the longest path to the nodes in the graph in increasing order:

$$f_i(\boldsymbol{d}) = \max_{j:(i,j)\in A} \left\{ f_j(\boldsymbol{d}) + d_{(i,j)} \right\}. \tag{1}$$

Hence, $f_n(\boldsymbol{d})$ can be calculated in $\mathcal{O}(|A| + |V|)$. For the sake of simplicity, we will not carry the subscript and refer to the project duration with $f(\boldsymbol{d})$ throughout the rest of the paper.

Typically, activity durations are assumed to include some form of uncertainty. In the traditional PERT approach, all activity durations are assumed to be independent and follow a beta

distribution. More specifically, it is assumed that three values are specified for each activity a: the most likely activity duration, m_a, an optimistic duration, l_a, and an pessimistic duration, u_a. By additionally assuming the standard deviation of the activity duration to be given by $\sigma_a = \frac{1}{6}(u_a - l_a)$, the beta distribution is fully specified. In particular, we know that its mean is given by

$$\mu_a = \frac{l_a + 4m_a + u_a}{6}.$$

The main objective of the PERT analysis is to determine an accurate estimate of the project duration's distribution. To this end, the constructed graph G is analyzed when all activity lengths are equal to their mean duration, that is, $f(\boldsymbol{\mu})$ is calculated. Based on this analysis, the activities that are in the longest path found when determining $f(\boldsymbol{\mu})$ are marked as critical. The project duration's variance is then calculated by

$$\sigma_P^2 = \sum_{a \in CP} \sigma_a^2,$$

where CP is the collection of critical activities. The project duration's distribution is assumed to be normal with the above characteristics, i.e.,

$$f(\boldsymbol{d}) \sim N\left(f(\boldsymbol{\mu}), \sigma_P^2\right).$$

3 A Distributionally Robust Analysis of PERT

In contrast to the stochastic approach in PERT, it is common in Distributionally Robust Optimization (DRO) to assume an uncertain parameter follows an unknown distribution of which only some characteristics are specified. In this paper, similar to PERT, we consider distributions for the activity duration with a known bounded support $\text{supp}(d_a) \subseteq [l_a, u_a]$. Moreover, we assume a known mean $\mathbb{E}_{\mathbb{P}}[d_a] = \mu_a$, potentially a known mean absolute deviation from the mean $\mathbb{E}_{\mathbb{P}}|d_a - \mu_a| = \delta_a$ and potentially a known probability to be bigger than the mean $\mathbb{P}(d_a \geq \mu_a) = \beta_a$. We choose to assume knowledge on these properties, as it allows us to use the aforementioned results from Distributionally Robust Optimization on tight upper and lower bounds for the expected project duration from (Ben-Tal and Hochman, 1972).

A further assumption necessary to apply those results is the pairwise independence of the activity durations. We note that we thus do not address this disadvantage of PERT, as it is shared by our method of analysis. All these assumptions are combined in the following three sets of distributions, also known as ambiguity sets:

$$\mathcal{P}_\mu = \left\{\mathbb{P} : \text{supp}(d_a) \subseteq [l_a, u_a],\ \ \mathbb{E}_{\mathbb{P}}[d_a] = \mu_a\ \ \forall a \in A, \quad d_a \perp\!\!\!\perp d_b\ \ \forall a \neq b\right\}, \tag{2}$$

$$\mathcal{P}_{(\mu,\delta)} = \left\{\mathbb{P} \in \mathcal{P}_\mu : \mathbb{E}_{\mathbb{P}}[|d_a - \mu_a|] = \delta_a \quad \forall a \in A\right\}, \tag{3}$$

$$\mathcal{P}_{(\mu,\delta,\beta)} = \left\{\mathbb{P} \in \mathcal{P}_{(\mu,\delta)} : \mathbb{P}(d_a \geq \mu_a) = \beta_a \quad \forall a \in A\right\}. \tag{4}$$

In general, Ben-Tal and Hochman (1972) state that we can calculate the lowest and highest possible expectation of a convex function f over these ambiguity sets, which relate by:

$$\inf_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right] \leq \inf_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right] \leq \sup_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta)}} \mathbb{E}\left[f(\boldsymbol{d})\right] \leq \sup_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right].$$

When $\delta_a = 2\frac{(u_a-\mu_a)(\mu_a-l_a)}{(u_a-l_a)}$, which is the maximum value the MAD can take given support and mean information, the third inequality is in fact an equality. Moreover, if the critical path is the same for any combination of values in $[l_a, u_a]$, all inequalities are equalities, as the expected project duration is then simply the sum of the expected duration of the activities on this single critical path.

In order to apply the results by Ben-Tal and Hochman (1972), it is thus necessary for $f(\boldsymbol{d})$ to be convex in $\boldsymbol{d}$. To see that in our setting $f(\boldsymbol{d})$ is convex, it is important to note that by its definition (1), $f(\boldsymbol{d})$ is the (recursive) maximum of several affine functions in $\boldsymbol{d}$. The fact that $f(\boldsymbol{d})$ is convex then follows from the fact that both the sum and maximum of convex functions is once again a convex function.

The best-case project duration over the simple ambiguity set (2), $\inf_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right]$ is equal to the expected project duration the standard PERT approach yields, $f(\boldsymbol{\mu})$. This confirms the criticism that PERT generally yields too optimistic values for the expected project duration.

Lemma 1. *(Jensen's bound)*
PERT's expected project duration $f(\boldsymbol{\mu})$ *equals* $\inf_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right]$.

Furthermore, we can provide a closed form expression for the worst-case expected project duration. The worst-case distribution corresponding to this worst-case expectation is supported on 3^m points, which we denote by

$$\tau_1^a = l_a, \;\; \tau_2^a = \mu_a, \;\; \tau_3^a = u_a,$$

with corresponding probabilities

$$p_1^a = \frac{\delta_a}{2\left(\mu_a - l_a\right)}, \;\; p_2^a = 1 - p_1^a - p_3^a, \;\; p_3^a = \frac{\delta_a}{2\left(u_a - \mu_a\right)},$$

for all $a \in A$. To make this paper self-contained, a proof to Lemma 2 is included in A.

Lemma 2. *For any convex function* $f : \mathbb{R}^m \to \mathbb{R}$*, it holds that*

$$\sup_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right] = \sum_{\boldsymbol{\alpha}\in\{1,2,3\}^m} \prod_{a\in A} p_{\alpha_a}^a f\left((\tau_{\alpha_1}^1, \ldots, \tau_{\alpha_m}^m)^\top\right).$$

Essentially, this result states that the distribution that attains the worst-case expected project duration is a three point distribution on the boundaries of the support, and the mean, resulting in 3^m possible outcomes for $\boldsymbol{d}$ with according probabilities and corresponding critical paths. It is important to remark that all possible combinations of these possibilities are considered, while the standard PERT analysis only considers a single critical path. We thus compute the worst-case expected project duration by determining $f\left((\tau_{\alpha_1}^1, \ldots, \tau_{\alpha_m}^m)^\top\right)$ for all possible values of $\boldsymbol{\alpha}$,

i.e., determining the longest paths in the corresponding deterministic graphs, and multiplying them with the corresponding probability. This results in a computation of total complexity $\mathcal{O}\left(3^{|A|}\left(|A|+|V|\right)\right)$. Observe that β_a is not used in the definition of this distribution, which means that no information on $\mathbb{P}\left(d_a \geq \mu_a\right)$ is necessary to find the worst-case expected project duration. It is however, necessary for finding the best-case expected project duration.

The best-case distribution is supported on 2^m points, denoted by

$$q_1^a = 1-\beta_a, \;\; q_2^a = \beta_a,$$

with corresponding probabilities

$$\nu_1^a = \mu_a - \frac{\delta_a}{2\left(1-\beta_a\right)}, \;\; \nu_2^a = \mu_a + \frac{\delta_a}{2\beta_a},$$

for all $a \in A$. The expression for the best-case expected project duration is given in the following Lemma. To make this paper self-contained, a proof to Lemma 3 is included in A.

Lemma 3. *For any convex function $f : \mathbb{R}^m \to \mathbb{R}$, it holds that*

$$\inf_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right] = \sum_{\boldsymbol{\alpha}\in\{1,2\}^m} \prod_{a=1}^{m} q_{\alpha_a}^a f\left(\left(\nu_{\alpha_1}^1, \ldots, \nu_{\alpha_m}^m\right)^\top\right).$$

This results in a computation of total complexity $\mathcal{O}\left(2^{|A|}\left(|A|+|V|\right)\right)$ for the best-case expected project length.

4 Algorithm

Because of the exponential nature of both the best- and worst-case expected project duration, the expressions in Section 3 seem impossible to compute within a reasonable time for any realistic projects. In practice, this means we can only use these formulas for projects with a very small number of activities. We therefore develop an algorithm that severely diminishes the number of scenarios for which the critical path needs to be computed explicitly.

