Approximating the longest path distribution of a stochastic DAG

Thomas McSweeney*

11th September 2020

1 Introduction

For any optimization problem, we obviously need to be able to evaluate how good any given solution is with regards to the optimization criteria in order to find an optimal, or otherwise acceptable, solution. The scheduling problems we consider here are clearly no exception: we want to know how good a computed schedule is, whether there are other schedules which are better, and so on. Evaluating the makespan of a given schedule would appear to be a straightforward problem—and so it is, when the task computation and communication costs are static. But if costs are *stochastic* then this may no longer be the case.

Now, as long as it specifies the execution order of tasks on processors, any schedule π for an application with task DAG G can be represented by another DAG G_π such that the longest path of G_π is equal to the makespan of π: G_π contains all the same vertices and edges as G, plus additional zero-weight *transitive* (or *disjunctive*) edges that indicate the execution order of the tasks on their chosen processors; the other weights of G_π are induced by the processor selections of π. There's some flexibility in how we add the transitive edges; the most straightforward way to do it is to simply add a transitive edge between a task t_i and the task t_h which is executed immediately before t_i on its chosen processor if an edge does not already exist between the two. This can be done cheaply, although it should be noted that the related problem of adding the minimal number of necessary transitive edges is NP-hard [citation - is this actually true?].

For example, consider the schedule π from Figure X and the graph G in Figure Y. We can construct the associated graph G_π as shown in Figure Z. The longest path through G_π can be computed in any standard way, such as dynamic programming (give pseudocode to refer to later?), and we see that it is indeed

*School of Mathematics, University of Manchester, Manchester, M13 9PL, England (thomas.mcsweeney@postgrad.manchester.ac.uk).

equal to the schedule makespan. More importantly, computing the longest path is an $O(n+e) \approx O(n^2)$ operation... Now suppose instead that the costs are actually stochastic, such as in Figure A... We try to compute the longest path in the same way as before, but the problem is that the path lengths themselves are now random variables (RVs)... Even assuming all of the distributions are known, we now have a series of maximizations and summations of RVs... The problem is that "no simple method exists for evaluating the distribution of the maximum of dependent RVs..." (doesn't follow from Hagstrom but might be the underlying explanation...) Unfortunately, Hagstrom [9] proved that this is a $\#P$-complete problem for discrete RVs and there is no reason to assume it is any easier for continuous costs instead.

Given the difficulty of the problem, bounds and approximations of the longest path distribution are needed instead... In this chapter, we describe several existing heuristic approaches for doing this. This chapter is essentially a bridge between those either side in that we don't consider how the schedule is computed, only how to evaluate its length... These principles underlie many stochastic scheduling heuristics... The problem of approximating the distribution of the longest path through a stochastic DAG has been studied in several contexts other than scheduling, initially in PERT network analysis [13] and later digital circuit design [1], so we try and use the more general terminology (i.e., longest path rather than makespan, also why we went into detail about the disjunctive graph)... Want to know robustness of schedule, for which best metric is std so want to know that...

We don't present many new results here, but we have implemented all of these methods in a software package and we consider one related subproblem in depth: how do we quickly update a longest path estimate as realizations become apparent?

2 Bounds on the moments

We already saw examples of these for discrete RVs in the previous chapter... Extend this and go into details...

Formal bounds on the distribution of the critical path length have been proven; for example, Kleindorfer gives both upper and lower bounds [11], the first of which was improved on by Dodin [6]. Often however we really want bounds on the moments of the distribution (which also tend to be cheaper to compute). We have already seen in Chapter **??** that, if the costs are discrete, then a cheap lower bound on the expected value of the makespan can be computed using upward ranking with the expected values of all costs. Moreover, a tighter bound can be found through Fulkerson's [8] alternative method, which was extended to continuous costs by Clingen [5] and later improved by Elmaghraby [7] and Robillard and Trahan [15]. All of these methods provide lower bounds for the expected value

alone; if we assume that all costs are normally distributed, then Kamburowski [10] was able to prove both lower and upper bounds on the expected value, as well as a lower bound on the variance (and a conjectured upper bound). Kamburowksi's bounds are based on a powerful common technique for approximating the first two moments of the critical path distribution, which is described in the following section.

