Critical path estimation in heterogeneous scheduling heuristics*

Thomas McSweeney[1 [0000−0001−9866−2229]], Neil Walton[1 [0000−0002−5241−9765]], and Mawussi Zounon[1,2 [0000−0002−6955−1500]]

[1] University of Manchester, Manchester, UK
thomas.mcsweeney@postgrad.manchester.ac.uk
[2] The Numerical Algorithms Group (NAG), Manchester, UK

Abstract. Critical path estimates are used...

Keywords: High-performance computing · Scheduling · Precedence constraints · Directed acyclic graphs.

1 Introduction

The concept of the *critical path* has no real meaning at the task prioritization stage of HEFT (or any similar listing heuristic): no processor selections have been made so the DAG weights are not yet fixed and no *longest* path can exist. For example, consider the simple DAG in Figure X... What is the critical path of this DAG? It isn't clear: we could, for example, define an *optimistic* critical path length by taking the smallest possible costs for all weights, as in the PEFT heuristic, or a *pessimistic* critical path by using the largest possible weights. The former is a lower bound on the cost of any schedule, no matter the composition of the target platform, so can be useful but is unlikely to reflect the true critical path of the DAG at runtime since no platform-specific information is used.

2 HEFT

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. 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

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

* Supported by the Engineering and Physical Sciences Research Council (EPSRC).

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

$$\mathbb{E}[w_i] = \sum_{\ell \in L_i} \ell m_i(\ell) = \frac{c_i n_c + g_i n_g}{n_p}, \tag{1}$$

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

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

$$u_i = \mathbb{E}[w_i] + \max_{k \in S_i} \left(u_k + \mathbb{E}[w_{ik}]\right) \tag{3}$$

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 (1) and (2) 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 (3), 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.

References