The algorithm used in this paper is based on the following observation. Given a scenario $\boldsymbol{\alpha} \in \{1,2,3\}^m$ and the critical path C in the corresponding graph with activity durations $\tau_{\alpha_1}^1, \ldots, \tau_{\alpha_m}^m$, the critical path must be the same for many other scenarios $\boldsymbol{\alpha}$. In particular, the critical path is the same for any scenario $\tilde{\boldsymbol{\alpha}}$ that has the same activity duration as $\boldsymbol{\alpha}$ for all activities on the critical path, and at most $\boldsymbol{\alpha}$'s activity duration for all other activities. Formally, these scenarios are given by all $\tilde{\boldsymbol{\alpha}} \in \{1,2,3\}^m$ such that

$$\begin{cases} \tilde{\alpha}_c = \alpha_c & \forall c \in C \\ \tilde{\alpha}_c \leq \alpha_c & \forall c \notin C. \end{cases}$$

We will refer to such scenarios as the induced scenarios by a pair $(\boldsymbol{\alpha}, C)$. For all induced scenarios, we know that the critical path for both scenarios must be the same, because $\tau_1^a \leq \tau_2^a \leq \tau_3^a$ for any $a \in A$. Hence, it also holds that $f\left(\tau_{\tilde{\alpha}_1}^1, \ldots, \tau_{\tilde{\alpha}_m}^m\right) = f\left(\tau_{\alpha_1}^1, \ldots, \tau_{\alpha_m}^m\right)$. This

Algorithm 1

Set $V = \emptyset$
Set $L = \{(2, \ldots, 2)\}$
Set $i = 1$
while $L \neq \emptyset$ **do**
    Choose $\boldsymbol{\alpha} \in L$ and remove it from L
    Compute $\lambda_i = f(\nu^1_{\alpha_1}, \ldots, \nu^m_{\alpha_m})$ and let $C \subseteq A$ be the longest path of length l
    Compute the total probability $\gamma_i = \prod_{c \in C} p^c_{\alpha_c} \cdot \textsc{remaining}((\boldsymbol{\alpha}, C), V)$
    Add $(\boldsymbol{\alpha}, C)$ to V
    for all $a \in C$ such that $\alpha_a > 1$ **do**
        Define $\tilde{\boldsymbol{\alpha}}$ by $\tilde{\alpha}_a = \alpha_a - 1$ and $\tilde{\alpha_c} = \alpha_c$ for all $c \neq a$
        if $\tilde{\boldsymbol{\alpha}} \notin L$ & $\nexists C$ such that $(\boldsymbol{\alpha}, C) \in V$ **then**
            Add $\tilde{\boldsymbol{\alpha}}$ to L
        end if
    end for
    Increase i by 1
end while
The worst-case expected project length is given by $\gamma^\top \lambda$.

observation means that for many scenarios, the critical path will not have to be computed as it has implicitly been determined while evaluating a different scenario. Based on this observation, we develop an algorithm that iteratively computes the critical path for a scenario for which it is still unknown and subsequently identifies all scenarios $\tilde{\boldsymbol{\alpha}}$ for which the critical path is guaranteed to be the same. These scenarios then do not have to be considered any further.

A basic implementation of this idea would be to store the critical path for every scenario and iterate over all scenarios for which it remains unknown. This approach, however, quickly runs into severe memory limitations, as for any graph with 21 or more activities, the number of stored scenarios would exceed the number of bytes available in a computer with 4GB RAM.

Instead, the algorithm we propose stores the scenarios for which the critical path is already known in a more intelligent way, enabling the algorithm to deal with larger problems. The pseudocode for this algorithm is presented in Algorithm 1. For simplicity's sake, the algorithm is presented for computing the expected project duration for a distribution supported on 2^m points, instead of 3^m. In this algorithm, REMAINING is a subroutine that computes the total probability of the scenarios induced by the current scenario and critical path couple $(\boldsymbol{\alpha}, C)$ and subtracts from that the probability of the induced scenarios whose critical path was already known. For more details on this subroutine, we refer the interested reader to B.

The algorithm starts by considering the scenario in which all activity durations are at their maximum possible value. For each scenario $\boldsymbol{\alpha}$ it considers, the critical path C in the corresponding graph is determined. Based on this critical path and the key observation the algorithm relies upon, the induced set of scenarios for which the critical path and its length are the same can be found. Some of these scenarios have been induced by a scenario processed earlier. Using the REMAINING subroutine, the total probability of all scenarios that have not been induced before

is calculated. Subsequently, a new scenario $\tilde{\boldsymbol{\alpha}}$ is created for each arc on the longest path that is not already set to its lowest possible value. These new scenarios are identical to $\boldsymbol{\alpha}$ for all but one activity, for which the duration is lowered. This guarantees that every possible scenario will either be considered or induced at some point during the algorithm. These new scenarios are added to a list L, from which the next scenario is chosen. Once L is empty, i.e., no more scenarios exist for which the critical path is unknown, the algorithm terminates.

While the algorithm we develop still has a worst-case computational complexity of $\mathcal{O}\left(3^{|A|}\left(|A|+|V|\right)\right)$, it allows us to compute the best- and worst-case expected project duration for projects with up to 40 activities on a critical path within reasonable time. Dependent on project structure, this potentially means projects with up to 120 activities could be processed.

5 Approximation of the Worst- and Best-Case Bounds

While the algorithm described in Section 4 is a huge improvement over evaluating the expressions from Section 3, it is still somewhat limited in the size of projects that can be tackled. In this section we propose adaptations to our approach that will decrease the quality of the obtained bound on the expected project duration while increasing the computational tractability. For ease of exposition we will focus on upper bounds for the worst-case expected project duration in this section. The second and third proposal in this section can easily be adapted to yield lower bounds as well and bounds for the best-case expected project duration follow from these lower bounds.

First of all, we remark that any partial completion of the algorithm can lead to an upper bound on the worst-case project duration, given that the scenarios are processed such that the corresponding project durations are in decreasing order. A disadvantage of this approach is the quality of the resulting bound, as it is particularly bad for low amounts of computational time. An advantage, on the other hand, is that it is very easy to implement and yields a bound for any specified maximum computation time.

Two other possible upper bounds can be obtained by altering characteristics of the project such that the worst-case expected project duration increases. In particular, we consider two operations that decrease the number of activities with uncertainty and thus lessen the computational burden. The easiest way to do this is to remove uncertainty from an activity by setting $d_a = u_a$ for some $a \in A$. The quality of the resulting bound is dependent on the activity chosen to perform this operation on. In general, it is attractive to select those activities that had low impact on the expected project duration to start with. This impact results from an activity duration having high variability, i.e., a high mean absolute deviation, and having a long duration in general. Therefore, we select the activities whose duration have the lowest mean absolute deviation.

The second operation we consider is the merger of two activities. More specifically, we consider the replacement of two activities that are each other's unique successor and predecessor by a single activity. Mathematically, this means we replace activities $(i,j),(j,k) \in A$ by a single

activity (i,k). By linearity of the expectation we know that

$$\operatorname{supp}\left(d_{(i,k)}\right) = \left[l_{(i,j)} + l_{(j,k)},\, u_{(i,j)} + u_{(j,k)}\right], \quad \mathbb{E}\left[d_{(i,k)}\right] = \mu_{(i,j)} + \mu_{(j,k)}.$$

Unfortunately, we cannot directly compute the mean absolute deviation of the newly introduced activity, and therefore the worst-case expected project duration after the merger cannot be calculated exactly. We do know, however, that the worst-case expected project duration is increasing in the mean absolute deviation of all activities (Postek et al., 2018). Therefore, any upper bound on $\mathbb{E}\left[\left|d_{(i,k)} - \mu_{(i,j)} - \mu_{(j,k)}\right|\right]$ will yield an upper bound on the original worst-case project duration.

Postek et al. (2018) suggest a number of different techniques to obtain such an upper bound. Here, we use proposition 5 from their work, that exploits the observation that the absolute value is a convex function and the mean absolute deviation can thus be bounded by the same theory used for the expected project duration. This means we consider $3^2 = 9$ different scenarios for $\left(d_{(i,j)}, d_{(j,k)}\right)$ to find a fairly tight upper bound on $\mathbb{E}\left[\left|d_{(i,k)} - \mu_{(i,j)} - \mu_{(j,k)}\right|\right]$. Similar to the first operation, we choose to merge those activities that have the lowest mean absolute deviation to stay as close to the true worst-case expected project duration as possible. We investigate the quality of the bounds resulting from these operations numerically in Section 6.

We note that for the best-case expected project duration we require a lower bound on $\mathbb{E}\left[\left|d_{(i,k)} - \mu_{(i,j)} - \mu_{(j,k)}\right|\right]$. For this, we can similarly use the idea that the mean absolute deviation is a convex function to bound it from below by considering $2^2 = 4$ different scenarios for $\left(d_{(i,j)}, d_{(j,k)}\right)$.

