マルチプロセッサ・スケジューリング問題に対する分枝限定法の適用

笠原　博徳

1. まえがき

マルチプロセッサ方式の並列処理システムは科学技術計算用超大型計算機（スーパーコンピュータ），Prolog等の論理型言語を処理する高速推論マシン，あるいは低価格高性能のロボットコントローラの開発等を始め，幅広い分野でその導入が図られている．このようなマルチプロセッサシステムを用いて処理の高速化を図る場合には，処理時間を最小とするために処理すべきタスク（プログラムの小部分）集合をプロセッサにどのように割り当て，どのような順序で実行すべきかを決定する問題の求解が非常に重要となる．この問題は実行終了時間（スケジュール長）最小マルチプロセッサ・スケジューリング問題[1]として知られ，長年にわたって活発な研究が行なわれてきたにもかかわらず効率のよいアルゴリズムが開発されていないきわめてむずかしい（intractable）最適化問題である．特に，実際の並列処理で要求されるスケジューリング問題が，（強）ＮＰ困難[2][3]であることが知られてからは，最適化アルゴリズムの作成と同時にスケジューリング理論の実システムへの応用すらほとんど諦められていた．これに対し，最近，分枝限定法を応用した実用的な[3]最適化スケジューリング・アルゴリズム[5]が開発され，そのアルゴリズムを用い実マルチプロセッサ・システム上で効率良い並列処理が実現できる[6][7][8]ことが示されてからは，再びマルチプロセッサ・スケジューリング問題が注目を集めている．本稿では，この実用的な最適マルチプロセッサ・スケジューリング・アルゴリズムで，どのように分枝限定法を応用しているかについて，簡単に解説を行なう．

かさはら　ひろのり　早稲田大学　理工学部電気工学科
〒160 新宿区大久保３－４－１

2. 実行終了時間最小マルチプロセッサ・スケジューリング問題

本稿で扱うマルチプロセッサ・スケジューリング問題は，能力の等しいm台のプロセッサで，n個の処理時間の異なるタスク[1][9]から成るタスク集合$T=\{T_1,\cdots,T_n\}$を処理割り込みがないという条件下で並列処理をするさい，その実行時間（スケジュール長）を最小にするようなスケジュールを求める問題である．この時，タスク集合Tは**図１**のようなタスクグラフと呼ばれる有限無サイクル有向グラフで記述されるものとする[1][5]．図１中のノードは１つのタスクを表わし，ノード内の数字はタスク番号iを，またノード横の数字は各タスクの処理時間t_i unit time（以下[$u.t$]）を，ノードN_iからN_jへのアークはT_iがT_jに先行する（T_iが実行終了するまでT_jは実行開始できない）という半順序（partially order）制約を表わしている．ただしここでノード０とノード９は入口と出口を表わすため便宜上導入されたダミーノードである．このスケジューリング問題は実際のマルチプロセッサ・システム上で並列処理を行なうさいに求解が要求される問題の１つであり，強ＮＰ困難な問題である[3]．

3. 分枝限定法の適用

分枝限定法は，本スケジューリング問題のような（強）ＮＰ困難な問題に対して精度の保証された最適解あるいは近似解を得るための強力な武器となる．しかし，このような問題に対しては，最小下限値探索（ＬＬＢ），ヒューリスティック探索（H）等のようにアクティブ・ノード[4]（部分問題）数が問題の規模とともに指数関数のオーダで増加してしまう探索法は使用できない．そこで，ここでは DF/IHS (Depth First/Implicit Heuristic

