並列分枝限定法による混合整数計画問題の解法

川岸太郎
(財) 新世代コンピュータ技術開発機構 (ICOT)
e-mail: kawagish@icot.or.jp

1 はじめに

整数計画問題の厳密解法の中で制約の構造に依存せずに、広範囲の問題クラスに有効な方法として分枝限定法がある。本研究では分枝限定法による混合整数計画問題の解法を取り上げ、その探索を分枝毎に並列に実行することにより規模の大きな問題を高速に解くことを目標として、ICOT で開発された KL1 言語を用いて並列プログラムを作成し、並列計算機 Multi-PSI 上で実験を行った。以下問題の定式化と逐次アルゴリズムを示し、次にそのアルゴリズムの並列化について述べる。

2 問題の定式化

次の最適化問題を考える。

Problem – ILP
実数変数 x_j と整数変数 y_j とに関する目的関数

$$z = \sum_{j=1}^{n_1} p_j x_j + \sum_{j=1}^{n_2} q_j y_j$$

を、次の線形制約条件の下で最小化する：

$$\begin{array}{l}
\sum_{j=1}^{n_1} a_{ij} x_j + \sum_{j=1}^{n_2} b_{ij} y_j \geq e_i,\ i = 1, \ldots, l, \\
\sum_{j=1}^{n_1} c_{ij} x_j + \sum_{j=1}^{n_2} d_{ij} y_j = f_i,\ i = 1, \ldots, m, \\
x \in R^{n_1},\ x_j \geq 0,\ j = 1, \ldots, n_1, \\
y \in Z^{n_2},\ y_j \text{は整数},\ j = 1, \ldots, n_2, \\
l_j \leq y_j \leq u_j,\ j = 1, \ldots, n_2,\ l_j, u_j \in Z, \\
a_{ij}, b_{ij}, e_i, c_{ij}, d_{ij}, f_i \text{は実数定数。}
\end{array}$$

実際問題では整数変数 y_j の値は 0, 1 のみを取る場合が多いが、ここでは一般的な場合を扱っている。

3 逐次分枝限定法による解法

上で与えた混合整数計画問題を解くための準備として、問題 ILP の整数変数を実数変数に置き換えた連続緩和問題 LP を考える。

アルゴリズム
LP の解はシンプレックス法で求めることができ、その整数変数が整数値を取る場合にはそれで解が決定する。そうでない場合には仮に整数変数 y_s の取る値が非整数値 $\bar{y}_s$ だったとして、その両隣の整数を区間境界とする二つの異なる制約条件 $l_s \leq y_s \leq [\bar{y}_s]$, $[\bar{y}_s] + 1 \leq y_s \leq u_s$ を

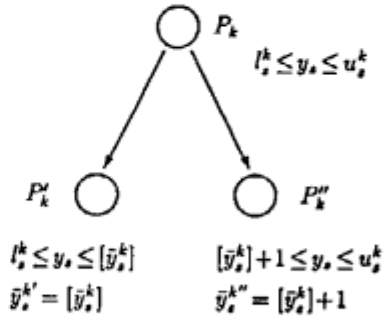

図 1: 分枝ノードの生成

既にある制約に付け加えて、元の問題を二分する。この手続き (分枝操作) を繰り返し行い探索空間を分割して行くことによって、部分問題の制約条件はその度ごとに区間制約が追加されて、区間 $([\bar{y}_s], [\bar{y}_s] + 1)$ の曖昧さが除かれ、ある深さに達すると連続解が整数解となるものが得られる。

さて問題空間を上のように分割するだけでは整数値の列挙に過ぎないが、元の部分問題の整数最適解は常に連続緩和問題の最適解よりも目的関数値が良くないことに注意すると、連続解の目的関数値が既に探索過程で得られている暫定的整数解のものよりも良くない部分問題は最適値を与え得ず、その分枝は探索不要となって、探索空間を小さくできることが分かる (限定操作)。

この様にして分岐と、目的関数値による分枝の限定操作を繰り返して最適解を見つけるのが分枝限定法である。分岐して得られる分枝を探索ノードを呼ぶことにする。

探索木の辿り方を決定する重要な要素として、(1) 生成されたノード集合からのノードの選択、(2) 分枝操作を行う変数の選択、の二つがある。これらの選択は、全探索空間内のなるべく小さな部分の探索で最適解が見つかる様になされることが望ましい。ここでは OR において効果的と考えられている分岐ヒューリスティックの最良の一つを用いる ([今野、鈴木 82])。(1) は深さ優先則とヒューリスティック評価値優先則の両方を合わせた方法を用いる。ヒューリスティック評価値は連続緩和問題の解の近傍で、整数変数を変化させた時の連続最適解の変化率で記述される。(2) も同様の評価値を用いて決められる。