6 Numerical Experiments

6.1 Illustrative Examples

For illustrative purposes, we first consider two examples commonly considered in the literature. More specifically, we consider the two projects analyzed by Birge and Maddox (1995). Figure 1 and 2 graphically depict the projects, Table 1 and 3 contain information on all activities and Table 2 and 4 summarize the results from Birge and Maddox (1995) as well as our approach. Since the data for these examples only include information on the first two moments of the distribution of all activity durations, we consider two different possible values for the mean absolute deviation: the minimum and maximum possible value given the variance, given by $\frac{2\sigma^2}{u-l}$ and σ, respectively (Postek et al., 2018).

These two examples perfectly illustrate the quality of the bound on the expected project duration we obtain. In example 1, where previous known bounds determine the expected project duration exactly, irrespective of the true distribution, we find the exact same value. We note that in this example there is only a single critical path for all scenarios, which is why the best- and worst-case project duration coincide.

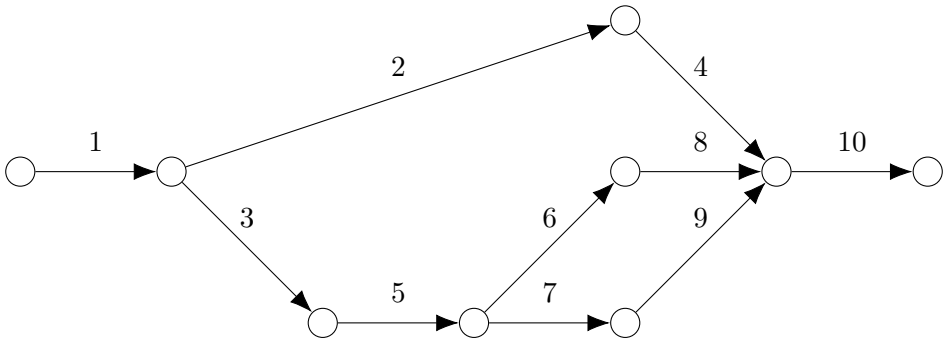


Figure 1: Project network for example 1, labels indicate the activity's index.

Acitivity	Minimum	Maximum	Mean	Variance
1	1	1	1	0
2	2	4	3	$\frac{1}{3}$
3	1	3	2	$\frac{1}{3}$
4	1	5	3	$\frac{4}{3}$
5	2	3	$2\frac{1}{2}$	$\frac{1}{12}$
6	3	7	5	$\frac{4}{3}$
7	3	5	4	$\frac{1}{3}$
8	4	5	$4\frac{1}{2}$	$\frac{1}{12}$
9	1	2	$1\frac{1}{2}$	$\frac{1}{12}$
10	4	6	5	$\frac{1}{3}$

Table 1: Activity duration data for example 1.

In example 2, on the other hand, there are multiple potential critical paths. The lower and upper bounds found by Birge and Maddox (1995) therefore still had a significant gap. Here, we can narrow the gap by a factor three if we know (or have an accurate estimate of) the mean absolute deviation. Moreover, if only the variance is known, we can tighten the bounds to $[3.07, 3.68]$, which is still a little over half the original gap. This empirically shows that the bounds by Birge and Maddox (1995) are not tight. We note that for illustrative purposes, the potential deviation in these examples is unrealistically large (up to 100% of the mean). The relative size of the gap with respect to the value of the expected project duration is therefore disproportionally large.

6.2 PSPLIB Instances

To test our approach on a larger scale, we use the RCPSP problems from the PSPLIB project scheduling library (Kolisch and Sprecher, 1997). This library contains a wealth of problems with $m = 30$, 60, 90 and 120 activities, that need to be adapted slightly to be fully suitable

	BM	$\delta_a = \frac{2\sigma_a^2}{u_e - l_e}$	$\delta_a = \sigma_a$
Upper Bound	20	20	20
Lower Bound	20	20	20

Table 2: Bounds for example 1. BM lists the bounds found by Birge and Maddox (1995), the other two columns list the results of our method for the listed values of the mean absolute deviation.

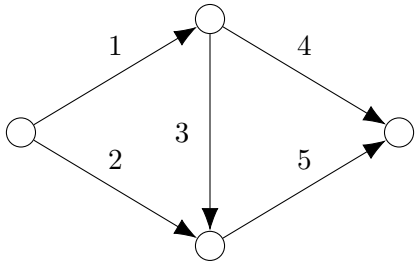


Figure 2: Project network for example 2, labels indicate the activity's index.

Acitivity	Minimum	Maximum	Mean	Variance
1	0	2	1	$\frac{2}{3}$
2	0	2	1	$\frac{2}{3}$
3	0	2	1	$\frac{2}{3}$
4	0	2	1	$\frac{2}{3}$
5	0	2	1	$\frac{2}{3}$

Table 3: Activity duration data for example 2.

	BM	$\delta_a = \frac{2\sigma_a^2}{u_e - l_e}$	$\delta_a = \sigma_a$
Upper Bound	4	3.36	3.68
Lower Bound	3	3.07	3.31

Table 4: Bounds for example 2. BM lists the bounds found by Birge and Maddox (1995), the other two columns list the results of our method for the listed values of the mean absolute deviation.

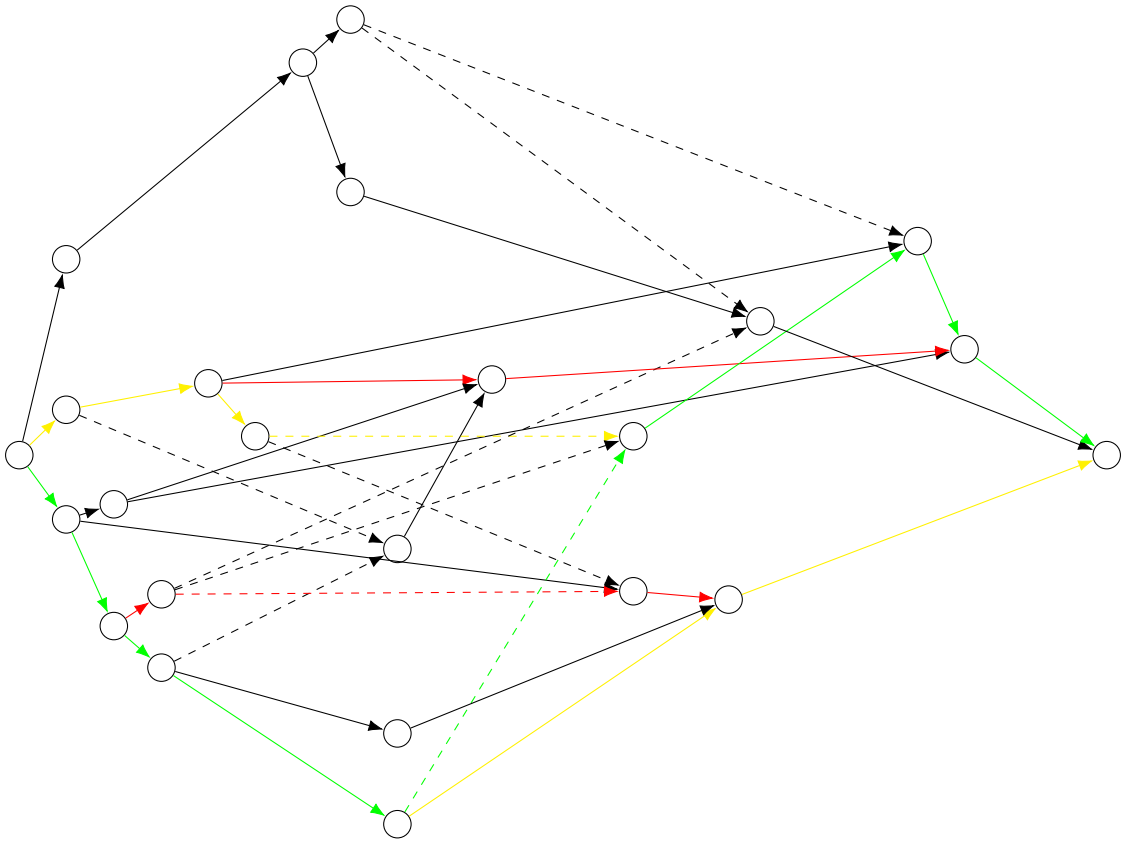


Figure 3: A visualization of `j301_2.sm`. Solid arcs correspond to activities, dashed arcs are auxiliary arcs that model additionally precedence relations. Green, orange and red arcs correspond to activities on the critical path for PERT, support I and support II, respectively.

for our method. First and foremost, the problems are modeled as an activity on nodes (AoN) network. To transform the projects into the desirable activity on arc (AoA) format, we employ the algorithm in (Sterboul and Wertheimer, 1981) as explained by Mouhoub et al. (2011).

