多次元ナップサック問題に対するRSB法の適用
～有効勾配法との比較～

小沢　和俊　（沼田一道　助教授）

1．はじめに

有限個の組合せの中から与えられた制約を満たす最適な組合せを求める問題を，組合せ最適化問題という．多くの組合せ最適化問題に対しては，能率のよい（全列挙の指標を含まない）厳密解法が知られていない．問題の規模が小さい場合には全数探索によって厳密解を求めることも可能であるが，問題の規模が大きくなると，組合せ数が爆発的に増大し，厳密解を求めるのは困難になる．そこで，できるだけ良い解をなるべく少ない計算量で求めようとする近似解法が必要になる．

本研究では，そのような問題の例として多次元ナップサック問題を取り上げ，それに対して新しい近似解法の枠組みである RSB 法を適用する．既存の有力な近似解法である有効勾配法と比較して，RSB 法の性能を評価する．

2．ナップサック問題

n 個のプロジェクトがあり，「資源制約の下で利益を最大化するにはどのプロジェクトの組合せを採用すればよいか」という問題がナップサック問題である．本研究では 0-1 ナップサック問題と多次元ナップサック問題を扱う．

2.1　0-1 ナップサック問題

n 個のプロジェクトがあり，$x_j=1$ の時プロジェクトを実施，$x_j=0$ の時プロジェクトを実施しないとして，c_j をプロジェクト j を実施することにより得られる利益，a_j をプロジェクト j が必要とする資源の量，b を資源の利用可能量とすると，0-1 ナップサック問題は（P）のように定式化される．

2.2 多次元ナップサック問題

0-1 ナップサック問題で制約式が複数，つまり a_{ij} をプロジェクト j が必要とする資源 i の量，b_i を第 i 資源の利用可能量とした問題である．多次元ナップサック問題は（Q）のように定式化される．

$$(P)\begin{cases} \text{最大化} \quad z = \sum_{j=1}^{n} c_j x_j \\ \text{制約} \quad \sum_{j=1}^{n} a_j x_j \le b \\ x_j \in \{0,1\}, \\ c_j > 0, \quad a_j \ge 0, \quad b > 0 \\ j = 1,2\ldots,n \end{cases} \qquad (Q)\begin{cases} \text{最大化} \quad z = \sum_{j=1}^{n} c_j x_j \\ \text{制約} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i \\ x_j \in \{0,1\}, \\ c_j > 0, \quad a_{ij} \ge 0, \quad b_i > 0 \\ i = 1,2\ldots,m \qquad j = 1,2\ldots,n \end{cases}$$

3 近似解法

本研究では近似解法として有効勾配法[1]とＲＳＢ法[2]を取り上げる．

3.1 有効勾配法

全プロジェクトを実施した時の資源の要求量から有効勾配の値を基準にして，プロジェクトを除外していき，資源の利用可能量を超えない範囲でできるだけ利益の大きくなるプロジェクト集合を求める方法である．以下に使用する記号を表記する．

$N=\{1,2,...n\}$:　プロジェクトjをすべて実行した場合の集合

$N_1: x_j=1$となるプロジェクトjの集合

(第jプロジェクトの）資源ベクトル：$\boldsymbol{a}_j=(a_{1j},a_{2j},\cdots,a_{mj})^T$

利用可能量ベクトル：$\boldsymbol{b}=(b_1,\cdots,b_m)^T$

要求量ベクトル：$\boldsymbol{p}=(\sum_{j\in N_1}a_{1j},\cdots,\sum_{j\in N_1}a_{mj})^T$

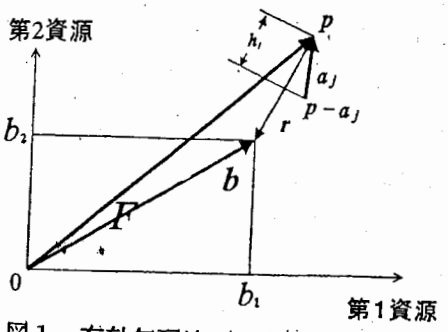

図1．有効勾配法（2次元の場合）

これらの記号を用いてアルゴリズムを記述すると以下のようになる．

step1.　$N_1:=N$,　　$\boldsymbol{p}:=\sum_{j\in N_1}\boldsymbol{a}_j$とし，$List:=\phi$とする．

step2.　$\boldsymbol{p}\le\boldsymbol{b}$となるまで，以下を繰り返す．

a) ベクトルrの各成分r_iを$r_i:=\max\{0,\ p_i-b_i\}$で定める．

b) すべての$j\in N_1$につき，$h_j:=(\boldsymbol{a}_j,\boldsymbol{r})/\|\boldsymbol{r}\|$, $g_j:=c_j/h_j$とする．

c) $g_k=\min_{j\in N_1}g_j$となる$k\in N_1$を求め$N_1:=N_1\setminus\{k\}$, $\boldsymbol{p}:=\boldsymbol{p}-\boldsymbol{a}_k$とし，