3 Approximating the moments

Rather than a bound, it may be more useful to have estimates of the moments (or ideally the exact distribution). These are heuristic solutions which offer no guarantees but may be more useful in practice....

3.1 Assuming normality

Fundamentally, the critical path is computed through a series of summations and maximizations of the cost RVs. By the Central Limit Theorem, sums of random variables are asymptotically normally distributed so if we assume that the effect of the maximizations is minor, then the makespan distribution is likely to be approximately normal. Indeed, this has often been observed empirically, even when all costs follow very different distributions [2]. If we assume further that all of the cost RVs can be characterized by their mean and variance (i.e., effectively that they are also normal), then sums can be computed though the well-known rule for summing two normal RVs $\epsilon \sim N(\mu_\epsilon, \sigma_\epsilon^2)$ and $\eta \sim N(\mu_\eta, \sigma_\eta^2)$,

$$\epsilon + \eta \sim N\big(\mu_\epsilon + \mu_\eta, \sigma_\epsilon^2 + \sigma_\eta^2 + 2\rho_{\epsilon\eta}\sigma_\epsilon\sigma_\eta\big), \tag{1}$$

where $\rho_{\epsilon\eta}$ is the linear correlation coefficient between the two distributions. Formulae for the first two moments of the maximization of two normal RVs—which is not itself normal—are less well known but were first provided by Clark in the early 1960s [4]. Let

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \qquad \Phi(x) = \int_{-\infty}^{x} \phi(t)dt$$

be the unit normal probability density function and cumulative probability function, respectively, and define

$$\alpha = \sqrt{\sigma_\epsilon^2 + \sigma_\eta^2 - 2\rho_{\epsilon\eta}\sigma_\epsilon\sigma_\eta}, \qquad \beta = \frac{\mu_\epsilon - \mu_\eta}{\alpha}. \tag{2}$$

Then the first two moments $\mu_{\max}$ and $\sigma_{\max}$ of $\max(\epsilon, \eta)$ are given by

$$\mu_{\max} = \mu_\epsilon\Phi(\beta) + \mu_\eta\Phi(-\beta) + \alpha\phi(\beta), \tag{3}$$

$$\sigma_{\max}^2 = (\mu_\epsilon^2 + \sigma_\epsilon^2)\Phi(\beta) + (\mu_\eta^2 + \sigma_\eta^2)\Phi(-\beta) + (\mu_\epsilon + \mu_\eta)\alpha\phi(\beta) - \mu_{\max}^2. \tag{4}$$

Using (1) for summations, and (3) and (4) for maximizations (i.e., when we have multiple incident edges at a node), we can move forward through the DAG and compute approximations μ_{cp} and σ_{cp} for the first two moments of the critical path distribution in a fairly standard dynamic programming manner.

This method appears to have first been proposed for estimating the completion time of PERT networks by Sculli [16]. There he assumed that all correlations were zero, which is often unrealistic since common ancestors make critical path estimates at two nodes dependent, even if all costs themselves are independent. Computing the correlation coefficients efficiently is tricky. However Canon and Jeannot [3] proposed two different approaches which alternatively prioritize precision and speed. The first is a dynamic programming algorithm called Cordyn which makes use of the following formulae from Clark's original paper in order to recursively compute all relevant correlations:

$$\rho_{\tau,\max(\epsilon,\eta)} = \frac{\sigma_\epsilon \rho_{\tau\epsilon} \Phi(\beta) + \sigma_\eta \rho_{\tau\eta} \Phi(-\beta)}{\sigma_{\max}}, \tag{5}$$

$$\rho_{\tau,\text{sum}(\epsilon,\eta)} = \frac{\sigma_\epsilon \rho_{\tau\epsilon} + \sigma_\eta \rho_{\tau\eta}}{\sigma_{\text{sum}}}, \tag{6}$$