Moreover, the instances from PSPLIB only provide a nominal value for the activity duration. We assume this nominal value is the mean ($\boldsymbol{\mu}$) and that the support is given by either (I) 95% and 115% or (II) 85% and 145% of this value. An instance from the PSPLIB library is visualized in Figure 3. Here, solid arcs correspond to activities, while dashed arcs are auxiliary arcs that model additional precedence relations that result from the reformulation to the AoA format. The critical path found by PERT is shown in green. Additional arcs that are potentially on a critical path are shown in orange (I) and red (II). We remark that any arc that is on a critical path for the narrow support (I) is also on a critical path for the wider support (II).

Because we are comparing our technique to PERT, we choose to assume the mean absolute deviation ($\boldsymbol{\delta}$) and probability that the duration is larger than the mean ($\boldsymbol{\beta}$) are exactly the values PERT assumes them to be, that is, values from the beta distribution PERT assumes. All numerical results presented have been obtained using Python 3.7 on a Lenovo Y700 with an i7-6700HQ processor and 16GB RAM.

As we present many results later in this section based on the bounds discussed in Section 5,

we first present some numerical results regarding the quality of these bounds. In particular, the average results on removing uncertainty from activities and merging activities over all 480 instances with 30 activities are shown in Figure 4. These figures both show two sets of boxplots. On the x-axis of all these figures, the number of activities from which the uncertainty was removed, or the number that were merged is shown. On the y-axis of the figures on the left, the average quality of the resulting bound with respect to the exact value is shown. In the figures of the right, the y-axis shows the average computation time.

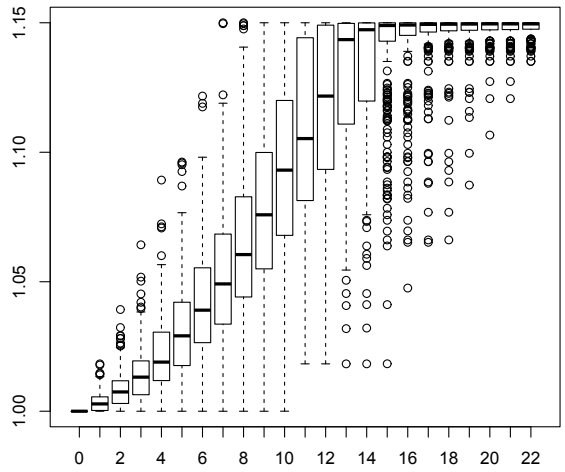
The results for the two bounds are quite different. Removing uncertainty can clearly completely remove the computational burden if all activities are assumed to be certain. Its quality, however, suffers accordingly. When only 2 or 3 activities are assumed to be certain the bound can already be up to 5% above the true worst-case expected project duration. It is important here to note that the activities of which the uncertainty was removed were selected to maximize the quality of the resulting bound.

Merging activities, on the other hand, yields bounds that are very close to the actual value. Clearly, the gain in computation time is much smaller for this bound, but it is generally sufficient to tackle all problems with 30 activities and most with 60. We therefore will limit ourselves to only consider this bound moving forward.

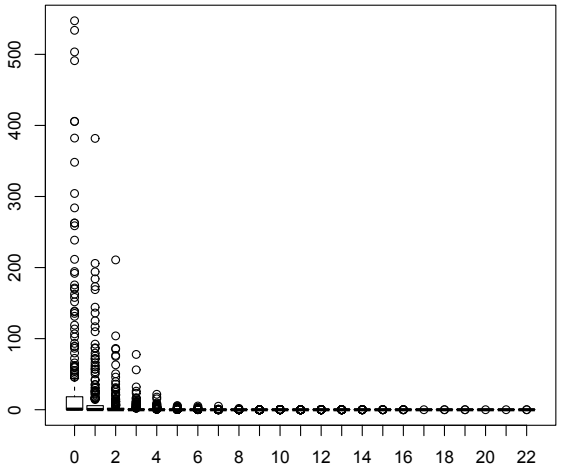
The best- and worst-case expected project duration for all 480 instances with 30 activities are shown for both values of the support (I and II) in Figures 5 and 6. For all instances, the ratios of the presented bounds with respect to the expected project duration found by PERT are reported, such that the best-case expected project duration without mean absolute deviation known always corresponds to 1. The resulting ratios were grouped by and averaged over the instances that have the same number of different critical paths. Figures 5 and 6 show the following four values:

- Best-case expected project duration without mean absolute deviation known (equal to PERT) in blue;
- Best-case expected project duration with mean absolute deviation known in green;
- Worst-case expected project duration with mean absolute deviation known in orange;
- Worst-case expected project duration without mean absolute deviation known in red.

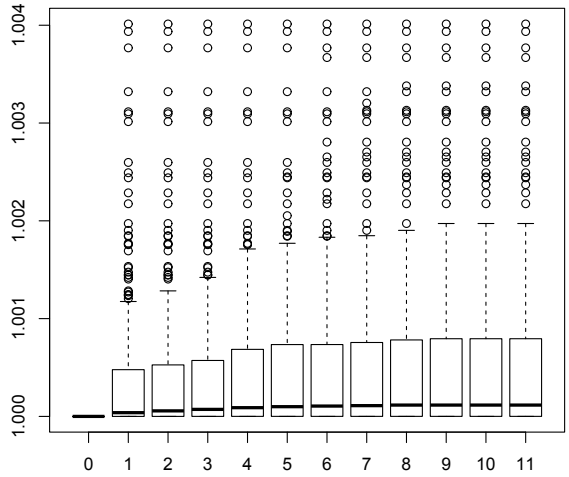
Figure 5 shows that on average, even in the most extreme projects, the difference between the worst- and best-case expected project duration does not exceed 2%. Moreover, when the mean absolute deviation is known, the difference is significantly smaller, which follows from the green and orange dots being significantly closer than the red and blue. In fact, the biggest difference between the worst- and best-case expected project duration when the mean absolute deviation is known is slightly less than 3%. As we allow the activity duration to be up to 15% higher than on average, and the support width is 20% of the mean value, we feel this 3% is rather small. We therefore tentatively conclude that knowing the actual distribution (and mean absolute deviation) has little influence on the expected project duration.



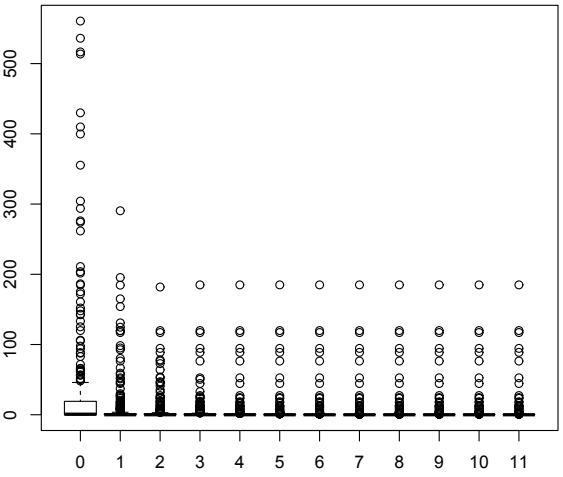
(a) Bound quality for removing uncertainty.



(b) Computation time in seconds for removing uncertainty.



(c) Bound quality for merging activities.



(d) Computation time in seconds for merging activities.

Figure 4: An overview of the quality and computation time of the bounds presented in Section 5.

