1998年度日本オペレーションズ・リサーチ学会
春季研究発表会

集合被覆問題に対する局所探索について

京都大学　*岸田 正博　KISHIDA Masahiro
01704164　京都大学　柳浦 睦憲　YAGIURA Mutsunori
01001374　京都大学　茨木 俊秀　IBARAKI Toshihide

1　はじめに

集合被覆問題は代表的な組合せ最適化問題の一つであり, NP 困難であることが知られている. これは, 与えられた要素集合を全てカバーする集合族の中で重み最小のものを求めるという問題である. この問題に対しては, 様々な近似解法が提案されており, 局所探索法を用いたアルゴリズムもいくつか存在する [1, 2]. しかし, これまでの局所探索法は, 解となる集合族に対して 1 つの集合を出し入れするという最も小さな近傍に基づくものだけであった. そこで, 本稿では, 同時に 3 つまでの集合を出し入れするという, より大きな近傍に基づく局所探索を提案する. ただし, 単純に近傍を広げただけでは効率が悪くなるので, 大きな近傍を扱う際には, 解の質を悪くすることなく効率的に近傍を探索する工夫を行っている. 代表的なベンチマーク問題に対する計算実験より, 同程度の計算時間を与えた場合, 大きな近傍を利用することによって, 小さな近傍のみを用いた方法よりも良い解が得られるという結果が得られた.

2　集合被覆問題

集合被覆問題とは, 要素集合 $M=\{1,...,m\}$ と, Mの部分集合族 $S_j, j \in N=\{1,...,n\}$ が与えられたとき, Mの全ての要素をカバーするようにいくつかの集合を選び, 選んだ集合に付けられた重みの総和を最小にする問題である. 0-1 変数 $x \in \{0,1\}^n$を用いると, この問題は以下のように定式化される.

$$\begin{array}{lll} \text{minimize} & \sum_{j=1}^{n} c_j x_j & (1) \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M & (2) \\ & x_j \in \{0,1\}, \quad j \in N & (3) \end{array}$$

a_{ij}: 集合 S_jが要素 i をカバーするなら 1, そうでなければ 0.

c_j: 集合 S_jの重み.

x_j: 集合 S_jが解に含まれるなら 1, そうでなければ 0.

3　局所探索法

局所探索法とは, 現在の解 x の近傍 $NB(x)$ 内に x より良い解があればそれに置き換える, という操作を反復するものである.

LOCAL SEARCH

Step 0: 適当な近似解法で初期解を求め, x とする. $k=1$ とする.

Step k: x の近傍 $NB(x)$ 内で x より良い目的関数値をもつ実行可能解 x'を探す. そのような x'が見つかれば, 解を x:=x'と更新したのち $k := k+1$ として Step kへ. そうでなければ現在の暫定解 x を出力して停止.

通常, 局所探索を 1 度行っただけでは, 未探索の領域にさらに良い解が隠れているという危惧が残るため, 本稿では, ランダム多スタート局所探索法とその変形である GRASP 法を用いる. これは, ランダムに, または簡単な近似解法によって生成された多数の初期解それぞれに局所探索法を適用し, 得られた最良の解を出力するというものである.

3.1　初期解の生成

初期解の生成には 2 種類の方法を用いる.

1 つは GRASP 法 [3] で用いられる方法であり, 現在の解 x に対し

$U=\{i \in M \mid \sum_{j=1}^{n} a_{ij} x_j = 0\}$
$u_j(x)=\sum_{i \in U} a_{ij}$
$r_j=c_j/u_j(x)$
$r_{min}=\min\{r_j \mid j \in N\}$
$R=\{j \in N \mid r_j \leq \alpha r_{min},\ \alpha \geq 1\}$

とするとき, Rよりランダムに 1 つの集合 S_jを選び解に加えるという操作を全ての要素がカバーされるまで繰り返すという方法である. ただし, αは 1 以上のパラメータである.

もう 1 つは, Uより要素 i をランダムに選び, それをカバーする集合 S_jの内で r_jが最小のものを解に加えるという操作を全ての要素がカバーされるまで繰り返すという方法である. 以後これを R-Gr 法と表す.

3.2 近傍

$\boldsymbol{x}$ と $\boldsymbol{x}'$のハミング距離を $d(\boldsymbol{x},\boldsymbol{x}')=|\{j \in N|\ x_j \neq x'_j\}|$ で表す. $\boldsymbol{x}$ の近傍 $NB_h(\boldsymbol{x})$ は以下のように定義される.

$NB_h(\boldsymbol{x})=\{\boldsymbol{x}' \in \{0,1\}^n|\ d(\boldsymbol{x},\boldsymbol{x}') \leq h\}$

すなわち, $\boldsymbol{x}$ からのハミング距離が h 以下の解集合である. 本稿では, $h \leq 3$ の近傍に基づく局所探索を行うが, $NB_h(\boldsymbol{x})$ の中で改善解を逃すことなく効率的に探索するために, 以下のような工夫を行っている. まず, 局所探索法において探索する解 $\boldsymbol{x}$ を実行可能解に限定する. $h(\leq 3)$ に対し, $NB_h(\boldsymbol{x})$ 内に改善解を見つけてそれに移動するか, または改善解がないことを結論するまでに必要な時間を考察する. 説明の都合上, 次の記号を定義する.

