2001年度日本オペレーションズ・リサーチ学会
秋季研究発表会

並列分枝限定法による混合整数計画問題解法

01307380 (株) 数理システム　　*田辺隆人 TANABE Takahito
(株) 数理システム　　望月公晴 MOTIZUKI Masaharu
(株) 数理システム　　逸見宣博 HENMI Nobuhiro

1 概要

運転計画問題，施設配置問題，生産計画問題などは，一般に混合整数計画法によって定式化される．混合整数計画法は分枝限定法によって解かれる．分枝限定法は問題の構造を生かした下界値計算，分枝変数選択，部分問題求解手法の工夫，あるいはヒューリスティクスの導入など，計算の効率化に向けた研究が盛んであり，数理モデリング技術の進歩や計算機性能の向上とあいまって，混合整数計画問題の実務レベルへの応用の可能性を拡げている．

しかし，本質的に分枝限定法は組みあわせの列挙を伴う数え上げアルゴリズムであるため，しばしば問題規模が大きくなると，計算量が爆発し，現実的な時間内では解を得ることができないという状況が生じる．

この状況を改善する一つの方策として古くから提案されているのが，分枝限定法の並列性に着目した並列化である．分枝限定法では列挙された子問題は通常リストに蓄積され，それぞれについて

- 下界値計算
- 限定操作
- 分枝変数の選択と分枝操作

なる処理が行われることになるが，リストに蓄積されたすべての部分問題についてこの処理が独立に実行できることに着目すれば並列化が可能である．本稿では，汎用パッケージである NUOPT を用いた分枝限定法の並列化について，その実装と実験結果を報告する．

2 設計

2.1 ハードウエア/ソフトウエアモデル

並列計算機のハードウエアモデルとして，独立したメモリを持つプロセッサが，TCP/IP 接続のような比較的低速なネットワークによって継がっている環境を想定する．共有メモリや各プロセッサの同期を取ることが可能な，並列化に特化した機構は前提としない．

このようなハードウエアモデルを前提とした設計としておくことにより，オフィスにおいて，夜間アイドル状態の PC を利用して計算を行うなどの利用機会が生まれる．

並列化を実現するソフトウエアとしては，CORBA を利用した．CORBA は TCP/IP など低い階層のインタフェースを隠蔽し，プロセッサ間のデータやメッセージのやりとりについて C++ 言語レベルでのインタフェースを供給するものである．

CORBA は通信の高速性については期待できないものの，マルチプラットフォームで動作する安定な実装が存在すること，C++ をベースとした我々のプログラミング環境への親和性が良いことなどから，ハードウエアモデルに沿った選択と言える．

2.2 利用形態と粒度の選択

過去の研究を見ても分枝限定法における並列性の利用形態，並列化する処理の粒度の設定には，様々な可能性があり得るが，我々の前提としているハードウエアモデルではプロセッサ間でのデータ受渡しのコストが高いため，おのずと選択の範囲は限られてくる．すなわち，各プロセッサが行うべき処理の粒度を粗くし，プロセッサ同士のコミュニケーションの頻度を下げたものとなる．

したがって，単一の子問題の内部の処理 (例えば下界値計算) を並列化するような粒度の細かな設計は排除され，少なくとも各子問題単位の処理は各プロセッサが並行して行うようにする必要がある．

しかし通常の逐次演算コードによる混合整数計画法についての次の実験から見るように，一般に混合整数計画法の分枝限定法の子問題数は大規模な問題で数万個以上となり，一問あたりの処理時間は平均して１秒以下である．

問題名	n	m	#子問題	計算時間 (秒)
MIP1	840	1192	27937	13712
MIP2	10724	2054	43154	7032
MIP3	378	234	1602351	12843

使用マシン：Sparc Station 10,
使用ソフトウエア: NUOPT

この状況を見ると，プロセッサ毎の処理の粒度を子問題単位としても，コミュニケーションコストが比較的大きく，我々の前提としているハードウエアモデルでは十分に性能を発揮できないことが予想される．

したがって，我々は分枝木の枝以下に生じるすべての子問題の処理を各プロセッサに割りあてて，処理の粒度をさらに粗くするようにアルゴリズムを設計した．具体的には，各プロセッサがそれぞれ子問題のリストを持って，独自にリストの中からの子問題の選択と分枝，選択した子問題の終端操作，分枝操作を行う．分枝した子問題は各自の保有するリスト中に登録する．各プロセッサを監視するマスタープロセスを立てて，マスタープロセスが，以下のいずれかのイベント:

- いずれかのプロセッサにおける実行可能解の発見 (上界値の更新)

・各プロセッサのリスト中の問題の数や質のかたより

の発生を検知してはじめて，各プロセッサがそれぞれのリスト中の子問題情報を交換することとする．この方法で，コミュニケーションコストを減少させ，我々のハードウエアモデルの弱点をカバーすることができる．また，各プロセッサで独立して行う分枝限定法には，逐次演算コードをそのまま用いることが可能なので，逐次演算コードに対するチューニングの成果が利用できることが期待できる．