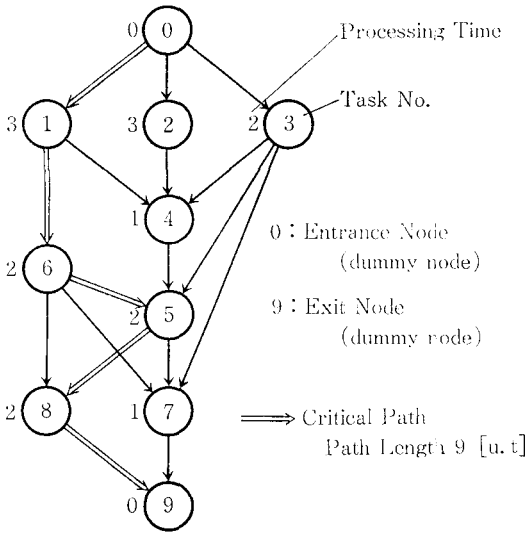

図 1　タスクグラフの例

Search)と呼ぶ一種のヒューリスティックを用いた深さ優先探索法の適用について述べる．DF/IHS は従来の一般的な DF/IHS とは異なり深さ最大のアクティブ・ノードすべてに対してヒューリスティック値を計算し最小のものを選ぶというようなヒューリスティック・アルゴリズムの用い方をしないところに特徴がある．DF/IHS は，探索中に生成されたノード(ある割当時点で，どのタスクをプロセッサに割り当てるかというタスクの組合せを示す)に対して，探索開始前に CP/MISF(Critical Path/Most Immediate Successors First)法と呼ぶマルチプロセッサ・スケジューリング用のヒューリスティック・アルゴリズムの概念を利用して優先順位をつけることにより，次分枝ノードセレクション時にはヒューリスティック値の計算をまったく行なわずに適切な選択を行なうことができる探索法である．これにより DF/IHS 法では，領域複雑度および探索時間を顕著に低減することを可能としている．DF/IHS はヒューリスティックを用い探索木中のノードに優先順位を与えるための前処理（リナンバリング）部分と DF/FIFO 形状の深さ優先探索（バックトラック）部分の2つから構成されており以下にそれらについて簡単に述べる．

3.1　前処理(タスク・リナンバリング)

前処理部では，各タスクのレベル l_i

$$l_i=\max_k \sum_{j\in\pi_k} t_j$$

π_k：出口ノードからノード i へ至る k 番目のパスを計算し，その後 l_i が大きいか，または l_i が等しい場合には直接の後続タスク数が多い順に各タスクのプライオリティ・リストを作成し（CP/MISF 法はこのプライオリティ・リストにしたがい実行可能なタスクをプロセッサに割り当てるアルゴリズムである），それと同順に全タスクのタスク番号をつけ換える．この前処理部分の時間複雑度は $O(n^2)$である．

3.2　深さ優先探索部

次に DF/IHS の後半部分である深さ優先探索部について述べる．分枝限定法とは，全体問題を複数の部分問題に分解し(分枝規則)，その部分問題をある規則(セレクション規則)にしたがって選択し，その部分問題の解が現在までに見つかっている解（上限値）よりも良い解を生成する可能性があるかを調べ（下限関数値と上限値を用いノード除去規則により調べる），あればさらに分解するという操作を最適解が見つかるまで繰り返すという方法である．以下では，DF/IHS における以上の規則を説明する．

3.2.1　分枝規則 B_p

DF/IHS における本問題の部分問題への分解法について述べる．以下では，理解を容易にするために，図1の8個のタスクを2台のプロセッサ上にスケジュールする際の分枝図（**図2**）を例に取り上げながら議論を進める．ただし図2中の各ノードは，ある割当時点においてどのタスクがプロセッサに割り当てられるかというタスクの1つの組合せ（図中円内上部の数字で示し，1，2はタスク1，タスク2がプロセッサに割り当てられることを示す)を表わしている．またここで割り当て時点とは，1つあるいは複数のプロセッサがそれまでに割り当てられていたタスクの実行を終了し，次のタスクの実行が可能となる時点を表わしている．また各ノードの左横に書かれている t は，そのノードの次のノードが生成される時刻，すなわちそのノードで割り当てられたタスクのうち，1つ以上のタスクが終了する最も早い時刻を表わしており，その時に使用可能になるプロセッサ数を m_{av}，また実行可能なタスク(レディタスク)を CP/MISF のプライオリティの高い順（すなわちリナンバリング後のタスク番号の小さい順）に並べ，その後アイドルタスクを並べたものをレディタスク・テーブル R として示している．ここでアイドルタスク（図中の ϕ）とは，最適解を得るために導入された，プロセッサを強制的にアイドル（停止状態）にする仮想的なタスクである．

