5J-06

DF/IHS法の木構造を用いたタスクスケジューリング問題の下界値計算手法

薬師 啓太† 中村 あすか† 前川 仁孝†

†千葉工業大学情報科学部情報工学科

1 はじめに

DF/IHS(Depth First /Implict Heuristic Search) 法 [1] は分枝限定法に基いたタスクスケジューリング問題の厳密解法である．本手法では，下界値を未割当てタスクから算出するため，同じ未割当てタスクを持つ部分問題で下界値算出に同じ計算を実行する場合がある．そこで，本研究は，DF/IHS 法を高速化するために，未割当てタスクが同じ部分問題を検出し，下界値計算で重複する演算を削減する．

2 タスクスケジューリング問題

タスクスケジューリング問題は，n 個のタスクを能力の等しい m 台の PE(Processing Element) で並列処理する際に，スケジュール長が最小となるタスクの割当て方を求める問題である．図 1 にタスクグラフの例を示す．図 1 のノードはタスク，エッジは先行制約，ノード内および右上の数字はタスク番号と処理時間，タスク S と G はダミータスクを表す．

3 DF/IHS 法

DF/IHS 法は，分枝限定法に基づき実行可能タスクの組合せから探索木を生成し，深さ優先探索で厳密解を求解する手法である．図 2 に，DF/IHS 法の探索木の例を示す．本例の探索木は根の部分問題が図 1 であり，ϕ はレディ状態を表す．本手法は，探索木中の部分問題に実行可能タスクを割り当てることで分枝し，子問題を生成する．また，下界値 lb の算出には式 (1)～式 (8) を用いる．式中の $I(\pi_a)$ は部分問題 π_a の未割当てタスクの集合，m は PE 数，t_0 はタスク割当てが可能となる時刻，t_j はタスク j の処理時間，l_j はタスク j のクリティカルパスである．

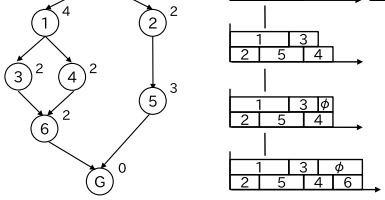

図 1: タスクグラフ

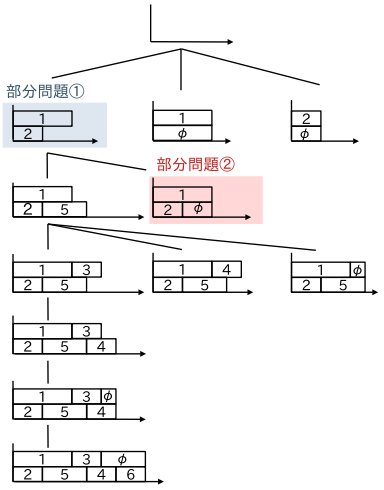

図 2: DF/IHS 法の探索木

$$lb = \max(lb_{cr}, \quad lb_{div}, \quad lb_{hu}) \tag{1}$$

$$lb_{cr} = \max_{j \in I(\pi_a)} l_j + t_0 \tag{2}$$

$$lb_{div} = \lceil \sum_{j \in I(\pi_a)} \frac{t_j}{m} \rceil + t_0 \tag{3}$$

$$lb_{hu} = lb_{cr} + \max_{0 \le t_k \le lb_{cr} - t_0} \lceil -t_k + \frac{s_k}{m} \rceil \tag{4}$$

$$s_k = \int_0^{t_k} F(\overline{\tau}, t) dt \tag{5}$$

$$F(\overline{\tau}, t) = \sum_{j \in I(\pi_a)} f(\overline{\tau_j}, t) \tag{6}$$

$$f(\overline{\tau_j}, t) = \begin{cases} 1, & for \quad t \in [\overline{\tau_j}, \overline{\tau_j} + t_j] \\ 0, & otherwise \end{cases} \tag{7}$$

$$\overline{\tau_j} = lb_{cr} - t_0 - l_j \tag{8}$$

4 木構造を用いた下界値計算手法

DF/IHS 法は，文献 [2] の計算手法を用いることで下界値 lb の算出に必要な演算回数を削減できるが，$I(\pi_a)$ が同じ部分問題で $F(\overline{\tau}, t)$ の再計算が必要となり無駄が生じる．このため，提案手法では式 (6) を式 (9) に置き換える．

A Calculation Algorithm of Lower Bound of Task Scheduling Problem Using Tree Structure of DF/IHS Method

Keita Yakushi†, Asuka Nakamura†, Yoshitaka Maekawa†
†Information and Computer Science, Chiba Institute of Technology, 275-0016, Narashino, Japan
yakushi@mae.cs.it-chiba.ac.jp

$$F(\overline{\tau}, t) = \sum_{j \in I(\pi_p)+T_\phi} f(\overline{\tau_j}, t) \tag{9}$$

ここで, π_p は親問題, T_ϕ は分枝で生成したレディ状態 ϕ を実行するタスクである. 式 (9) は, j の親集合を, 親問題 π_p の未割当てタスクの集合に T_ϕ を加えたものであり, 親問題で計算済みの lb_{cr}, t_k, s_k を参照することで演算回数を削減できる.

