分枝限定法の探索方法に関する研究

浅野　京介　（沼田一道助教授，桧垣正浩助手）

1．はじめに

整数計画問題や混合整数計画問題などは，一般に簡単には解く事が出来ない．分枝限定法は，これら組合せ最適化問題に対する代表的な解法である［4］．

様々な問題に対して，分枝限定法を用いた解法が提案されており，近年の研究により比較的大きな規模の問題も解けるようになってきた．これらの研究の多くは，問題の特徴を生かした下界値（上界値）の計算法を中心としている［2］［3］．しかし，現実の規模の問題を解くには，まだまだ計算時間，記憶容量の面からも効率のよい解法が望まれている．

2．研究目的

分枝限定法において，下界値計算（上界値計算），探索方法，分枝変数の選択などが効率を左右する要因となっている．しかし多くの研究は下界値（上界値）の改良に費やされ，ヒューリスティックな部分である探索方法や分枝変数の選択は，あまり研究の対象となっていない．

そこで，効率を考える上で重要であるが，あまり研究されていない，より良い探索方法の選択を行うための研究をナップサック問題を例として行う．本研究では，下界値（上界値）計算方法を固定し，探索方法を変えることによっての演算時間の違い，必要な記憶容量の違いを調べ，効率的な計算をするためのより良い探索方法の選択方法を実験的に比較し検討する．

3．分枝限定法

分枝限定法を，最小化問題を対象として，説明する．

直接的に解くことが困難な整数計画問題（P_0）が与えられたとき，適当な条件によって（P_0）の実行可能領域X_0をいくつかの部分領域$X_1, X_2, \ldots, X_k$に分解し，その各々に対応する問題（P_j）（j=1,2,...,k）を考える（これを（P_0）の子問題という）．このk個の子問題を解いて得られた値の中で最も良い値を見つければ，問題（P_0）の最適値が得られたことになる．また，子問題（P_j）がまだ解くことが困難な場合，X_jをさらにいくつかの部分領域に分け，（P_j）を分解する．この手順を繰り返し適用すれば，いつかは直接解ける問題に到達し，それらを解いて得られた解を統合すれば（P_j）の最適値が得られ，結局（P_0）が解け，最適解を求めることが出来るというのが分枝操作と呼ばれる分枝限定法の基本的な考え方である．

しかし，分枝操作だけでは，問題の規模が大きくなると，実用時間内で解けなくなるため，探索の必要のない部分問題を探す．そこで，ある子問題（P_j）がもとの問題（P_0）の最適値を与えないことが何らかの理由によって判断出来れば，（P_j）をさらに分枝する必要はないことを利用する．この操作は，限定操作と呼ばれている．

限定操作の具体的手順として，下界値（または上界値）によるものがある．ある整数計画問題を（P_j）とすると，直接解くことが出来ないため制約条件を緩和した問題を考える．実行可能領域X_jを完全に含むある集合X_j'に対する問題（P_j'）は（P_j）の緩和問題という．ここで（P_j）と（P_j'）の間には次の関係が成立する．

（1）（P_j'）が実行可能解を持たないとき（P_j）も実行可能解を持たない．

（2）(P_j') の最適解を $\bar{x}$ としたとき $\bar{x} \in X_j$ なら $\bar{x}$ は (P_j) の最適解である．

（3）最小化問題における (P_j') の最適値 $\bar{z}$ と (P_j) の最適値 z^* の間には $\bar{z} \leqq z^*$ が成立する．この $\bar{z}$ を (P_j) の下界値という．

したがって緩和問題 (P_j') を解いた結果，（1）または（2）が成立すれば (P_j) も解けたことになる．またそれ以外は，（3）が成立するため (P_j') の最適値を (P_j) の下界値とし，他の子問題から得られている元問題 (P_0) の実行可能解の目的関数値 z' （z' を暫定値という）より良くなければ，それ以上分枝する必要はない．

4．探索方法

本章では，分枝限定法における様々な探索方法を説明する［1］［4］．ここで，説明の都合上，最小化問題を扱う．

4．1　深さ優先探索

分枝する必要があり，まだ分枝されていない頂点の集合である未解決の頂点集合に最も新しく加わった頂点を次の分枝頂点とする方法である．よって，早い段階で実行可能解が得られる可能性は大きいが，目的関数の値をあまり考慮せずに分枝頂点を選択する．また，未解決の頂点集合があまり大きくならないという長所がある．