where τ is any normal random variable. Cordyn has time and space complexities $O(nv)$ and $O(n^2)$, respectively, so Canon and Jeannot propose an alternative called CorLCA which is based on the construction of a simplified version of the DAG called a *correlation tree* that has all the same vertices as the original but only retains a subset of the edges. In particular, where multiple edges are incident to a vertex—i.e., a maximization must be performed—only the edge which contributes most to the maximization is retained in the correlation tree. Then (approximate) correlations between any two RVs representing current finish time estimates at any two vertices can be computed easily by finding the lowest common ancestor (LCA) of the two in the correlation tree. The complexity of CorLCA depends on the cost of the method used for the lowest common ancestor queries, but Canon and Jeannot hypothesize this can be done with time and space complexities of $O(1)$ and $O(m)$, respectively, giving time and space complexity $O(m)$ and $O(n)$ for the entire algorithm. This is not proven, however extensive numerical simulations performed by the original authors suggest that CorLCA is more efficient than Cordyn with only a relatively small reduction in accuracy.

Although there are no theoretical guarantees, in practice the mean and variance estimates obtained using the Clark equation approach tend to be fairly good, even when correlations are disregarded (i.e., Sculli's method) [10]; as a rule performance improves as the cost distributions move closer to normality and the number of tasks increases. Furthermore, Sculli's method in particular is typically much faster than alternative approaches, with CorLCA and Cordyn being slightly more expensive but still competitive in this regard (particularly CorLCA) [3].

Another method of estimating the critical path distribution also makes use of the relative ease of dealing with normal RVs. In the *canonical model* [18, 19],

all RVs are expressed as the sum of their expected value and a weighted sum of standard normal distributions that characterize the variance,

$$\eta = \mu + \sum_i v_i \delta_i,$$

where all $\delta_i \sim N(0,1)$ and are independent of one another. The advantage of this is that evaluating summations and maximizations becomes much more straightforward. Let $\eta = \mu_\eta + \sum_i v_{\eta,i}\delta_i$ and $\epsilon = \mu_\epsilon + \sum_i v_{\epsilon,i}\delta_i$. Then

$$\omega = \eta + \epsilon = (\mu_\eta + \mu_\epsilon) + \sum_i (v_{\eta,i} + v_{\epsilon,i})\delta_i.$$

Suppose now that $\omega = \max(\eta, \epsilon)$. Let α and β be defined as in (2), and $\Phi(x) = \int_{-\infty}^{x} \phi(t)dt$ be the standard normal cdf. Note that computing β requires the linear correlation coefficient $\rho_{\eta\epsilon}$ which can be efficiently calculated as:

$$\rho_{\eta\epsilon} = \frac{\sum_i v_{\eta,i} v_{\epsilon,i}}{\sqrt{\sum_i v_{\eta,i}^2} \cdot \sqrt{\sum_i v_{\epsilon,i}^2}}.$$

By definition, $\mathbb{P}[\eta > \epsilon] = \Phi(\beta)$ and we can therefore approximate ω by

$$\begin{aligned}\hat{\omega} &= \Phi(\beta)\eta + \Phi(-\beta) \\ &= \Phi(\beta)\mu_\epsilon + \Phi(-\beta)\mu_\epsilon + \sum_i \big(\Phi(\beta)v_{\eta,i} + \Phi(-\beta)v_{\epsilon,i}\big)\delta_i.\end{aligned}$$

This is both similar and in some sense contrary to the Clark equation approach, in that the latter precisely computes the first two moments of the maximization of two normal RVs, whereas the canonical method approximates the distribution of the maximization of any two RVs using linear combinations of normal RVs. Canon and Jeannot found in an empirical study that the canonical representation approach tended to fall between Sculli's method and CorLCA in terms of both speed and approximation quality [3].