$n'=\sum_{j=1}^{n} x_j$

$t=\max\{\sum_{i=1}^{m} a_{ij}|\ j \in N\}$

$l=\max\{\sum_{j=1}^{n} a_{ij}|\ i \in M\}$

$d(\boldsymbol{x},\boldsymbol{x}')=1$ であるような改善解は, 現在の解から1つ集合を取り除いたものである. 各集合に対し, その集合を取り除いても実行可能性を保てるかどうかを $O(1)$ 時間で判定するため, その集合が単独でカバーしている要素数をメモリーに記憶しておく. その結果, 調べるべき集合の候補が $O(n')$ 個, 取り除ける候補が見つかったときのメモリーの更新に $O(tl)$ かかるので, この探索は $O(n'+tl)$ の計算時間で実行できる.

$d(\boldsymbol{x},\boldsymbol{x}')=2$ であるような解は, 現在の解から集合を1つ除き, 新たに1つ加えたものを考えれば十分である (同時に2つの集合を除く候補は考えなくて良い). $d(\boldsymbol{x},\boldsymbol{x}')=1$ で用いたメモリーに加えて, 各集合に対し, $u_j(\boldsymbol{x})$ の値をメモリーに記憶しておくことにより, 取り除く各候補に対して, それを加えることによって実行可能解になるような候補を $O(tl)$ 時間で見つけることができるので, 全体の手間は $O(n'tl)$ 時間である.

$d(\boldsymbol{x},\boldsymbol{x}')=3$ であるような解は, 現在の解から集合を1つ除き, 新たに2つ加えたものと, 2つ除き, 新たに1つ加えたものの2種類ある. まず前者については, 取り除く各候補に対し, まず新たに $u_j \geq 1$ となった集合を列挙し, 次にそれらのペアを調べるという手順で探索を行うことにより, $O(n'tl^2)$ 時間で全体の探索が可能である.

後者については, 取り除く各候補に対し, $d(\boldsymbol{x},\boldsymbol{x}')=2$ のときと同様の方法で, 加えることにより実行可能解が得られるような集合を列挙したのち, それらを加えることによりさらに取り除ける集合が新たにできるかを調べるという手順で, $O(n'tl^2)$ 時間で全体の探索が可能である.

結局, $NB_3(\boldsymbol{x})$ 全体の探索を $O(n'tl^2)$ 時間で行えることが分かった. $NB_3(\boldsymbol{x})$ の探索をこのような工夫なしに行うと, 通常 $O(n'tn^2)$ 時間必要となるが, $l \ll n$ である問題例が多いことから, かなりの高速化が期待できる.

4 計算結果と結論

実験は, ワークステーション Sun Ultra 2 Model 2300 (300 MHz, 1 GB memory) 上で C 言語を用いて行った. 用いた問題例は, Beasely の OR-Library からの代表的なベンチマーク問題で, 全て最適解が知られている.

表1に, GRASP 法と R-Gr 法で求められた初期解に対し, 近傍を NB_1, NB_2, NB_3 としたランダム多スタート局所探索法による結果を示す. 表の値は, 計算時間 60 秒で探索を打ち切ったときに得られている暫定解の最適解からの誤差 (%) の平均値である. GRASP 法で用いるパラメータ α は, α=2 とした.

表1. 異なる近傍に対する実験結果

初期解	問題タイプ (m, n, density)	近傍 NB_1	近傍 NB_2	近傍 NB_3
GRASP	4 (200, 1000, 2%)	2.212	0.240	0.047
	5 (200, 2000, 2%)	2.571	0.537	0.100
	6 (200, 1000, 5%)	1.446	0.290	0.0
	A (300, 3000, 2%)	3.731	1.117	0.336
	B (300, 3000, 5%)	1.519	0.263	0.0
	C (400, 4000, 2%)	5.590	2.567	0.811
	D (400, 4000, 5%)	4.685	1.898	0.274
R-Gr	4 (200, 1000, 2%)	5.035	2.353	0.525
	5 (200, 2000, 2%)	5.477	2.847	0.909
	6 (200, 1000, 5%)	4.386	2.721	0.780
	A (300, 3000, 2%)	4.351	2.326	1.232
	B (300, 3000, 5%)	2.339	1.569	0.250
	C (400, 4000, 5%)	5.201	3.433	1.653
	D (400, 4000, 5%)	5.265	3.735	1.589

表より, 初期解の生成法によらず, 大きな近傍を用いたほうが良質の解が得られるという傾向が観測できる. また, GRASP 法と R-Gr 法では, GRASP 法の方が性能が良いことも分かる. 集合被覆問題に対しては, ラグランジュ乗数を利用した手法が有効であることが知られているので, 今後は, そのような手法を組込むなど, さらに工夫を行う予定である.

参考文献

[1] J.E. Beasley and P.C. Chu, "A Genetic Algorithm for Set Covering Problem," *European Journal of Operational Research* 94 (1996) 392-404.

[2] L.W. Jacobs and M.J. Brusco, "A Local-Search Heuristic for Large Set-Covering Problems," *Naval Research Logististics* 42 (1995) 1129-1140.

[3] T.A. Feo and M.G.C. Resende, "A Probablistic Heuristic for A Computationally Difficult Set Covering Problem," *Operations Research Letters* 8 (1989) 67-71.