Optimising Hidden Stochastic PERT Networks

Tommi Pesu
Imperial College London
Department of Computing, Imperial College London, South Kensington Campus, SW7 2AZ London, United Kingdom
ttp09@imperial.ac.uk

William J. Knottenbelt
Imperial College London
Department of Computing, Imperial College London, South Kensington Campus, SW7 2AZ London, United Kingdom
wjk@imperial.ac.uk

ABSTRACT

This paper introduces a technique for minimising subtask dispersion in hidden stochastic PERT networks. The technique improves on existing research in two ways. Firstly, it enables subtask dispersion reduction in DAG structures, whereas previous techniques have only been applicable to single-layer split–merge or fork–join systems. Secondly, the exact distributions of subtask processing times do not need to be known, so long as there is some means of generating samples. The technique is further extended to use a metric which trades off subtask dispersion and task response time.

Keywords

Stochastic PERT, Subtask dispersion, Task response time

1. INTRODUCTION

Project **E**valuation and **R**eview **T**echnique (PERT) is a widely used scheduling technique in industry [1, 8, 2]. PERT networks are DAG structures which define restrictions on the completion times of activities comprising some overall task. In stochastic PERT, the service times of activities are represented by probability distributions instead of numerical constants. *Hidden* in our context indicates that the user does not know the underlying structure of the PERT network.

In parallel processing systems, there are two useful metrics for describing performance: subtask dispersion [9, 7] (difference in time between the subtask that completes first and the subtask that completes last) and task response time [5, 11] (time for all the subtasks of the task to complete). If subtask dispersion and task response time are both important, a trade-off product metric can be used [10].

Minimising subtask dispersion in hidden stochastic PERT networks is useful in scenarios where information is distributed to multiple competing parties and receiving the information before others gives an edge to a party. An example of this happens in online games that require fast reflexes. Players who have a lower than average lag have an advantage, as they are able to react first to in-game events [12]. In this example the underlying routing structure between the computers taking part is a PERT-like network but the user is typically not aware of its structure or the performance characteristics of individual activities (in this case sending a message between two routers). Here the server's *task* is to broadcast a message to the clients. This comprises in turn several *subtasks*, each of which involves the delivery of the message to one of the clients. The server wishes that all the clients receive the information with a low subtask dispersion and a low task response time. The server can control subtask dispersion and task response time by adding delays to the transmission of messages. However, as the server does not have control of the whole network, it can only add delays to a limited set of neighbouring routers.

This paper presents a new technique for minimising subtask dispersion – or a trade-off product metric involving subtask dispersion and task response time – in hidden stochastic PERT networks. The technique approximates subtask dispersion and task response time by simulation for a given set of added delays – a decision made because analytical techniques such as [6] cannot be applied when specifics of the underlying PERT network are unknown. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [4], noted for its ability to cope with noise, is used to minimise the approximation function.

VALUETOOLS 2016, October 25-28, Taormina, Italy

DOI 10.4108/eai.25-10-2016.2267060

2. PRELIMINARIES

2.1 Stochastic PERT

A stochastic PERT network is a DAG that describes a processing network for completing a task. Figure 1 shows an example. An activity is a edge between two nodes. Each sink node in the PERT defines a subtask. A subtask is considered complete once all activities that can be used to reach it from the source node are completed. The task is completed when all activities in the PERT network are completed. The service time of an activity is denoted by the probability distribution f_n. Service of an activity cannot begin before all activities pointing to the node where the activity starts from have been serviced.

2.2 Task Response Time

Task response time measures the time it takes to service all activities. As the user often does not know exact details of the topology or service time distributions of activities in the PERT network, the user is not able to construct a analytical solution to the problem. Therefore in this paper, task response time is calculated via Monte Carlo simulation.

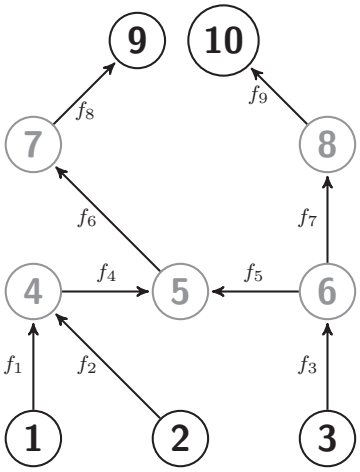

Figure 1: An example of a stochastic PERT DAG

2.3 Subtask Dispersion

For a stochastic PERT network, subtask dispersion is defined as the difference in time between the subtask that was last to complete and the subtask that was first to complete. It is possible to calculate subtask dispersion analytically, if the exact details of the PERT network are known [9, 6].