3 実装

本システムの目的は逐次コードと比較して並列化によるパフォーマンス向上を得ることである．我々は比較対象を既に単一プロセッサでチューニングされた汎用パッケージ NUOPT に設定したが，混合整数計画法の分枝限定法全体の性能は部分問題の選択や下界値計算の際の部分問題求解の効率性に非常に敏感であることから，注意深い実装が要求される．

次の例は混合整数計画法に対する分枝限定法の逐次コードの実行時間の比較である．実装 1 に対して実装 2 は問題選択のヒューリスティクスや先行する部分問題の情報 (基底や冗長変数，冗長制約式) の利用を促進したものである．

	n	m	実装 1 (秒)	実装 2 (秒)
IPa	201	134	42	24
MIPb	160	97	130	17
MIPc	870	781	250	17

使用マシン：PentiumIII 400MHz,
使用ソフトウエア: NUOPT

すなわち，同じ逐次コードでも部分問題の選択や求解方法の良し悪しが数倍程度のパフォーマンスの差を招く．

我々の設計では，子問題を求解する順番は逐次コードと全く同じとなることは保証されず，部分問題の選択ヒューリスティクスの働き方も変化してくる．並列化することにより加速率が計算機台数の逆数を越える (効率異常加速 [1]) などの好ましい現象も期待できる．しかし，もし並列化によって，部分問題の選択が悪い方に変わった場合，あるいは，部分問題の求解性能が下落した場合には，数台のプロセッサを利用することによる並列化効果を打ち消してしまう程度のパフォーマンスの下落が生じる可能性があることを上記の実験は示している．

このことから，並列分枝限定法のコードの性能を検証するにあたっては，並列性度を上げたときのスケーラビリティのみならず，単一のプロセッサでチューニングされた逐次コードとのパフォーマンスを比較する必要があることがわかる．我々は実装と実験の過程での比較対象として，NUOPT を用いた．チューニングは各問題に対して次をモニタしながら行った．

- 全体の子問題数
- 各プロセッサで処理される子問題数
- 各プロセッサのアイドル時間
- 部分問題求解に所要する単体法ピボット回数
- 通信に所要する時間

アイドル時間は，問題の分配の良し悪しを判定する指標となる．我々が採用したプロセッサ間での子問題分配アルゴリズムは次の通りである．

- 各プロセッサ p のリスト先頭にある N 個の子問題の中で最良の推定値を持つもの B_p，最悪の推定値を持つもの W_p を調べる．
- 全プロセッサ p で $\bar{f}(W_p)$ が最良になるプロセッサ p_b から $\bar{f}(B_p)$ が最悪であるプロセッサ p_w に $N/2$ 個の子問題を分配する．

なお，N はパラメータで，$\bar{f}(A)$ は問題 A の推定値を示す．このアルゴリズムは，各プロセッサにある問題の質 (推定値) を揃えることを意図している．リスト中の子問題数を意識した分配アルゴリズムも考えられるが，これは複数のプロセッサのリストが空になった場合にのみに利用を限っている．質を優先して問題を配分する理由は，良い可能解の出現によって，多数の問題が一気に終端するなど，リスト中の問題数の変化が均等ではないという観測による．

通信に所要する時間はアルゴリズムの性質から予想できる通り，無視できるオーダーとなることがわかった．

次においては逐次コードのヒューリスティクスや実装方法に沿った実装を行った．

- 部分問題の選択
- 問題をプロセッサ間で受けわたすときの基底情報の再利用様式

これらは逐次コードに対するパフォーマンスを出す上で非常に重要であることが，経験的に確かめられた．

4 計算結果

以下は本並列化実装によって miplib 問題を解いた一例である．実験には Linux の PC クラスタを用いて，プロセッサ数 4,16 の場合の，逐次コードからの加速率 r_4,r_16 を測定した．

問題名	n(nint)	m	逐次 (秒)	r_4	r_{16}
pk1	86(86)	45	1957	3.0	9.7
misc07	260(259)	212	815	4.0	7.9
gesa2_o	1224(720)	1248	1500	3.2	8.2
pp08aCUTS	240(64)	245	23177	4.3	18.8
vpm1	378(168)	233	8524	3.5	13.0

CPU：PentiumIII 730MHz メモリ:256M
n：変数総数 nint:整数変数総数 m:制約式総数

その他の実験例については当日会場にて示す．

参考文献

[1] 品野 勇治, 桧垣 正浩, 平林 隆一，並列分枝限定法の可能性について, 第 6 回 RAMP シンポジウム予稿,17-32,1994.

[2] 茨木俊秀，福島雅夫 *FORTRAN77* 最適化プログラミング, 岩波書店,1991.