$List$の最後尾にkを加える．

step3.　$List$を後ろから順に見て，その要素jに対して

$\boldsymbol{p}+\boldsymbol{a}_j\le\boldsymbol{b}$であれば，$\boldsymbol{p}:=\boldsymbol{p}+\boldsymbol{a}_j$, $N_1:=N_1\cup\{j\}$

とする操作を繰り返す．

有効勾配法は，0-1 ナップサック問題に対しての近似解法である貪欲解法を多次元ナップサック問題に拡張したものである．貪欲解法は全てのプロジェクトを実施しないとして，c_j/a_jの値の大きいプロジェクトから資源制約を超えない範囲で実施していくが，有効勾配法は，全てのプロジェクトを実施するとして，r方向への単位寄与量あたりの利益の減少量の小さいプロジェクトから除外していく．

3.2 RSB 法

良い近似解は最適解と共通する部分を含む場合が多い．そこで，多数の局所最適解を生成して，その目的関数値の良いもの何個かの共通部分を最適解の構成要素として固定し，問題のサイズを縮小する．これを問題のサイズが十分小さくなるまで繰り返す方法である．以下に RSB 法の枠組みを示す．

step1.問題のサイズを$s:=n$とする．

step2.ランダムに選んだ初期解を逐次改善して局所最適解を求めることを，L回行う．

step3.L個の局所最適解から目的関数値の良いものをk個選び出す．

step4.目的関数値の良いほうから順に，解の一致部分を抽出し，一致部分がなくなる（直前）までの一致部分を固定し，問題のサイズを縮小する（sを更新する）．

step5.問題のサイズが十分小さくなったら step6 に進む．そうでなければ，縮小された問題を新たな問題として，step2 にもどる．

step6.今まで得られた最良の解を出力する．

3.2.1 RSB 法のナップサック問題への適用

前節の RSB 法の枠組みをナップサック問題に適用する．本研究では step2 で局所探索を行うときの近傍のとり方について以下の２つを試みた．

①変数x_jに０か１を生成し，それをそのまま用いる方法．

②変数x_jの添え字j（１からn）の順列を用いる方法．

それぞれの方法について初期解の構成方法および近傍のとり方について述べる．

初期解：①の場合は実行可能解をランダムに選ぶ．

②の場合は順列$\sigma \in S_n$をランダムに選び，$x(\sigma)$を初期解とする．

近傍：①の場合は，現在の解ベクトルxとハミング距離２以下にある解x'を近傍探索の対象とする．すなわちxの近傍$U(x)$を次のように決める．

$$U(x)=\{x' \mid \|x'-x\|_H \le 2, x \in \{0,1\}^n\} \quad \text{ただし} \|x'-x\|_H = \sum_{i=1}^{n} |x_i'-x_i|$$

②の場合は順列$\sigma \in S_n$に従い，プロジェクトの採否を決める．容量内であれば採用し，入らなければ不採用とする．σによって一意に決まる解を$x(\sigma)$と書く．現在解$x(\sigma)$の近傍$V(x(\sigma))$を，σの２要素を交換した順列ρから生成されるベクトル$x(\rho)$全体の集合とする．

$V(x(\sigma))=\{x'(\rho) \mid \rho=\sigma \cdot \tau$　：τは互換$\}$

RSB の枠組みの step2 に上記を組み込んだ解法をそれぞれ RSB 法①，RSB 法②とする．

Step3 以降では良い目的関数値を与えるもの上位k個選び，k個の解で０または１で全て同じになる変数を抽出し，固定して問題のサイズを縮小する．RSB 法②の場合も，変数の固定はxについて行う．

4 実験

4.1 テストデータの作成

テストデータは Piginger[3]に基づいて作成した．本研究では Uncorrelated instances(unc.)，Uncorrelated instances with similar weights(unc.sim.w.)，Inverse weakly correlated instances(inv.w.cor.)の３つのテストデータを作成した．なお，bはa_jの和に対しての割合（%）で与えた．変数の数n=50，100，制約式の数m=2，5 としてそれぞれに対して５つのデータを生成した．inv.w.cor でa_jの下限が負となってしまう場合は１とした．