However, we consider settings where the exact topology and service time distributions are not known. Therefore calculation of subtask dispersion is simulated via Monte Carlo method, as was the case for task response time.

If it is possible to reschedule delays of the network to take into account completion of activities in the PERT network then the technique in this paper has the potential to generate even better results. For an example of how this knowledge can be used to reduce subtask dispersion see [7].

2.4 Trade-off Metric

It is possible to measure the overall performance of the system with a trade-off metric defined as the product of subtask dispersion and task response time [10], i.e.

$$T(\mathbf{d}) = E[D]E[R] \tag{1}$$

Here we extend this with a weight $0 \leq \alpha \leq 1$ which indicates the relative importance of the two metrics:

$$\mathrm{T}(\mathbf{d}, \alpha) = E[\mathrm{D}]^{1-\alpha} E[\mathrm{R}]^{\alpha}, 0 \leq \alpha \leq 1 \tag{2}$$

3. METHOD

The method presented here differs significantly from past techniques used to reduce subtask dispersion. Past works have focused on single-level parallel processing systems and have assumed full knowledge of system parameters to construct analytical functions which are minimised [9, 7, 10]. However, these techniques are manifestly unsuitable for application in the context of hidden PERT networks.

Our new technique not only works on more general DAG structures, but also does not require knowledge of the underlying probability distributions of activity service times. Instead it only needs to be able to randomly sample these. It is assumed that the user has control over a subset of the activities in the system S_c by being able to apply non-negative delays before the processing of activities in that set.

The optimisation technique has two main phases. First, subtask dispersion – for a given set of n subtasks and delays for activities in S_c – is calculated via repeated simulation of t tasks. When t is increased, accuracy increases at the expense of computation effort and *vice versa* when it is reduced. Second, CMA-ES is applied on the approximation function to find optimal delays. CMA-ES was chosen due to it being considered something of a standard in blackbox optimisation [3] which performs better in the context of non-convex optimisation than many classical methods. This is an important feature as the estimates generated via random sampling contain noise.

Where it is desired to trade off subtask dispersion and task response time, we can similarly apply CMA-ES to minimise the penalty function $\mathrm{T}(\mathbf{d}, \alpha)$, i.e.

$$\begin{aligned} &\min_{\mathbf{d} \geq 0} \mathrm{T}(\mathbf{d}, \alpha) \\ &\text{s.t. } \mathbf{d}_i = 0 \quad i \notin S_c \end{aligned} \tag{3}$$

in which the two main components of $T(\mathbf{d})$ are generated as averages of Monte Carlo simulations of t tasks.

4. RESULTS

We apply our technique in the context of the stochastic PERT network of Figure 1. It has three sources $(1, 2, 3)$ and two sinks $(9, 10)$. The user-controlled activities have distributions f_1, f_2 and f_3.

Distributions of the f_i are as follows[1]:

$$f_i = \begin{cases} \text{exponential}(0.2) & \text{i=1} \\ \text{exponential}(0.5) & \text{i=2} \\ \text{normal}(1, 0.5) & \text{i=3} \\ \text{uniform}(0.2, 0.7) & \text{i=4} \\ \text{power}(3) & \text{i=5} \\ \text{normal}(0.5, 1) & \text{i=6} \\ \text{power}(2) & \text{i=7} \\ \text{uniform}(0.75, 0.8) & \text{i=8} \\ \text{normal}(5, 1) & \text{i=9} \end{cases}$$

The durations of the two subtasks T_9 and T_{10} can be derived from the component activities as follows:

$$T_9 = \max(\max(f_1, f_2) + f_4, f_3 + f_5) + f_6 + f_8 \tag{4}$$

$$T_{10} = f_3 + f_7 + f_9 \tag{5}$$

Method 1 Optimised task response time

The results for no delays were calculated using a zero delay vector for 10^7 tasks. Metrics were averaged over 10 runs, which resulted in the following:

Subtask dispersion:	3.300 time units
Task response time:	6.677 time units
Trade-off, $\alpha = 0.5$:	4.694 time units

Method 2 Optimised subtask dispersion.

Here 1 000 samples per measurement point were taken. The CMA-ES algorithm was run on the problem 10 times and the subtask dispersion and task response time for each set of delays was calculated via simulation of 10^7 tasks.

[1]For specifics about each function please refer to Python's `numpy.random` library.