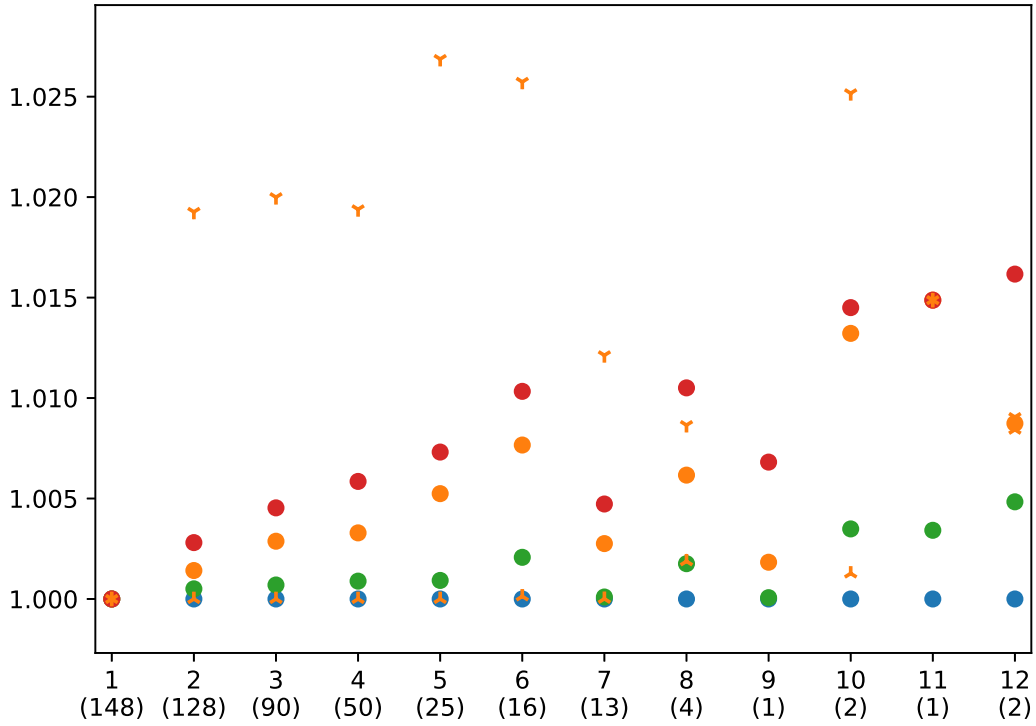


Figure 5: Average ratio to PERT grouped by number of different critical paths for instances with 30 activities with a support from 95% to 115% of the nominal value. Depicted are: PERT in blue, best-case with mean absolute deviation in green, worst-case with mean absolute deviation in orange and worst-case without mean absolute deviation in red. Orange arrows indicate the lowest and highest found value per group of instances for the worst-case with known mean absolute deviation. The number of instances in each group is included in parentheses.

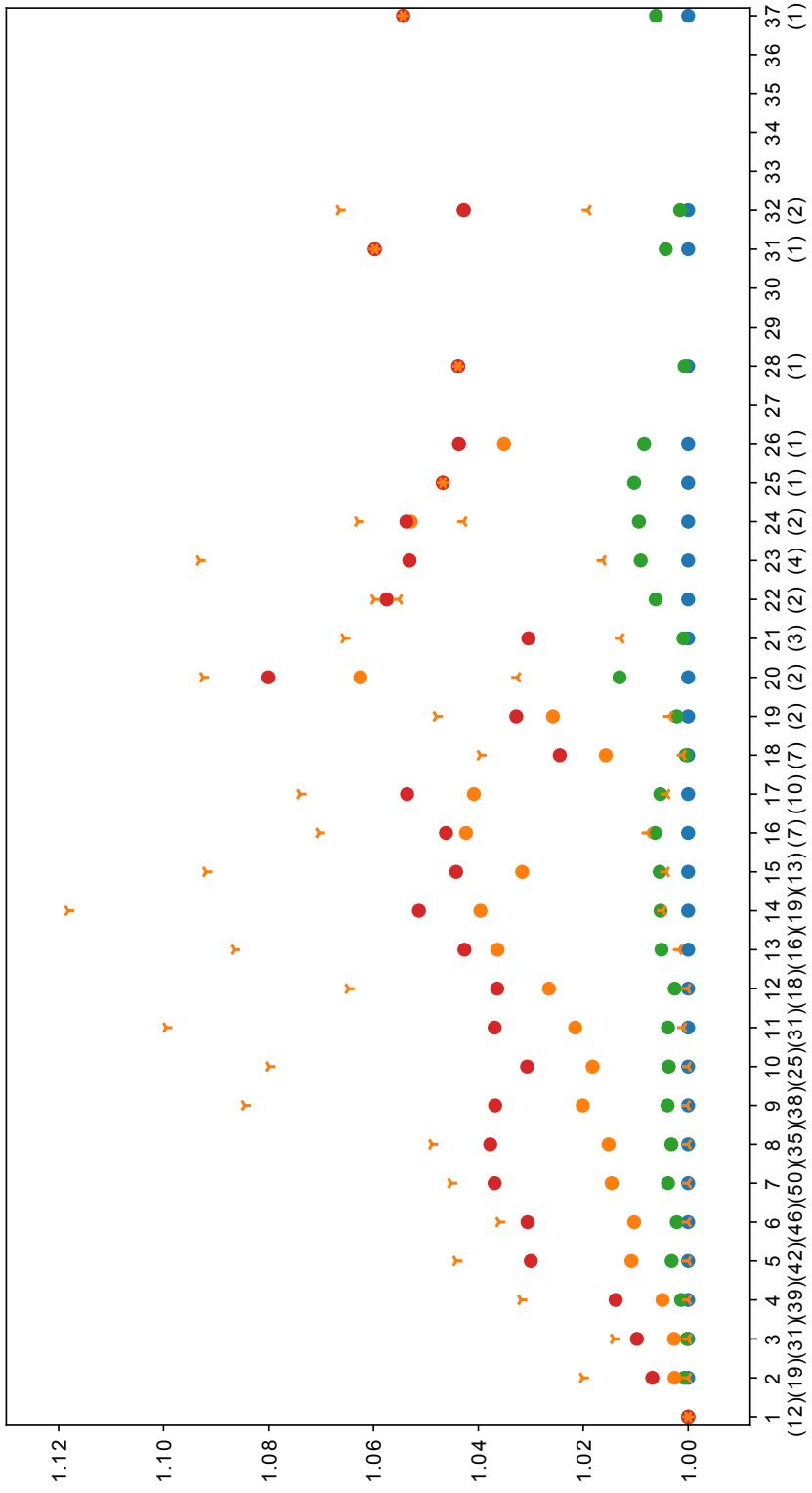


Figure 6: Average ratio to PERT grouped by number of different critical paths for instances with 30 activities with a support from 85% to 145% of the nominal value. Depicted are: PERT in blue, best-case with mean absolute deviation in green, worst-case with mean absolute deviation in orange and worst-case without mean absolute deviation in red. Orange arrows indicate the lowest and highest found value per group of instances for the worst-case with known mean absolute deviation. The number of instances in each group is included in parentheses.

To investigate whether the width of the support significantly changes the expected project duration, Figure 6 shows the results when the activity duration is between 85% and 145% of its mean. As mentioned before, assuming a wider support implies that more activities are potentially on the critical path, and thus the computation time increases as well. Hence, for a fair number of instances the exact worst- and best-case expected project duration could not be calculated within 5 minutes. Therefore, we included a bound on those values based on merging activities for those instances. While the difference to PERT is clearly bigger for a larger support, the change seems mostly proportional to the change in the width of the support. This strengthens our belief that knowledge of the support and mean is much more important than knowledge of the actual distribution or the mean absolute deviation.

D contains similar figures to Figure 5 for projects with 60, 90 and 120 activities and a support from 95% to 115% of the mean: Figures 9, 10 and 11, respectively. For these sizes, we randomly selected 50 instances from PSPLIB in order to prevent the total computation time from becoming entirely unreasonable. While for both 90 and 120 activities there clearly exist some instances for which the difference between worst- and best-case expected project duration is bigger, on average the difference is still fairly small. Moreover, we remark that due to the size of these instances all results presented here have the maximum number of activities merged that was possible. The true difference, therefore, is likely to be somewhat smaller than the figures suggest.

7 Conclusion

In this paper, we develop a general approach to analyze the sensitivity of the Program Evaluation and Review Technique to its assumptions on activity durations following the beta distribution with a specified variance. In contrast to previous research, which compares the use of this beta distribution with a variety of other distributions, we use techniques from distributionally robust optimization that does not require any assumptions on the type of distribution. We only assume some basic distributional information on the activity durations to be known: the support, mean and mean absolute deviation, as well as pairwise independence. This allows us to use the results of Ben-Tal and Hochman (1972) that specify the worst- and best-case distribution, from which we find tight upper and lower bounds on the expected project duration.

Because the approach we propose is computationally intensive, we propose several potential speedups that result in slightly looser bounds. This allows us to analyze PERT's assumptions on a large collection of instances from PSPLIB. Through extensive numerical experiments, we show that the effect of PERT's assumption regarding the beta distribution is limited. Furthermore, we conclude that there is only a limited benefit to knowing the exact mean absolute deviation.

Acknowledgments

The research of the first author was funded by the Netherlands Organisation for Scientific Research (NWO) Research Talent [Grant 406.17.511].

References

Bard, J. F. and Bennett, J. E. (1991). Arc reduction and path preference in stochastic acyclic networks. *Management Science*, 37(2):198–215.

Ben-Tal, A. and Hochman, E. (1972). More bounds on the expectation of a convex function of a random variable. *Journal of Applied Probability*, 9(4):803–812.

Birge, J. R. and Maddox, M. J. (1995). Bounds on expected project tardiness. *Operations Research*, 43(5):838–850.

Demassey, S., Artigues, C., and Michelon, P. (2005). Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, 17(1):52–65.

Hahn, E. D. (2008). Mixture densities for project management activity times: A robust approach to PERT. *European Journal of Operational Research*, 188(2):450–459.

Hajdu, M. (2013). Effects of the application of activity calendars on the distribution of project duration in PERT networks. *Automation in Construction*, 35:397–404.

Johnson, D. (1997). The triangular distribution as a proxy for the beta distribution in risk analysis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 46(3):387–398.