表１．テストデータ

	b(%)	a_j(下限)	a_j(上限)	c_j(下限)	c_j(上限)
unc	50	1	1000	1	1000
unc.sim.w.	50	1000	1100	1	1000
inv.weakly cor.	50	c − 100(1)	c + 100	1	1000

4.2 実験の概要

4.1 節のそれぞれのテストデータに対して有効勾配法，3.2.1 節の RSB 法①，RSB 法②により得られる解の精度を比較する．それぞれのデータに対して LP_SOLVE により厳密解を求め，先の３つの近似解法による解の厳密解に対する割合（それぞれの解が与える目的関数値の割合)を求め，解の精度を評価する．プログラム言語には Delphi3.1 を用いた．

4.3 実験結果

計算機による実験結果を以下の表に示す．なおテストデータ inv.w.cor.(5,50)1-5 およびn=100 のす

べてのデータに対してはLP_SOLVEによる解が得られなかったため，厳密解の替わりにRSB法②による解を用いた(表中の＊)．

表2．実験結果

	有効勾配法		RSB①		RSB②		LP_SOLVE
ファイル名	/厳密	計算時間	/厳密	計算時間	/厳密	計算時間	計算時間
unc.(2,50).1-5	0.9873	12	0.920135	1418	1	4445	○
unc.(5,50).1-5	0.987896	13	0.915138	1998	1	5772	○
unc.(2,100).1-5	0.992946	21	0.863802	5946	1＊	69436	—
unc.(5,100).1-5	0.993254	23	0.870123	5792	1＊	86351	—
unc.sim.w.(2,50).1-5	0.990363	11	0.93175	1142	1	5168	○
unc.sim.w.(5,50).1-5	0.989684	11	0.890353	3862	1	6428	○
unc.sim.w.(2,100).1-5	0.993342	37	0.900073	4864	1＊	78338	—
unc.sim.w.(5,100).1-5	0.994758	41	0.872453	8297	1＊	102001	—
inv.w.cor.(2,50).1-5	0.991512	11	0.977016	2167	1	3902	約1分
inv.w.cor.(5,50).1-5	0.984675	12	0.980024	2928	1＊	4556	—
inv.w.cor.(2,100).1-5	0.99046	41	0.944511	7102	1＊	59177	—
inv.w.cor.(5,100).1-5	0.990567	38	0.97199	17687	1＊	71339	—

—：実行不能（１０分以上解を出力せず/LP_SOLVEがgive up）

○：リターンキーを押して直ちに解を出力

時間の単位はミリ秒（pentiumⅢ 450MHz，Win/98上で実行）

5．考察

有効勾配法により得られる解の精度はどのテストデータに対してもかなり良かった．計算時間も非常に小さく，既存の近似解法としての定評どおりの結果となった．またRSB法①は有効勾配法，RSB法②と比べ，実行時間は比較的小さいが，解の精度はあまり良くなかった．これは，近傍 $U(\boldsymbol{x})$の中を局所探索するとき，実行可能解しか考慮していないので，“良い”局所最適解を発見できる機会が少ないためであろう．RSB法②は実行時間はかなりかかるが，LP_SOLVEで厳密解が得られたすべてのテストデータに対して厳密解を見出している．これは，近傍$V(\boldsymbol{x}(\sigma))$においては，$\boldsymbol{x}$の実行可能性を気にせず探索を行えるので“良い”局所最適解が比較的良く発見されるからであろう．テストデータ inv.w.cor は$\boldsymbol{a}_j$と$\boldsymbol{c}_j$に相関があるデータであるが，LP_SOLVEでは解きにくいデータであり，逆にRSB法②では比較的実行時間が小さくなることが観察された．LP_SOLVEの実行時間（実行可能性）が，n=50とn=100で断絶的に違うのは，組合せ最適化問題に対する厳密解法の通例である．

6．おわりに

本研究では多次元ナップサック問題に対して有効勾配法，RSB法①，RSB法②を適用し，それらの性能を比較した．RSB法②による解は非常に精度の良いものであったが，実行時間はかなり大きい．RSB法②の実行時間の改善は今後の課題である．RSB法①は，実行可能でない解を含めて局所探索を行うことにより精度の改善が期待できるが，これも今後の課題である．

[参考文献]

[1]今野　浩，鈴木　久敏：「整数計画法と組み合わせ最適化」，日科技連　，１９８２．

[2]沼田　一道，児玉　淳：「解の淘汰により変数の固定を行う探索法について」，システム制御情報学会論文誌，vol，１２，pp５７－５９　，１９９９．

[3]Martello,S.,Pisinger,D and Toth,P.,: New Trends in Exact Algorithms for the 0-1 knapsack Problem,Technical report,97/10,DIKU,University of Copenhagen,Denmark,1997.