Critical path estimation in heterogeneous scheduling heuristics

Thomas McSweeney*

30th July 2020

1 Introduction

Recall from Chapter X that the *Heterogeneous Earliest Finish Time* (HEFT) heuristic prioritizes all tasks t_i, $i = 1, \ldots, n$, by recursively computing a corresponding sequence of numbers u_i which are intended to represent the *critical path* lengths from each task to the end. However, as noted previously, the concept of the *critical path* is not clearly defined: DAG weights are not fixed at this stage so there are multiple ways we could define a *longest* (i.e., *costliest*) path. Consider for example the simple DAG shown in Figure X, where the labels represent...

Although HEFT successfully finds the optimal schedule, it's not clear what the u_i really represent since they certainly aren't a lower bound on the schedule makespan... Conceptually, the HEFT critical path isn't a lower bound on any schedule makespan but a lower bound on what the schedule is likely to be...

Alternative ways to definite the critical path in HEFT have been considered before, most notably by Zhao and Sakellariou [1], who empirically evaluated the performance of HEFT when averages other than the mean (e.g., median, maximum, minimum) are used to compute upward (or downward) ranks. Ultimately they concluded that mean values did not appear to be a superior choice to the others, although none of the options uniformly dominated all others; the bigger takeaway was that HEFT is very sensitive to how the task ranks are computed, with significant variation being seen for individual graphs. In this chapter we undertake a similar investigation with the aim of establishing if there are choices which do consistently outperform the standard HEFT task ranking phase.

This will be an empirically-driven study, as much of this subject is... We created a simulator much like that described in the previous chapter, although in this case it simulates more general heterogeneous environments...

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

2 Optimistic bounds

Functionally, the critical path is treated as a lower bound on the makespan, so that minimizing the critical path gives us the most scope to minimize the makespan (assuming we make good use of our parallel resources). Assuming that we want to define the critical path in such a way that it represents a lower bound on the makespan of any possible schedule, there are several ways we could do so. The most straightforward would be to just set all weights to their minimal values and compute the longest path in the standard way. This is a loose bound since not all combinations may be possible. A better way to compute a tighter lower bound is to first define for all tasks t_i and processors p_a u_i^a, the critical path length from the task (inclusive), assuming it takes the weight associated with processor p_a (i.e., W_i^a). In particular, for all $i = n, \ldots, 1$ (i.e., working upward from the leaves) we recursively compute

$$u_i^a = W_i^a + \max_{k \in S_i} \left(\min_{b=1,\ldots,q} \left(u_k^b + W_{ik}^{ab} \right) \right), \tag{1}$$

with $u_i^a = W_i^a$ if t_i is an exit task. Then for all tasks t_i the associated value

$$u_i = \min_{a=1,\ldots,q} u_i^a \tag{2}$$

gives a true lower bound on the makespan of any schedule.

While this is a lower bound on the makespan of any schedule, it is not clear that it is truly useful in practice since the underlying principle for homogeneous processors does not hold...

Alternatively, we could use any other average over the set of processors rather than the minimum, such as the mean or maximum. These are less intuitively useful...

3 Sharper bounds?

The HEFT approach is to instead compute what we *expect* the critical path to be, which it does by using mean values over all processors to set costs and then computing the critical path of this associated *fixed-cost* DAG using dynamic programming. In some sense this approach to the scheduling problem is a symbiotic problem...By taking means in this manner, effectively the node and edge weights are viewed as discrete random variables (RVs) with associated probability mass functions (pmfs) given by assuming that the task is equally likely to be scheduled on all of the processors in the selection phase. More precisely, let m_i be the pmf corresponding to the task weight variable w_i and m_{ik} that for the edge weight w_{ik}, then

$$m_i(c_i) \coloneqq \mathbb{P}[w_i = c_i] = \frac{n_c}{n_p}, \qquad m_i(g_i) \coloneqq \mathbb{P}[w_i = g_i] = \frac{n_g}{n_p},$$

and

$$m_{ik}(0) = \frac{n_c + n_g}{n_p^2}, \qquad m_{ik}(C_{ik}^c) = \frac{n_c(n_c-1)}{n_p^2},$$
$$m_{ik}(G_{ik}^g) = \frac{n_g(n_g-1)}{n_p^2}, \qquad m_{ik}(C_{ik}^g) = m_{ik}(G_{ik}^c) = \frac{n_c n_g}{n_p^2}.$$

The expected values of the node and edge weights are therefore given by

$$\mathbb{E}[w_i] = \sum_{\ell \in L_i} \ell m_i(\ell) = \frac{c_i n_c + g_i n_g}{n_p}, \tag{3}$$

$$\begin{aligned}\mathbb{E}[w_{ik}] &= \sum_{\ell \in L_{ik}} \ell m_{ik}(\ell) \\ &= \frac{n_c(n_c-1)C_{ik}^c + n_c n_g (C_{ik}^g + G_{ik}^c) + n_g(n_g-1)G_{ik}^g}{n_p^2}.\end{aligned} \tag{4}$$

Finally, upward ranks u_i for all tasks are computed by setting $u_i = \mathbb{E}[w_i]$ for all exit tasks, moving up the DAG and recursively computing

$$u_i = \mathbb{E}[w_i] + \max_{k \in S_i} \left(u_k + \mathbb{E}[w_{ik}] \right) \tag{5}$$

for all other tasks.

For example, suppose we wish to schedule the DAG from Figure X on a platform with 4 CPUs and 2 GPUs. To determine the upward rank of all tasks, HEFT implicitly transforms the original DAG into the fixed-cost one in Figure Y by computing the expected value of all nodes and edges using equations (3) and (4) respectively.

The upward ranks of all tasks are then given by

$$\begin{aligned}
u_4 &= 3, \\
u_3 &= 2 + \left(u_4 + \frac{8}{9} \right) = \frac{53}{9}, \\
u_2 &= 3 + \max\left\{ u_3 + \frac{7}{9}, u_4 + \frac{10}{9} \right\} = 3 + \max\left\{ \frac{20}{3}, \frac{37}{9} \right\} = \frac{29}{3}, \\
u_1 &= 3 + \max\left\{ u_2 + \frac{1}{2}, u_3 + \frac{19}{18} \right\} = 3 + \max\left\{ \frac{61}{6}, \frac{125}{18} \right\} = \frac{79}{6}.
\end{aligned}$$

The final value u_1 is effectively treated as an estimate of the critical path length of the original DAG.

To sum up, since all possible node and edge weights are known but their actual values at runtime aren't (at least without restricting the processor selection phase), HEFT estimates critical path lengths from all tasks in a task graph G through a two-step process:

1. An associated stochastic DAG G_s is implicitly constructed with node and edge pmfs m_i and m_{ik} as defined above.

2. The numbers u_i are recursively computed for all tasks in G_s using (5), and taken as the critical path lengths from the corresponding tasks in G.

In the following two sections, we propose modifications of both steps, in turn, so as to obtain more useful critical path estimates in HEFT. The performance of the task ranking procedures defined by using these alternative estimates is then evaluated through extensive numerical simulations.

Conclusion: weighted mean promising but no real evidence to justify using others...

4 Experimental comparison of rankings

Baseline comparison with random sort.

5 Processor selection

References

[1] Henan Zhao and Rizos Sakellariou. An experimental investigation into the rank function of the Heterogeneous Earliest Finish Time scheduling algorithm. In *Euro-Par 2003 Parallel Processing*, Harald Kosch, László Böszörményi, and Hermann Hellwagner, editors, Berlin, Heidelberg, 2003, pages 189–194. Springer Berlin Heidelberg.