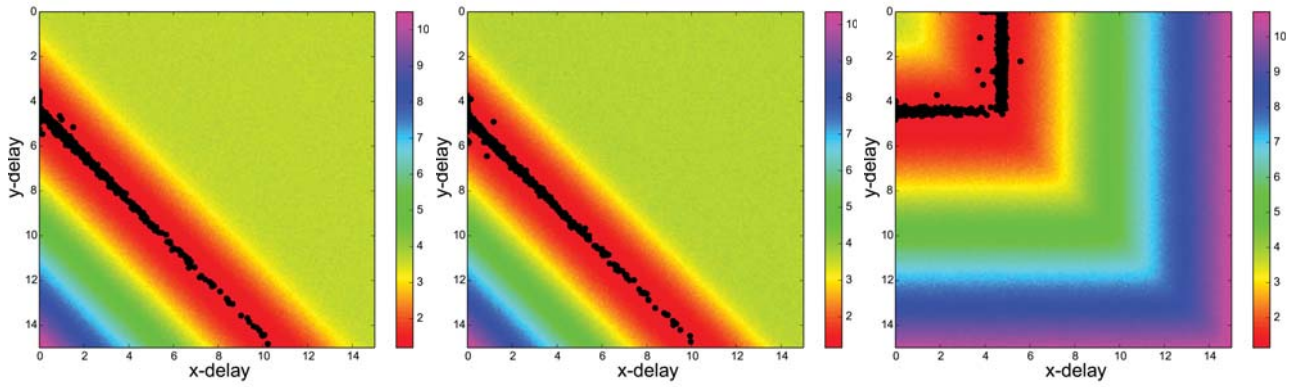

Figure 2: Subtask dispersion with delay vectors $(0, x, y)$ (left), $(x, 0, y)$ (middle) and $(x, y, 0)$ (right)

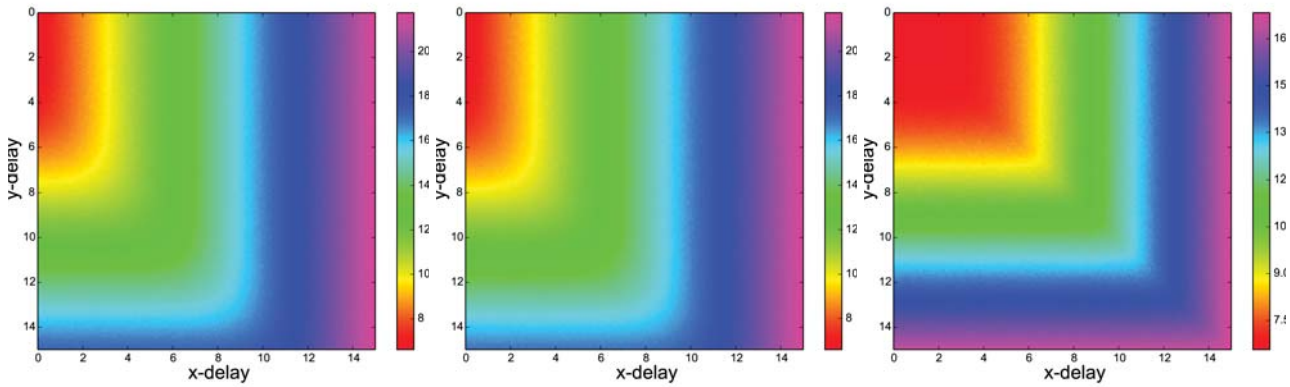

Figure 3: Task response time with delay vectors $(0, x, y)$ (left), $(x, 0, y)$ (middle) and $(x, y, 0)$ (right)

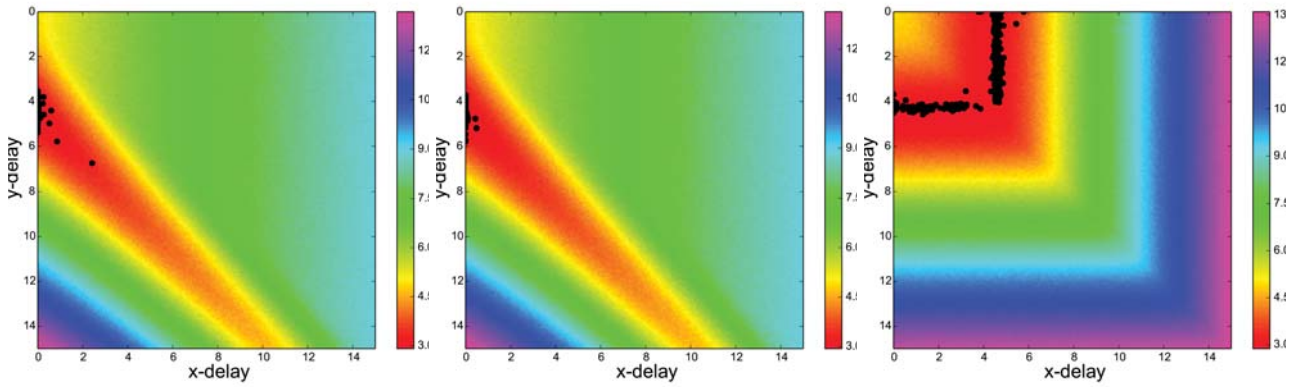

Figure 4: Trade-off $(\alpha = 0.5)$ with delay vectors $(0, x, y)$ (left), $(x, 0, y)$ (middle) and $(x, y, 0)$ (right)