4．2　幅優先探索

未解決の頂点集合の中で深さ最小の頂点を次の分枝頂点とする方法である．一般に早い段階で実行可能解が得られず，未解決の頂点集合が大きくなってしまう．しかし，深さ優先探索での最悪の探索をした場合に比べると，早く最適解を見つけることが可能である．

4．3　下界値（上界値）優先探索

未解決の頂点集合の中から下界値の最も小さいものを次の分枝頂点とする方法である．これは，全ての未解決の頂点の中から最も良い実行可能解を生成する公算の高いものを選んでくるため，思惑どおり良い実行可能解が求まれば，暫定値が改善されて未解決の頂点集合の要素数が一挙に縮小される可能性がある．しかしその一方で，もし良い実行可能解が生成されないと一般に未解決の頂点集合が大きくなるという欠点がある．

4．4　混合探索

どの探索方法にも一長一短があるため，実際には，それぞれの長所を取り入れた混合探索が考えられている［4］．一般に，最初に深さ優先探索でいくつかの暫定解を見つけてから，下界値（上界値）優先探索に切り換える方法が知られている．

5．ナップサック問題と緩和問題

ナップサック問題は，最も単純な0－1整数計画問題である．x_j を0－1変数，nを変数の数，c_j を目標関数の係数，bを制約の右辺定数，a_j を制約の係数とすると次のように定式化できる．

$$\text{目標関数}\quad z=-\sum_{j=1}^{n} c_j x_j \rightarrow \text{最小化} \qquad (1)$$

$$\text{制約条件}\quad \sum_{j=1}^{n} a_j x_j \leqq b \qquad (2)$$

$$x_j \in \{0, 1\}\quad j=1, \ldots, n \qquad (3)$$

ここで，係数c_j，a_j，b はすべて非負である．

下界値（上界値）計算には，制約条件（３）を$0 \leqq x_j \leqq 1$ に緩和した連続緩和問題を用いる．この連続緩和問題は，貪欲解法で簡単に解ける．

６．試作したシステム

各々の探索方法を比較，評価するためのシステムを試作した．図１に試作したシステムの概要を示す．多くの問題に対応できるように解く問題の問題構造に依存する部分と依存しない部分に分離して設計してある．

本システムの対応できる探索方法は，深さ優先探索，幅優先探索，下界値（上界値）優先探索，深さ優先探索から下界値（上界値）優先探索に切り換える探索方法である．解析対象のプログラムは，システム側にノードの個数，最小節点番号，下界値（上界値）を与える．また，システム側から指定した探索方法における次に解くべき部分問題が指示される．

出力項目は，それぞれの探索方法に対して，①実行時間，②解いた部分問題の数，③メモリ使用量である．

実行時間：問題を解くのにかかったＣＰＵ時間．

解いた部分問題の数：実行時間は，解いた部分問題数に大きく影響される．下界値（上界値）の計算回数を示す．

メモリ使用量：分枝限定法において，未解決の部分問題を記憶しなければならない．実際に問題を解くためには，未解決の部分問題が最も多くなったときにメモリに入らなければならない．メモリ使用量とは，この部分問題数の最大値を示す．