3.2 Monte Carlo

Monte Carlo methods have a long history in approximating the completion time distribution of PERT networks, dating back to at least the early 1960s [17]. The idea is to simulate the realization of all RVs and evaluate the critical path of the resulting deterministic graph. This is done repeatedly, giving a set of critical path instances whose empirical distribution function is guaranteed to converge to the true distribution by the Glivenko-Cantelli theorem [3]. Furthermore, analytical results allow us to quantify the approximation error for any given the number of realizations—and therefore the number of realizations needed to reach a desired accuracy.

The major disadvantage of Monte Carlo methods is their cost, particularly when the number of realizations required is large, although modern architectures are well-suited to MC methods because of their parallelism so this problem may no longer be as acute as it once was. Note also that this analysis is predicated on the notion that cost distributions are known precisely. In practice they are more likely to be estimated based on samples of previous experiences so that, despite the analytical guarantees, the uncertainty surrounding the cost distribution fits may mean that the critical path distribution which MC methods converge to is actually only an approximation of the true distribution.

3.3 Series-parallel reductions

If the graph is *series-parallel* then we can compute the exact distribution of the critical path length in polynomial-time through a series of reductions [6, 14]. If this isn't the case, similar methods have been proposed that give approximations to the distribution [6, 12], however these tend to be less accurate than Monte Carlo-based methods and more expensive than approaches based on Clark's equations [3].

4 Updating longest path estimates

Suppose we are following a static schedule for which we have already computed an estimate of the makespan, how do we efficiently update the estimate dynamically at runtime, perhaps in response to a large delay? This is straightforward for deterministic static schedules since we can just recompute the critical path length of the schedule DAG using observed costs for those task that have already been processed and estimates for those that have not yet been processed. The same basic approach can also be used for stochastic schedules of course, but the picture is slightly more complex, as illustrated below.

For a given task DAG and target platform, suppose we have a static schedule that dictates what tasks each processor should execute and in what order, which they will do greedily (i.e., without artificial delays). Before runtime we work through the DAG and estimate the makespan distribution up to when each of the tasks has completed. These makespan estimates are modeled as normal RVs in accordance with the central limit theorem so that for each task t_i we have an associated makespan estimate $F_i \sim N(\mu_i, \sigma_i^2)$, which is computed through

$$F_i = W_i + \max_{h \in P_i} \{F_h + W_{hi}\}, \tag{7}$$

where W_i and W_{hi} here represent the relevant computation and communication costs (since the processors are fixed by the schedule). We make the standard assumption that all of the computation and communication costs are independent

so the summations are done using the rule for normal RVs (1) with the correlation coefficient assumed to be zero. We use Clark's equations (3) and (4) to estimate the distribution of the maximization. The terms within are not generally independent because the parent tasks they represent may have common ancestors, so we can either ignore all correlations (which corresponds to Sculli's method) or estimate the correlation coefficients (as in CorLCA and Cordyn).

We are concerned with the situation at any given point during runtime when some tasks may have completed execution while others are still to be done. Let f_h be the realization of the makespan estimate F_h once task t_h has actually been completed. In order to update the makespan estimate we need to move through the DAG and make use of the realizations of previous task makespans. Consider a generic task t_i and suppose that some of its parent tasks have been processed but others have not. The most straightforward way to update F_i is to simply use f_h instead of F_h in equation (7) for all those parents that have been realized. However, it would perhaps be better if we could use the realized parent makespans to update the makespan estimates F_h for those parents that haven't yet finished. Let $f_m = \max_{h \in P_i} f_h$ be the greatest makespan of those parent tasks that have been processed. For all other parent tasks t_h that have *not* yet been executed we want to estimate the remaining cost, given that it has not been completed but t_m has been—i.e., we want to estimate $F'_h = F_h - f_m$ given that we know $f_h > f_m$. This can be done by applying the following result.