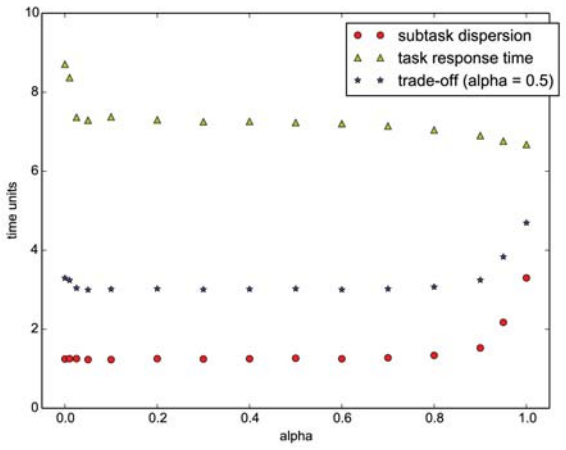

Figure 5: Influence of α on optimisation metrics

This resulted in the following:

Subtask dispersion:	1.247 time units
Task response time:	8.711 time units
Trade-off, $\alpha = 0.5$:	3.296 time units

Method 3 Optimised trade-off with $\alpha = 0.5$.

Again 1 000 samples per measurement point were taken. The CMA-ES algorithm was run on the problem 10 times and the subtask dispersion and task response time for each set of delays was calculated via simulation of 10^7 tasks.

This resulted in the following:

Subtask dispersion:	1.267 time units
Task response time:	7.233 time units
Trade-off, $\alpha = 0.5$:	3.027 time units

Figures 2, 3 and 4 display how task response time, subtask dispersion and trade-off ($\alpha = 0.5$) behave as the set of delay vectors is varied. In each picture one of the three delays of the controlling set S_c is set to 0 and the two others are allowed to vary between 0 and 15 with a stepsize of 0.1. The colouring of the image represents the resulting subtask dispersion (averaged over 1 000 samples) given the corresponding delay vector. The black dots in the images show the solutions produced by the CMA-ES technique. These are consistently located in the middle of the red band, suggesting near-optimal results.

Figure 5 shows how metrics vary as α is varied. All three metrics approach near-optimal values when α is between 0.2 and 0.8. When α approaches 1 the lowest possible task response time is found; however, this comes at the expense of subtask dispersion. Similarly, when α approaches 0 the lowest possible subtask dispersion is found; however, this comes at the expense of task response time.

5. CONCLUSION

Our method appears able to robustly control subtask dispersion in our case study hidden stochastic PERT network. Broader experimentation could establish whether these promising results hold more generally. Procedures for obtaining accurate confidence intervals should also be investigated.

6. REFERENCES

[1] V. Adlakha and V. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966–1987. *INFOR: Information Systems and Operational Research*, 27(3):272–296, 1989.

[2] D. R. Fulkerson. Expected critical path lengths in PERT networks. *Operations Research*, 10(6):808–817, 1962.

[3] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proc. 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1689–1696, 2010.

[4] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[5] P. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 64(7):664–689, 2007.

[6] P. G. Harrison and W. J. Knottenbelt. Passage time distributions in large Markov chains. In *ACM SIGMETRICS Performance Evaluation Review*, volume 30, pages 77–85. ACM, 2002.

[7] T. Pesu and W. J. Knottenbelt. Dynamic subtask dispersion reduction in heterogeneous parallel queueing systems. *Electronic Notes in Theoretical Computer Science*, 318:129 – 142, 2015.

[8] D. Sculli. The completion time of PERT networks. *Journal of the Operational Research Society*, 34(2):155–158, 1983.

[9] I. Tsimashenka, W. Knottenbelt, and P. Harrison. Controlling variability in split-merge systems. In *Proc. ASMTA 2012*, pages 165–177, Grenoble, France, June 2012.

[10] I. Tsimashenka and W. J. Knottenbelt. Trading off subtask dispersion and response time in split-merge systems. In *Proc. ASMTA 2013*, pages 431–442, Ghent, Belgium, July 2013.

[11] E. Varki. Response time analysis of parallel computer and storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1146–1161, 2001.

[12] S. Zander, I. Leeder, and G. Armitage. Achieving fairness in multiplayer network games through automated latency balancing. In *Proc. 2005 ACM SIGCHI ACE*, pages 117–124, 2005.