3.2.2　セレクション規則 S

セレクション規則 S は現在のアクティブ・ノードの集合の中から次の分枝ノードを選ぶために使用されるもの

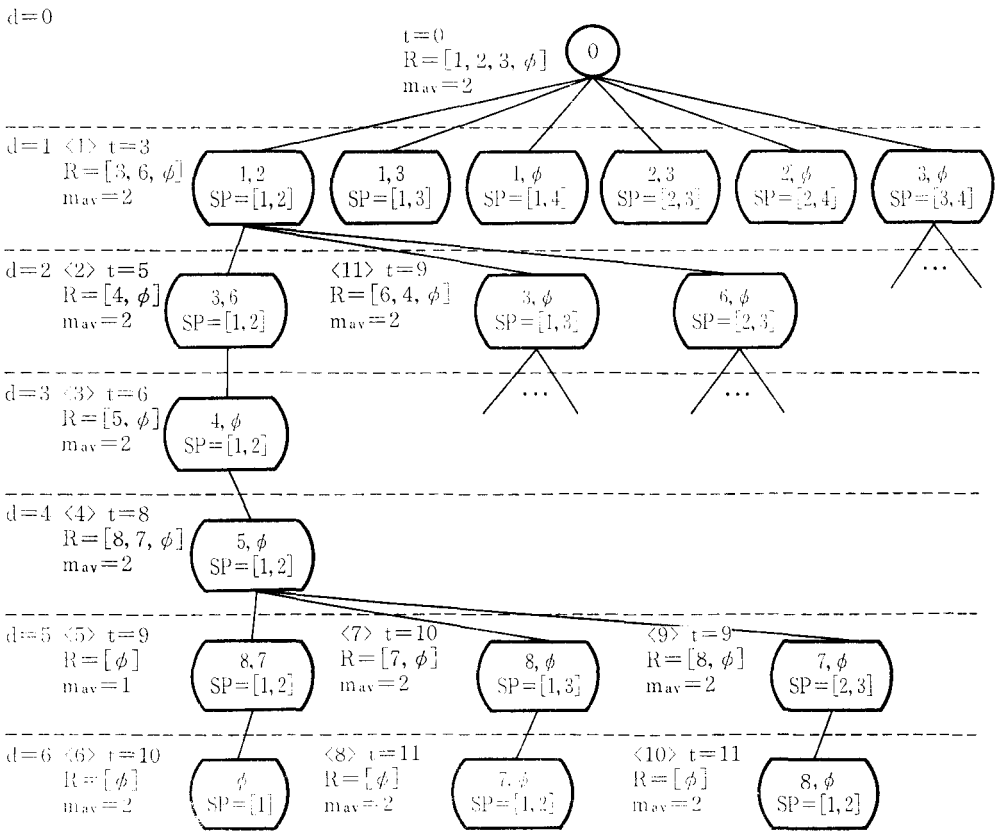

図 2 図1に対する分枝図

であり，DF/IHS では DF/FIFO という領域複雑度を非常に低く抑えることができるセレクション形態をとりながら DF/H と同様またはそれ以上の計算効率をあげる DF/IHR (DF/IH Rule) を用いている．DF/IHS はセレクション過程のみを見るとヒューリスティック値をまったく考慮せずに深さ最大のアクティブ・ノードを単に左から右へ選択を行なっているように見えるが，実は前処理におけるタスク・リナンバリングによりヒューリスティック的な意味が事前に与えられているため結果として DF/H と同様な効果を導くという方法である．このセレクションは前述のレディタスク・テーブル R とノード・セレクション・ポインタ SP を用いて行なわれる．SP は各分枝ノードにおけるアクティブ子ノードの生成，およびその選択を行なうために用いられるポインタであり，SP の値[2,4]は R 上で左側から2番目と4番目に並んでいるタスクを選ぶということを示している．この R と SP を用いるとノード・セレクションはアクティブ・ノードの中で深さ最大のノードの集合から SP の値が辞書式順序で最も小さいノードを選ぶだけで簡単に行なえ，その SP 値の変更（すなわちセレクション）に要する計算量は $O(m)$ である．また図2では，限定操作は行なわない場合の探索順序を〈 〉中に示している．この辞書式順序の選択により，通常の DF/H と同様な探索順序を実現できると同時に，初期解として現在最良のヒューリスティック解である CP/MISF の解が得られる．さらにこの R と SP の使用により DF/IHS の領域複雑度は $O(n^2+mn)$ に抑えることができる．