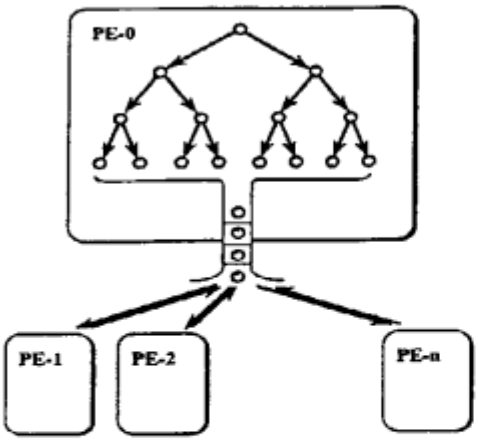

図 2: 並列プロセスの生成と負荷分散

4 分枝限定法の並列化

分散メモリの並列マシン Multi-PSI 上で並列処理を行うことを目的として、上で述べた逐次アルゴリズムの並列化を考える。逐次アルゴリズムにおいて生成される分枝を、並列プロセスとして複数のプロセッサに割り振り、各プロセッサが同時に逐次アルゴリズムと同様の枝刈りをしながら探索を進める。

負荷分散については、逐次アルゴリズムにおける探索木を一定の深さまで分岐して生成される子ノードを、サイクリックに順次異なるプロセッサに投げる。各プロセッサは負荷が均等になる様に、複数の子ノードを割り当てられ、それら子ノードに対してそれ以降の深さの探索を逐次に行う。並列プロセスへの分散を探索木の一定の深さで打ち切るのは、仕事の粒度が小さくなってプロセッサ間通信のコストが増えるのを避けるためである。

ノード選択および分枝変数選択のヒューリスティックについては、一つ一つのプロセッサ内では逐次アルゴリズムと同じものを用いるが、プロセッサには複数の子ノードが割り当てられるので、それらのノードの評価値によって選択順位を付ける。ノード選択については深さ優先と評価値優先の双方を用いるが、ノードの評価値に従った戦略に関しては、KL1 言語が提供している実行ゴール・プライオリティ制御機能を用いて実現する。(KL1 言語ではプログラム中の任意のゴールに対して、実行優先度を予め割り当てることができる [Ueda-Chikayama 90])。

また探索領域を分けて仕事を分散させるのと同時に、一つのプロセスで得られる探索情報を他のプロセスに生かして枝刈りをさせることによって、全体での枝刈りが促進される。つまり並列に走っている複数個の逐次探索プロセスの間で、暫定最適解を大域データとして転送し合い、できるだけ最新の暫定解によって枝刈りを行うようにする。これは大域データを転送するゴールの優先度を、実際に探索を行うゴールよりも高くすることによって実現している。

5 実験結果

ジョブショップ・スケジューリング問題を例題として実験をした。4 ジョブ 3 機械の問題に対してプロセッサ台数の増加による速度向上、および最適解を得るまでに探査したノード数を次に上げる。

プロセッサ台数	1	2	4	8
速度向上	1.0	1.5	1.9	2.3
探索ノード数	242	248	395	490

分散メモリマシンでは予想されることだが、複数プロセッサに分散させた探索プロセス間でのメッセージ転送に時間が掛かるために、暫定解が転送されて実際に枝刈りが始まるまでに、転送時間が 0 であったなら枝刈りされていたはずのノードの探索が進んでしまっていることが、探索ノード数から分かる。

このように複数台で計算した場合に、1 台で計算したときよりも最終解を得るまでに要する探索空間が増大してしまうのであるが、これを如何に小さく抑えるかが探索問題に対する並列プログラミングにおける問題と言える。

6 まとめ

一般的に探索問題に対するアルゴリズムは、探索の経過に依存したヒューリスティックを用いて枝刈りを行い、実際に探査する探索空間の縮小を計るので、並列プログラムにおいて逐次アルゴリズムで用いているヒューリスティックをうまく生かすためには、負荷をプロセッサに適切に分散すること、そして枝刈り情報をプロセッサ間で高速に転送することが重要なこととなる。本研究では、このことを考慮して分枝限定法による混合整数計画問題の解法を分散メモリ並列マシン上で実現し、探索の高速化が実現できることを示した。

謝辞

本研究を行うに当たり、研究の機会を与えて下さった相場亮 ICOT 第 4 研究室長代理、有益な助言を下さった東芝情報処理・機器技術研究所の永井保夫氏および ICOT 第 7 研究室の清慎一氏、市吉伸行氏に感謝します。

参考文献

[Benichou et al 71] M.Benichou, L.M.Gauthier, P.Girodet, G.Hentges, G.Ribiere, O.Vincent, Experiments in Mixed-Integer Linear Programming, *Mathematical Programming* 1 (1971) 76-94.

[今野、鈴木 82] 今野、鈴木編. 整数計画法と組み合わせ最適化, 日科技連,1982 年.

[沖 他か 89] 沖廣明、瀧和男、清慎一、古市昌一. Multi-PSI における並列詰め碁プログラムの実現と評価、並列処理シンポジウム JSPP 1989, 351-357.

[Ueda-Chikayama 90] K.Ueda and T.Chikayama. Design of the kernel language for the parallel inference machine, *The Computer Journal*, December 1990.