Klein Haneveld, W. K. (1986). Robustness against dependence in PERT: An application of duality and distributions with known marginals. In *Stochastic Programming 84 Part I*, pages 153–182. Springer.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1):205–216.

Kotiah, T. and Wallace, N. D. (1973). Another look at the PERT assumptions. *Management Science*, 20(1):44–49.

Littlefield Jr, T. and Randolph, P. (1987). Reply - an answer to Sasieni's question on PERT times. *Management Science*, 33(10):1357–1359.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669.

Mouhoub, N. E., Benhocine, A., and Belouadah, H. (2011). A new method for constructing a minimal PERT network. *Applied Mathematical Modelling*, 35(9):4575–4588.

Postek, K., Ben-Tal, A., Den Hertog, D., and Melenberg, B. (2018). Robust optimization with ambiguous stochastic constraints under mean and dispersion information. *Operations Research*, 66(3):814–833.

Rahimian, H. and Mehrotra, S. (2019). Distributionally robust optimization: A review.

Reich, D. and Lopes, L. (2011). Preprocessing stochastic shortest-path problems with application to PERT activity networks. *INFORMS Journal on Computing*, 23(3):460–469.

Sterboul, F. and Wertheimer, D. (1981). Comment construire un graphe PERT minimal. *RAIRO-Operations Research*, 15(1):85–98.

Van Eekelen, W., Den Hertog, D., and Van Leeuwaarden, J. S. (2020). MAD dispersion measure makes extremal queue analysis simple.

Wiesemann, W., Kuhn, D., and Sim, M. (2014). Distributionally robust convex optimization. *Operations Research*, 62(6):1358–1376.

Zhu, G., Bard, J. F., and Yu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, 18(3):377–390.

A Proofs

The following proof is taken from Van Eekelen et al. (2020, Theorem 1).

Proof of Lemma 2. First of all, note that since $d_a \perp\!\!\!\perp d_b$ for all $a \neq b$, we can write

$$\sup_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right] = \sup_{\mathbb{P}^1\in\mathcal{P}^1_{(\mu,\delta,\beta)}} \cdots \sup_{\mathbb{P}^m\in\mathcal{P}^m_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right], \tag{5}$$

where

$$\mathcal{P}^a_{(\mu,\delta,\beta)} = \{\mathbb{P} : \operatorname{supp}(d_a) \subseteq [l_a, u_a],\ \ \mathbb{E}_{\mathbb{P}}\left[d_a\right] = \mu_a \mathbb{E}_{\mathbb{P}}\left[|d_a - \mu_a|\right] = \delta_a\},$$

is the ambiguity set containing the marginal distribution of d_a for $a = 1, \ldots, m$. For any a, we consider the problem

$$\sup_{\mathbb{P}^a\in\mathcal{P}^a_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right]. \tag{6}$$

For the sake of brevity and simplicity, we consider all d_b for $b \neq a$ fixed for now, such that f is only a function of d_a. With a slight abuse of notation, we will refer to $f(d_a)$ instead of $f(\boldsymbol{d})$. We can write (6) as the following semi-infinite linear program:

$$\begin{aligned}
\sup_{\mathbb{P}} \quad & \int_{d_a} f(d_a)\,\mathrm{d}\mathbb{P}(d_a) && \text{(7a)}\\
\text{s.t.} \quad & \int_{d_a} \mathrm{d}\mathbb{P}(d_a) = 1 && \text{(7b)}\\
& \int_{d_a} d_a\,\mathrm{d}\mathbb{P}(d_a) = \mu_a && \text{(7c)}\\
& \int_{d_a} |d_a - \mu_a|\,\mathrm{d}\mathbb{P}(d_a) = \delta_a. && \text{(7d)}
\end{aligned}$$

Consider the dual of (7),

$$\begin{aligned}
\min_{\lambda_1,\lambda_2,\lambda_3} \quad & \lambda_1 + \lambda_2\mu_a + \lambda_3\delta_a && \text{(8a)}\\
\text{s.t.} \quad & f(d_a) - \lambda_1 - \lambda_2\mu_a - \lambda_3|d_a - \mu_a| \ \leq 0 \qquad \forall d_a \in [l_a, u_a]. && \text{(8b)}
\end{aligned}$$

Define $F(d_a) = \lambda_1 + \lambda_2\mu + \lambda_3|d_a - \mu_a|$ such that (8b) can be written as $f(d_a) \leq F(d_a)\ \forall d_a \in [l_a, u_a]$, that is, $F(d_a)$ majorizes $f(d_a)$. Since the dual problem (8) has three variables, the tightest majorant F

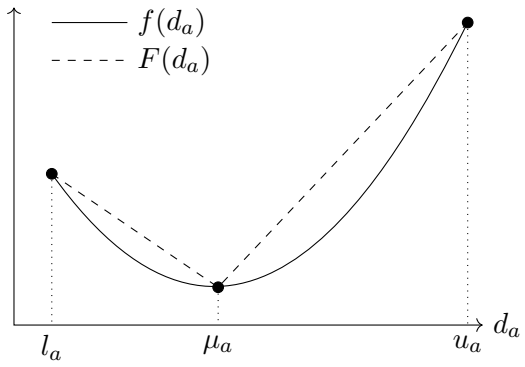


Figure 7: A convex function $f(d_a)$ and its piecewise linear majorant $F(d_a)$.

touches $f(d_a)$ at three points: $d_a = \mathbb{1}_a$, μ_a and u_a, which we denote by τ_1^a, τ_2^a and τ_3^a. This is illustrated in Figure 7. The optimal probabilities can now be easily found by solving the linear system with three unknowns that is obtained when limiting the probability distribution in (7) to be only supported on τ_1^a, τ_2^a and τ_3^a.

If we apply this reasoning for d_m in (5), we find it is equal to

$$\sup_{\mathbb{P}^1 \in \mathcal{P}^1_{(\mu,\delta,\beta)}} \cdots \sup_{\mathbb{P}^{m-1} \in \mathcal{P}^{m-1}_{(\mu,\delta,\beta)}} \mathbb{E}\left[\sum_{\alpha_m \in \{1,2,3\}} p^m_{\alpha_m} f\left((d_1, \ldots, d_{m-1}, \tau^m_{\alpha_m})^\top \right) \right].$$

Since all probabilities are nonnegative, the inner maximization problem once again concerns the maximum expectation of a convex function, and we can apply the reasoning above. Applying that reasoning m times gives the final expression given in Lemma 2 and completes the proof. □

The following proof is taken from Van Eekelen et al. (2020, Theorem 3).

Proof of Lemma 3. Similar to the proof of Lemma 2, we can focus on the univariate setting:

$$\inf_{\mathbb{P}^a \in \mathcal{P}^a_{(\mu,\delta,\beta)}} \mathbb{E}[f(\boldsymbol{d})]. \tag{9}$$

For the sake of brevity and simplicity, we consider all d_b for $b \neq a$ fixed for now, such that f is only a function of d_a. With a slight abuse of notation, we will refer to $f(d_a)$ instead of $f(\boldsymbol{d})$. The optimization problem (9) can be written as the following semi-infinite linear program:

$$\min_{\mathbb{P}} \quad \int_{d_a} f(d_a)\, \mathrm{d}\mathbb{P}(d_a) \tag{10a}$$

$$\text{s.t.} \quad \int_{d_a} \mathrm{d}\mathbb{P}(d_a) = 1 \tag{10b}$$

$$\int_{d_a} d_a\, \mathrm{d}\mathbb{P}(d_a) = \mu_a \tag{10c}$$

$$\int_{d_a} |d_a - \mu_a|\, \mathrm{d}\mathbb{P}(d_a) = \delta_a \tag{10d}$$

$$\int_{d_a} \mathbb{1}_{\{d_a \geq \mu_a\}}\, \mathrm{d}\mathbb{P}(d_a) = \beta_a. \tag{10e}$$

The dual of (10) is given by

$$\max_{\lambda_1,\lambda_2,\lambda_3,\lambda_4} \quad \lambda_1 + \lambda_2\mu_a + \lambda_3\delta_a + \lambda_4\beta_a \tag{11a}$$

$$\text{s.t.} \quad f(d_a) - \lambda_1 - \lambda_2\mu_a - \lambda_3|d_a - \mu_a| - \lambda_4\mathbb{1}_{d_a \geq \mu_a} \geq 0 \qquad \forall d_a \in [l_a, u_a]. \tag{11b}$$

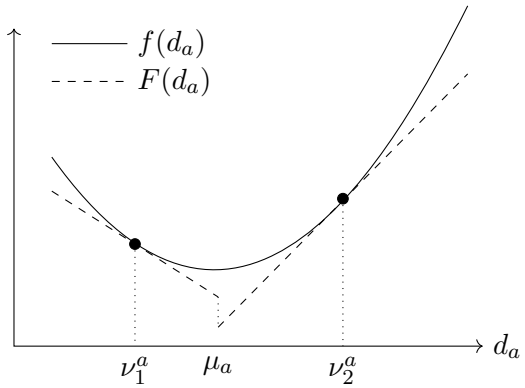


Figure 8: A convex function $f(d_a)$ and its non-continuous piecewise linear minorant $F(d_a)$.