3.2.3 下限関数 L

分枝限定法においては枝の限定(除去)操作のために，精度が高く計算量の小さい下限関数の利用が重要である．DF/IHS では，タスク間に先行制約がないときの処理時間とプロセッサが非常に多いときの処理時間を利用した2種の単純な下限と Fernandez により拡張された Hu の下限[12]を変形して使用している．しかしここで，Fernandez の下限は時間複雑度が $O(t_{cr}(\pi_a))$ の擬多項式時間アルゴリズムとなってしまうので，計算回数を減じるために，限定操作においては計算量の小さい単純な下限を用いて限定できない場合にだけ Fernadez の

表 1　DF/IHS の性能

Number of tasks n (average)	Average number of arcs	$\varepsilon=0$		$0<\varepsilon\leq 0.05$		$0.05<\varepsilon\leq 0.1$		$0.1<\varepsilon$	
		Number of cases	t(s)	Number of cases	t(s)	Number of cases	t(s)	Number of cases	t(s)
5—20 (17)	31	155	6.5	0	——	0	——	0	——
21—40 (39)	77	25	1.8	1	6.1	1	0.1	0	——
41—60 (57)	97	26	0.2	2	0.1	3	0.1	0	——
80	148	25	8.7	3	3.1	7	11.0	0	——
100	179	59	3.3	18	3.9	12	0.2	0	——
200	346	36	1.9	21	1.1	2	3.6	0	——
total		226		49		25		0	
rate(%)		75.3		16.3		8.3		0	

Note : t represents average computing time by DF/IHS on HITAC M 280 H System.

下限を用いてテストしている．

3.2.4　上限値 U

上限値 U は通常最適解より大きい任意の値とするか $U=\infty$ とする．この U の値は探索中に U より小さい値が得られればその値に更新され，その値を $\hat{U}$ と表わす．U の値が最適解に近いほど，また $\hat{U}$ が探索の早い時期に最適解に近い値をとればとるほど，探索に要する時間が小さくなることが知られている．その点でも，本 DF/IHS は探索の最初に最適解またはそれにきわめて近い解が得られる可能性が高い CP/MISF 解が得られるため，計算時間の面から見ても優れたアルゴリズムであることがわかる．

3.2.5　除去（限定）規則 E

除去（限定）規則 E は，新しく生成された，あるいは現在すでにアクティブであるノードを L，U を使用して限定するための規則であり，DF/IHS ではチェックされるノードが表わす部分問題 π_a において，$L(\pi_a)\geqq U$ が成り立てば，そのノードは除去される．

3.2.6　許容近似精度 ε

DF/IHS では，暫定解 $\hat{U}$ を最適解 t_{opt} との相対誤差 $\varepsilon(0\leqq\varepsilon\leqq 1)$ 以下に厳密に抑えることが可能である．

$$(\hat{U}-t_{opt})/t_{opt}\leqq\varepsilon_o$$

具体的には任意の ε が与えられた場合，$\hat{U}$ の更新時に新しい上限値 $\hat{U}_\varepsilon$ を $\hat{U}_\varepsilon=\hat{U}/(1+\varepsilon)$ とし，$\hat{U}_\varepsilon$ より小さい解が見つかるまで探索を続ける．もしそのような解が見つからずに探索が終了すれば，現在の暫定解 $\hat{U}$ の相対誤差が許容近似精度 ε 以下であることが確認されるわけである．$\varepsilon=0$ の時は当然最適解が得られるが，もし $\varepsilon=0$ として，許容計算時間内で終了しない場合には，ε を 0.05～0.1 とすれば計算時間が顕著に改善されることがわかっている．