この出力結果により，解きたい問題における各探索方法の有用性を判断するための役立てることが出来る．

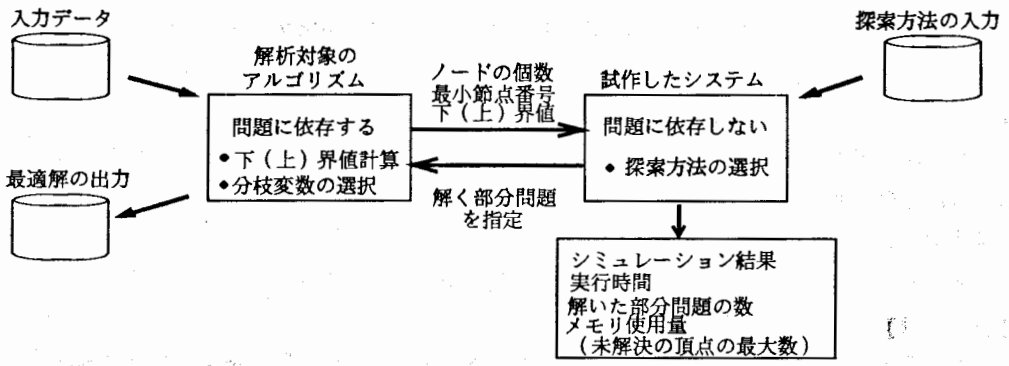

図1．試作したシステムの概要

７．数値実験

数値実験は，ナップサック問題の変数の数が２５０，５００，７５０個であり，目標関数の係数を０から１００，制約式の係数を０から１００の一様乱数で発生させ，制約式の右辺定数を２０００として１０問ずつ作成して行った．実験結果を表１，表２に示す．使用したプログラム言語はＣ言語であり，コンピュータはＯＭＲＯＮ社のＬＵＮＡ８８Ｋである．

表１．演算時間の平均の結果（単位：秒）

変数の数	深さ優先	幅優先	下界値優先	混合（１）	混合（２）
２５０	６．２１	—	２．３２	２．５７	３．０９
５００	１５．８１	—	６．２２	７．６４	９．０２
７５０	４５．７４	—	１３．１４	１６．０５	１８．７５

表２．未解決の頂点の最大数の平均の結果（単位：個）

変数の数	深さ優先	幅優先	下界値優先	混合（１）	混合（２）
２５０	１９２．３	—	４１０．１	４７５．８	４６１．５
５００	３８６．６	—	８５４．２	９８１．４	９４１．０
７５０	６４３．４	—	１１７５．０	１２８６．５	１３０８．０

表中の数字は，それぞれ１０問の平均を表している．混合（１）は深さ優先から暫定解を１つ見つけると下界値（上界値）優先に切り換わるもので，混合（２）は暫定解を２つ見つけて切り換わるものである．— はメモリが足らずに最適解を見つけることが出来なかったものである．

演算時間に関しては，下界値優先探索が一番優れており，深さ優先探索のおよそ１／３の演算時間で最適解が求められている．メモリの使用量に関しては，深さ優先探索が優れている．下界値優先のおよそ半分である．

また，深さ優先から下界値優先探索に切り換えたものは，下界値優先探索を演算時間においてもメモリの使用量においても良い結果がでなかった．ナップサック問題において，連続緩和問題を用いて下界値（上界値）計算を行う方法では，この混合探索はあまり有効でないと思われる．

８．おわりに

探索方法を変えることによっての演算時間の違い，必要な記憶容量の違いを調べるためのシステムを試作した．本研究で試作したシステムによって，各々の探索方法における演算時間，メモリの使用量（未解決の頂点の最大数），解いた問題の数を知ることが出来た．そのために，実験的に各々の探索方法の特性が分かり，ナップサック問題において探索方法を選択する目安となった．しかし，幅優先探索は，メモリが足らずに特性を実験的に見ることが出来なかった．これは，ナップサック問題において，分枝限定木の浅いノードで暫定解を見つけることが出来なかったため，限定操作が行えなかったからである．

また，深さ優先探索から下界値（上界値）優先探索に切り換える探索方法は，両者の長所を生かした方法と言われているが，ナップサック問題においてはあまり有用ではないことが分かった．

本研究で試作したシステムでは，分枝限定木の管理が問題に依存する側にもあるため，解析対象のアルゴリズムをコーディングする際に制約が強くかかってしまうため，今後改善が望まれる．

【参考文献】

［１］茨木　俊秀：”組合せ最適化－分枝限定法を中心として－”，講座・数理計画法８，産業図書，１９８３．

［２］桧垣　正浩，品野　勇治：”分枝限定アルゴリズムの可視化及び解析ツール”，電子情報通信学会春季大会講演論文集，１９９３．

［３］桧垣　正浩，品野　勇治：”分枝限定アルゴリズムの可視化及び解析ツールの試作”，電子情報通信学会秋季大会講演論文集，１９９３．

［４］今野　浩，鈴木　久敏：”整数計画法と組合せ最適化”，ＯＲライブラリー第７巻，日科技連，１９８２．