We define $F(d_a) = \lambda_1 + \lambda_2\mu_a + \lambda_3|d_a - \mu_a| + \lambda_4 \mathbb{1}_{d_a \geq \mu_a}$, such that (11b) can be written as $F(d_a) \leq f(d_a)$, i.e., $F(d_a)$ minorizes $f(d_a)$. In this situation, $F(d_a)$ has a kink and a discontinuity at $d_a = \mu_a$, as illustrated in Figure 8. The dual problem thus boils down to finding the tightest minorant that maximizes the objective (11a). From the supporting hyperplane theorem, we know that $F(d_a)$ touches the epigraph of $f(d_a)$ at two or fewer points, on opposite sides of μ_a, i.e., some x_1 and x_2 such that $x_1 \leq \mu_a \leq x_2$. If we insert this knowledge in the dual, we find

$$\begin{aligned} \max_{\lambda_1,\lambda_2,\lambda_3,\lambda_4} \quad & \lambda_1 + \lambda_2\mu_a + \lambda_3\delta_a + \lambda_4\beta_a \\ \text{s.t.} \quad & \lambda_1 + \lambda_2 x_1 + \lambda_3(\mu_a - x_1) + \lambda_4 = f(x_1) \\ & \lambda_1 + \lambda_2 x_2 + \lambda_3(x_2 - \mu_a) = f(x_2). \end{aligned}$$

Using Lagrangean duality, one can show the optimal solution satisfies

$$x_1 = \mu_a - \frac{\delta_a}{2(1-\beta_a)}, \qquad x_2 = \mu_a + \frac{\delta_a}{2\beta_a},$$

which correspond to ν_1^a and ν_2^a. Substituting these values and solving for the dual variables $\lambda_1, \ldots, \lambda_4$, we find the optimal objective is given by $(1-\beta_a)f(\nu_1^a) + \beta f(\nu_2^a)$, which concludes the proof.

□

B Algorithm Details

For ease of exposure, the algorithmic details given in Algorithm 2 and 3 concern two possible values for each activity, as would be the case when calculating the best-case project duration. With some minor modifications the algorithm can also be applied when an activity can take three or more values.

Algorithm 2 REMAINING subroutine

function REMAINING $((\boldsymbol{\alpha}, C), V)$
  Define $V_{(\boldsymbol{\alpha},C)} = \left\{\bar{\boldsymbol{\alpha}} : \exists(\bar{\boldsymbol{\alpha}}, \bar{C}) \in V \text{ for which } \bar{C} = C,\ \ \bar{\alpha}_c = \alpha_c\ \ \forall c \in C\right\}$
  if $V_{(\boldsymbol{\alpha},C)} = \emptyset$ **then**
    Set $\omega = \prod_{c \notin C} \sum_{i=1}^{\alpha_c} p_i^c$
  else
    Set $\omega = 0$
    for all $c \notin C$ such that $\alpha_c > \bar{\alpha}_c \quad \forall \bar{\boldsymbol{\alpha}} \in V_{(\boldsymbol{\alpha},C)}$ **do**
      Update $\omega = \omega + \left[\sum_{i=\max_{\bar{\boldsymbol{\alpha}} \in V_{(\boldsymbol{\alpha},C)}} \{\bar{\alpha}_c\}+1}^{\alpha_c} p_i^c\right] \cdot \prod_{\chi \notin C \cup \{c\}} \sum_{i=1}^{\alpha_\chi} p_i^\chi$
      Update $\alpha_c = \max_{\bar{\boldsymbol{\alpha}} \in V_{(\boldsymbol{\alpha},C)}} \bar{\alpha}_c$
    end for
    Update $\omega = \omega +$ RECURSIVE$((\boldsymbol{\alpha}, C), V_{(\boldsymbol{\alpha},C)}, \emptyset)$
  end if
end function

The REMAINING subroutine calculates the total probability of all scenarios induced by $(\boldsymbol{\alpha}, C)$ that were not previously considered, i.e., were already induced by an element of V. First, it selects the scenarios with the same critical path and the same activity durations on this critical path from V, as those are the only scenarios that could have induced the same scenarios. If such scenarios do not exist, all scenarios induced by $(\boldsymbol{\alpha}, C)$ are new and calculation of the total probability is easy. If such scenarios do exist, on the other hand, the calculation is split into two parts. First of all, we check if any activity c that is not on C has a longer duration in $\boldsymbol{\alpha}$ than in any previously considered scenario. For all such activities c, we can be sure that any scenarios induced by $(\boldsymbol{\alpha}, C)$ for which the value on c is equal are new scenarios and can thus be included in the total probability. For the second part of the calculation, all activity durations are reduced to the maximum value that was considered before this iteration and the recursive subroutine RECURSIVE is called to calculate the remaining probability.

First and foremost, RECURSIVE checks whether there exists a pair $(\bar{\boldsymbol{\alpha}}, \bar{C}) \in V$ such that $\bar{\alpha}_c \geq \alpha_c$ for all $c \in C$, that is, whether it is dominated by any scenario processed before. In this case, (α, C) does not induce any scenario we have not encountered before and thus the remaining probability is 0. Otherwise, we find all activities i such that we have considered scenarios before, which had higher values on every activity but i, and add those activities to a set F. We are only interested in scenarios for which the value of these activities is fixed, as a lower duration on this activity would mean it has been induced previously. After this, we can reduce the set of relevant previous scenarios in V to all scenarios such that their value exceeds the value of α on all activities in F, as those are fixed. This reduced set is denoted by $\hat{V}$. If only one or none of such scenarios exist, calculating the total probability of all new induced scenarios is fairly straightforward. If, on the other hand, more than one scenario reside in $\hat{V}$, we consider two cases for which we compute the total probability of induced scenarios recursively. More specifically, we choose an activity that is not fixed and not at its lowest possible value and consider the case where this activity is fixed and the case where its value is lower.

C Project Preprocessing

Besides intelligently calculating the project duration for relevant scenarios, another observation is key in efficiently evaluating the worst-case expected project duration. If there exists a node $k \in \{1, \ldots, n\}$ such that for each scenario $\boldsymbol{\alpha}$ the critical path in the corresponding project contains k, the expected project

Algorithm 3 RECURSIVE subroutine

function RECURSIVE $((\boldsymbol{\alpha}, C), V, F_0)$
  if $\exists \bar{\boldsymbol{\alpha}} \in V$ such that $\bar{\alpha}_c \geq \alpha_c \ \forall c \notin C$ **then**
    return 0
  else
    Set $F = \{i \notin C : \exists \bar{\boldsymbol{\alpha}} \in V \text{ s.t. } \bar{\alpha}_i < \alpha_i,\ \bar{\alpha}_c \geq \alpha_c \ \forall c \notin C \cup F_0 \cup \{i\}\} \cup F_0$
    if $|F| > 0$ **then**
      Set $\hat{V} = \{\bar{\boldsymbol{\alpha}} \in V : \bar{\alpha}_c \geq \alpha_c \ \forall c \in F\}$
    else
      Set $\hat{V} = V$
    end if
    if $|\hat{V}| \leq 1$ **then**
      if $|\hat{V}| = 0$ **then**
        return $\prod_{c \in F} p^c_{\alpha_c} \cdot \prod_{c \notin C \cup F} \sum_{i=1}^{\alpha_c} p^c_i$
      else
        Denote the only element of F by χ
        Set $B = \{c \notin C : \chi_c < \alpha_c\}$
        return $\prod_{c \in F} p^c_{\alpha_c} \cdot \left[\prod_{c \in B} \sum_{i=1}^{\alpha_c} p^c_i - \prod_{c \in B} \sum_{i=1}^{\chi_c} p^c_i\right] \cdot$
        $\prod_{c \notin C \cup F \cup B} \sum_{i=1}^{\alpha_c} p^c_i$
      end if
    else
      Find an arc j such that $j \notin F$ and $\alpha_j > 0$
      Define $F_1 = F \cup \{j\}$ and $V_1 = \{\bar{\boldsymbol{\alpha}} \in V : \bar{\alpha}_c \geq \alpha_c \ \forall c \in F_1\}$
      Define $\hat{\alpha}$ by $\hat{\alpha}_j = \alpha_j - 1$ and $\hat{\alpha}_c = \alpha_c$ for all $c \neq j$
      return RECURSIVE$((\boldsymbol{\alpha}, C), V_1, F_1)$ + RECURSIVE$((\hat{\boldsymbol{\alpha}}, C), \hat{V}, F)$
    end if
  end if
end function

duration is equal to the sum of the expected longest path from 1 to k and the expected longest path from k to n. In other words, if there exists a node that is on any longest path, irrespective of the activity duration, it suffices to find the worst-case longest path from 1 to k and the worst-case longest path from k to n. This observation relies on the assumption that all activity lengths to be independent. We will refer to nodes k with the above property as separating nodes.