4.　性能評価

本章では DF/IHS の性能について述べる．ここでは，タスク数5～200の300ケースに対して適用し，評価を行なった．各テスト問題はタスクグラフ形状をランダムとする，縦長・横長にする，類似したパターンを縦横に繰り返す，プロセッサ数 m を2～10の間で変える，t_i を1～3 $[u.t]$，1～9$[u.t]$ のような範囲でランダムに発生させる等というように考えられうる種々の場合を考慮にいれて作成しているほか，ヒューリスティックが最悪の性能を示すような場合についてもチェックを行なっている．これらの検討結果を**表1**に示す．表1における検討は，日立大型計算機 M 280 H を用いて行なっており，CPU タイム・リミットを180秒としている．$\varepsilon=0$ は最適解が得られたことを表わしており，$0<\varepsilon$ の欄に示されている例は $\varepsilon=0$ ではタイム・リミット・オーバーとなり最適解を得ることができなかった（正確には，暫定解が最適解である場合もあるので断定できない）が精度 ε の近似解が求まったことを示している．表1を見ると全体の約75％のケースにおいて最適解が求まり，その計算に要する時間も平均して数秒のオーダーであることがわかる．また全体の92％のケースにおいて最適解から5％以下の誤差で解が求まっており，許容誤差を10％以下とすると

300例すべてがこの範囲に含まれるという優れた性能を示している．またここで，このスケジューリング問題が$t_i=1\,(i=1,\cdots,n)$と固定してもNP困難であり，さらに$m=2$と固定し$t_i=1$ or 2としてもなおNP困難であること，またDPを使用した場合，$m=2$と固定しても$n=40$程度で求解不能であった[9]ことを考え合わせると，この結果がいかに優れたものか推察できよう．

5. むすび

本解説では，実行時間最小マルチプロセッサ・スケジューリング問題に対する，DF/IHS法と呼ぶ一種の分枝限定法の適用について述べた．分枝限定法は，対象問題の特徴を適切に押さえたヒューリスティックをうまく取り入れ，計算時間および記憶領域を低く押さえるようなインプリメントを行なえば，NP困難な最適化問題に対しても実用的な意味で最適解を求めることができる強力な手法であることがおわかりいただけたと思う．また，DF/IHS法はここで述べた実行時間最小マルチプロセッサ・スケジューリングだけでなく，トータル加重フロー時間最小マルチプロセッサ・スケジューリング等，他の目的関数のスケジューリング問題の求解にも有効であることが確かめられている[10]．

文　献

[1] Coffman, E. G.: "Computer and Job-shop Scheduling Theory", John Wiley & Sons (1976).

[2] Lenstra, J. K. and Kan, A. H. G. R: "Complexity of Scheduling under Precedence Constrains", Oper. Res., 26, 1, pp. 22-35 (Jan. 1987).

[3] Garey, M. R. and Johnson, D. S.: "Computers and Intractability, A Guide to the Theory of NP-Completeness", Freeman, San Fransisco (1979).

[4] 茨木俊秀："組合せ最適化"，産業図書(昭58).

[5] Kasahara, H. and Narita, S.: "Practical multiprocessor scheduling algorithms for efficient parallel processing", IEEE Trans. Comput. C-33, 11, pp. 1023-1029 (Nov. 1984).

[6] Kasahara, H. and Narita, S.: "Prallel processing of robot-arm control computation on a multimicroprocessor system", IEEE J. of Robotics and Automation, *RA*-1, 2, pp. 104-113 (June 1985).

[7] Kasahara, H. and Narita, S.: "An approach to supercomputing using multiprocessor scheduling algorithms", in Proc. IEEE 1st Int. Conf. on Supercomputing Systems, pp. 139-148 (Dec. 1985).

[8] Kasahara, H. et. al.: "A parallel processing scheme for the solution of sparse linear equations using static optimal-multiprocessor-scheduling algorithms", in Proc. 2nd Int. Conf. on Supercomputing, pp. 433-442 (May 1987).

[9] Ramamoorthy, C. V. Chandy, K. M. and Gonzales, M. J.: "Optimal Scheduling Strategies in a Multiprocessor System", IEEE Trans. Comput., C-21, 2, pp. 137-146 (Feb. 1972).

[10] 笠原，和田，甲斐，成田："トータル加重フロー時間最小化問題に対する DF/IHS の応用" 信学論 (D), J70-D, No. 6, pp. 1083-1091 (62-06).

[11] Kohler, W. H.: "Chracterization and Theoretical Comparison of Branch-and-Bound Algorithms for Permutation Problems", J. Assoc. Comput. Mach., 21, 1, pp. 140-156 (Jan, 1974).

[12] Fernandez, E. B. and Bussel, B.: "Bounds on the Number of Processors and Time for Multiprocessor Optimal Schedules", IEEE Trans. Comput., 22, 8, pp. 745-751 (Aug. 1973).