提案手法による変更点は, 割当て済みタスクを未割当てタスクとして扱う点であり, lb が下界である点は保証される. 図 3 に, 図 2 の部分問題①②を例とした提案手法の有無による部分問題の違いを示す. 図 3 の網掛けは割当て済みタスクであり, タスク $1'$ はタスク 1 の実行未完了分の処理時間である. 部分問題②は, ①にレディ状態 ϕ を割り当てた問題であり, (a) に T_ϕ を加えたタスクグラフで表すことができる. 提案手法で式 (9) を用いることは (c) の下界値を求めることと同義である. 図 3 のタスクグラフ (a), (b), (c) より算出した下界値 lb_a, lb_b, lb_c は, $lb_a \leq lb_c \leq lb_b$ が成り立つことから, DF/IHS 法は提案手法を用いても厳密解を求められることが確認できる. なお, 式 (9) は ϕ を割り当てる分枝を想定しているが, ϕ の割当てがない分枝では, $t_\phi = 0$ となり親問題と同じ下界値が得られる.

5 評価

提案手法の有効性を確認するために, PE 数 $m = 2, 4, 8, 16$ で標準タスクグラフセット [3] のタスク数 $n = 50$ の問題 180 問のうち 1 時間以内に求解できた問題, 計 701 問にスケジュールを行う. 評価環境は, CPU が Intel Xeon CPU E5-2687W v2, メモリが 64GB である. 表 1, 表 2 に式 (10), 式 (11) より算出した削減率と高速化率を示す.

$$\text{削減率 [\%]} = (1 - \frac{\text{提案手法の探索ノード数}}{\text{DF/IHS 法の探索ノード数}}) \times 100 \tag{10}$$

$$\text{高速化率 [倍]} = \frac{\text{DF/IHS 法の実行時間}}{\text{提案手法の実行時間}} \tag{11}$$

表 1 より, 探索ノード数が多くなるほど削減率は大きくなる. これは, DF/IHS 法が図 2 のように探索木の右側に ϕ を含む部分問題を多く生成するためである. DF/IHS 法が探索木を左から深さ優先探索することから, 探索ノード数の多い問題では, ϕ を多く含む探索木の右側まで探索する. ϕ の割当てが多い部分問題ほど提案手法で得られる下界値が大きくなりやすいため, 探索ノード数の多い問題で高い削減率が得られたと考えられる.

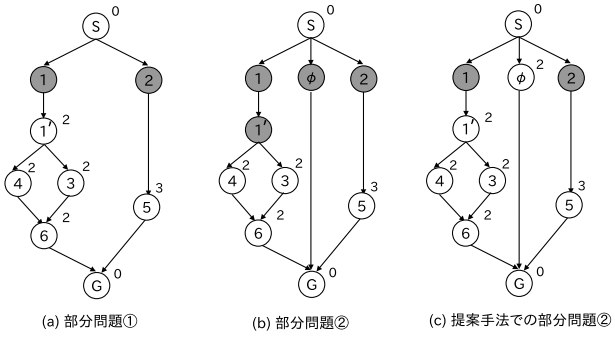

図 3: 部分問題のタスクグラフ

表 1: 削減率 [%]

探索ノード数	問題数	平均	最大	最小
$10^6 \leq N$	22	29.4	65.7	2.76
$10^3 \leq N < 10^6$	57	23.3	55.1	6.25
$N < 10^3$	622	0.211	32.7	0
全体	701	4.12	65.7	0

表 2: 高速化率 [倍]

実行時間 [s]	問題数	平均	最大	最小
$1 \leq t_n$	16	1.07	1.52	0.97
$0.001 \leq t_n < 1$	42	1.01	1.10	0.89
$t_n < 0.001$	643	0.99	1.24	0.74
全体	701	0.99	1.52	0.74

6 おわりに

本研究では, DF/IHS 法を高速化するために, 親問題で求めた値を用いて, 下界値計算で重複する演算を削減する手法を提案した. 評価の結果, 提案手法を用いることで最大約 1.5 倍の高速化率が得られることを確認した.

参考文献

[1] 笠原博徳, 成田誠之助:マルチプロセッサ・スケジューリング問題に対する実用的な最適及び近似アルゴリズム，電子情報通信学会論文誌 D，Vol.67, No.7, pp.792–799 (1984).

[2] 中村あすか, 前川仁孝 : タスクスケジューリング問題の厳密解求解における探索ノード数削減アルゴリズム, 情報処理学会論文誌プログラミング (PRO), Vol.7, No.1, pp.1–9 (2014).

[3] Standard Task Graph Set, available from <https://www.kasahara.cs.waseda.ac.jp/schedule/> (accessed 2024/01/12).