Unfortunately, these separating nodes are not necessarily easy to identify. When any path from 1 to n passes through k, that is, removing k from the graph would disconnect it completely, it is clear that k is a separating node. The easiest way to identify all other separating nodes is removing all edges from the graph that are never contained in a longest path from 1 to n. An intuitive, but computationally expensive method to find these edges is to compute $\sup_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right]$. Since the worst-case distribution for this ambiguity set is only a two-point distribution on the boundaries of the support, any edge that is not on the longest path for any of the considered scenarios in this evaluation, will not be on any longest path. Calculating $\sup_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right]$ first can thus yield a significant decrease in computation time of $\sup_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta)}} \mathbb{E}\left[f(\boldsymbol{d})\right]$ and $\inf_{\mathbb{P}\in\mathcal{P}_{(\mu,\delta,\beta)}} \mathbb{E}\left[f(\boldsymbol{d})\right]$.

A more computationally tractable approach to identify separating nodes involves applying the following sufficient but not necessary condition. First, we note that the shortest possible longest path has length $f(\boldsymbol{l})$. Now, for any activity $a \in A$ we find the longest possible path from 1 to n containing this activity. Then, we know that a can never be on the critical path if this length is strictly smaller than $f(\boldsymbol{l})$. We once again stress that although this is a sufficient condition, it is not necessary.

By first checking the sufficient condition outlined above and subsequently computing the values of interest in the order indicated, that is computing $\sup_{\mathbb{P}\in\mathcal{P}_\mu} \mathbb{E}\left[f(\boldsymbol{d})\right]$ first, we can limit the graph to only the relevant edges fairly efficiently, thereby identifying all separating nodes. All results reported in this paper result from an approach in which theses ideas have been implemented.

We acknowledge that more efficient and elaborate techniques for reducing the project of interest exist. As our implementation only serves an illustrative purpose, we chose not to implement those even though we suspect using such techniques could improve the applicability of our techniques even further. We refer the interested reader to, e.g., (Bard and Bennett, 1991; Reich and Lopes, 2011).

D Additional Figures

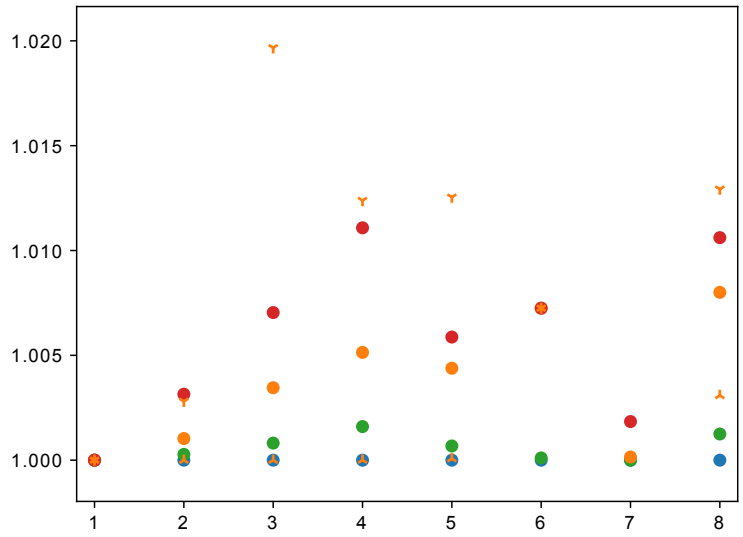


Figure 9: Average deviation from PERT grouped by number of different critical paths for instances with 60 activities. Depicted are: PERT in blue, best-case with mean absolute deviation in green, worst-case with mean absolute deviation in orange and worst-case without mean absolute deviation in red. Orange arrows indicate the minimum and maximum value for the worst-case with known mean absolute deviation. The support is from 95% to 115% of the mean.

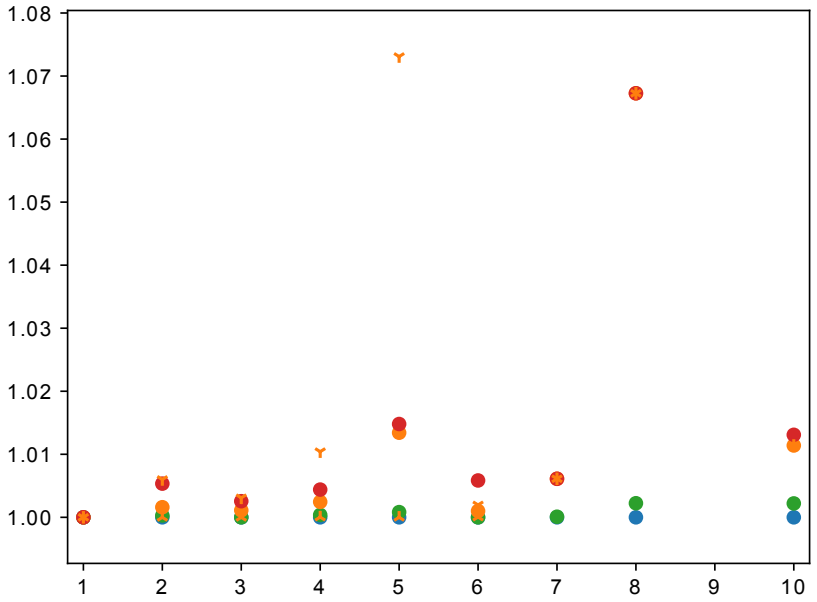


Figure 10: Average deviation from PERT grouped by number of different critical paths for instances with 90 activities. Depicted are: PERT in blue, best-case with mean absolute deviation in green, worst-case with mean absolute deviation in orange and worst-case without mean absolute deviation in red. Orange arrows indicate the minimum and maximum value for the worst-case with known mean absolute deviation. The support is from 95% to 115% of the mean.

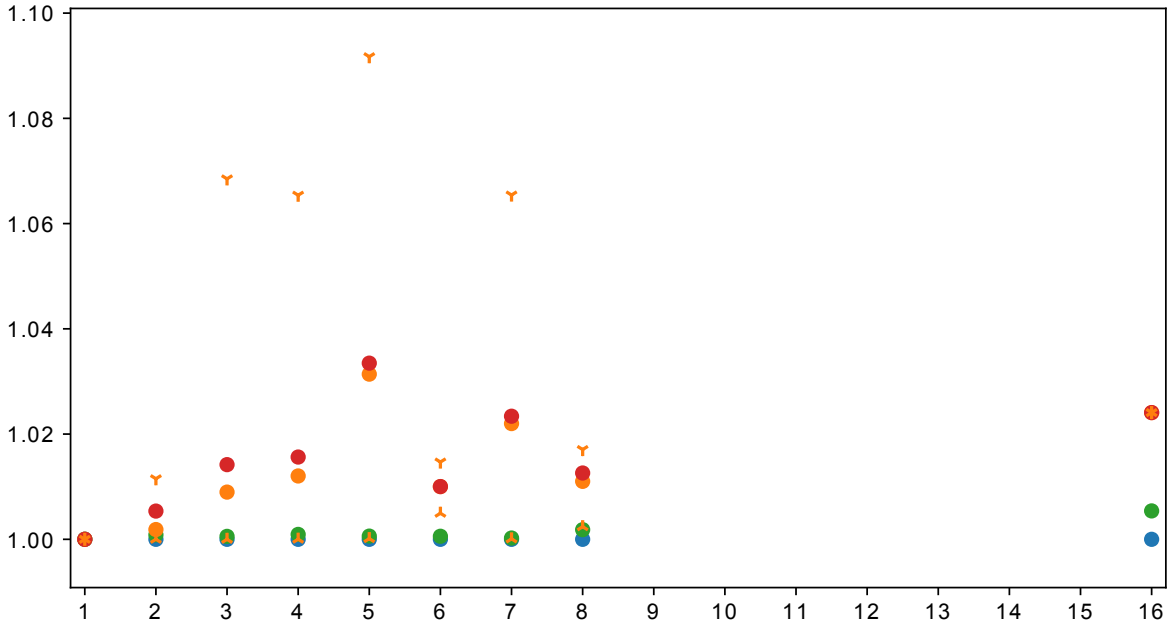


Figure 11: Average deviation from PERT grouped by number of different critical paths for instances with 120 activities. Depicted are: PERT in blue, best-case with mean absolute deviation in green, worst-case with mean absolute deviation in orange and worst-case without mean absolute deviation in red. Orange arrows indicate the minimum and maximum value for the worst-case with known mean absolute deviation. The support is from 95% to 115% of the mean.