Proposition 1 *Let u be a normally distributed vector with mean $\bar{\mu}$ and covariance Σ_u, $u \sim N(\bar{\mu}, \Sigma_u)$, and suppose that $u = (v, w)$. Then*

$$(w \mid v) \sim N\big(\bar{w} + \Sigma_{wv}\Sigma_{vv}^{-1}(v - \bar{v}),\ \ \Sigma_{ww} - \Sigma_{wv}\Sigma_{vv}^{-1}\Sigma_{vw}\big).$$

Note that if all of the linear correlation coefficients ρ_{uv} are known (which would be the case if we used CorLCA or Cordyn), we can use the definition $\rho_{uv} = \Sigma_{uv}/\sigma_u\sigma_v$ to find Σ_{uv}. For compactness of notation let $\rho_{hm} = \rho_{mh}$ be the linear correlation coefficient between F_h and F_m. Then we have

$$\begin{aligned} F'_h &\sim N\left(\mu_h + \frac{\rho_{hm}\sigma_h\sigma_m}{\sigma_m^2}(f_m - \mu_m),\ \ \sigma_h^2 - \frac{\rho_{hm}\sigma_h\sigma_m \cdot \rho_{mh}\sigma_m\sigma_h}{\sigma_m^2}\right) \\ &\sim N\left(\mu_h + \frac{\rho_{hm}\sigma_h}{\sigma_m}(f_m - \mu_m),\ \ (1 - \rho_{hm}^2)\sigma_h^2\right). \end{aligned}$$

The idea is that by using F'_h rather than F_h for those parent tasks that have not yet completed we can make greater use of the data available and should therefore obtain a superior estimate of the new makespan.

5 Results

We created a software framework to study the problem of approximating the makespan distribution of stochastic DAGs...

5.1 Benchmarking

We don't spend too much time on this since it was done much more thoroughly by Canon and Jeannot but we consider Sculli, CorLCA, possibly canonical and Cordyn for the Cholesky DAGs sets (i.e., we focus on a single example)...

5.2 Update rule

Consider different permutations (fraction of tasks initially realized, distributions of the costs, etc) in a systematic manner and compare with MC estimates...

6 Conclusions

What we really want to know: is there enough justification for considering the correlations in the context of a stochastic scheduling heuristic?

References

[1] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: From basic principles to state of the art. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):589–607, 2008.

[2] L. Canon and E. Jeannot. Evaluation and optimization of the robustness of DAG schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):532–546, 2010.

[3] Louis-Claude Canon and Emmanuel Jeannot. Correlation-aware heuristics for evaluating the distribution of the longest path length of a DAG with random weights. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3158–3171, 2016.

[4] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145–162, 1961.

[5] C. T. Clingen. A modification of Fulkerson's PERT algorithm. *Operations Research*, 12(4):629–632, 1964.

[6] Bajis Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33(4):862–881, 1985.

[7] Salah E. Elmaghraby. On the expected duration of PERT type networks. *Management Science*, 13(5):299–306, 1967.

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

[9] Jane N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18(2):139–147.

[10] Jerzy Kamburowski. Normally distributed activity durations in PERT networks. *Journal of the Operational Research Society*, 36(11):1051–1057, 1985.

[11] George B. Kleindorfer. Bounding distributions for a stochastic acyclic network. *Operations Research*, 19(7):1586–1601, 1971.

[12] Arfst Ludwig, Rolf H. Möhring, and Frederik Stork. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research*, 102(1-4):49–64, 2001.

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

[14] J. J. Martin. Distribution of the time through a directed, acyclic network. *Operations Research*, 13(1):46–66, 1965.

[15] Pierre Robillard and Michel Trahan. Technical note—expected completion time in PERT networks. *Operations Research*, 24(1):177–182, 1976.

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

[17] Richard M. Van Slyke. Letter to the editor—Monte Carlo methods and the PERT problem. *Operations Research*, 11(5):839–860, 1963.

[18] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett. First-order incremental block-based statistical timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2170–2180, 2006.

[19] L. Zhang, W. Chen, Y. Hu, and C. C. Chen. Statistical static timing analysis with conditional linear MAX/MIN approximation and extended canonical timing model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1183–1191, 2006.