数理解析研究所講究録
第1682巻 2010年 176-183

反復割当問題に対する問題縮小アルゴリズム

Reduction and Exact Algorithms for the Repeated Assignment Problem

防衛大学校情報工学科
横谷 大輔, 山田 武夫

YOKOYA Daisuke, YAMADA Takeo
{g47090, yamada}@nda.ac.jp
Department of Computer Science, The National Defense Academy
Yokosuka, Kanagawa 239-8686, Japan

1 はじめに

$n \times n$ の割当問題を K 回反復する問題で，重複した割当を禁止するものを**反復割当問題** (RAP: repeated assignment problem)[12] と呼ぶ．RAP は次のような 0-1 計画問題 [9] として定式化される．

$$\text{RAP}: \quad \text{minimize} \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij}^k \tag{1}$$

$$\text{subject to} \sum_{j=1}^{n} x_{ij}^k = 1, \quad \forall i,k, \tag{2}$$

$$\sum_{i=1}^{n} x_{ij}^k = 1, \quad \forall j,k, \tag{3}$$

$$\sum_{k=1}^{K} x_{ij}^k \le 1, \quad \forall i,j, \tag{4}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall i,j,k. \tag{5}$$

ここで，c_{ij}^k は k 回目の割当てコストであり，(4) は同じ割当の重複を禁止する制約を表している．RAP で $c_{ij}^k \equiv c_{ij}$ の場合は矢島ら [10] による研究があるが，本報告では一般の場合について検討する．RAP は n^2K 変数，$n^2 + 2nK$ 制約式の 0-1 整数計画問題である．$n = 1000$, $K = 10$ の場合は，1 千万変数，102 万制約式となり，かなり大きなサイズの問題で，現在市販されている商用の MIP ソルバーで厳密解を得ることは難しい [6]．この問題に対し，上下界値を利用した釘付けテストにより問題を縮小し，それを解くことによって厳密解を得るアルゴリズムを提案する．

2 上下界値

RAP の上界値と下界値について述べる．まず，上界値は 2.1 節で述べる**反復ハンガリー法**により得られる．次に，下界値については連続緩和とラグランジュ緩和を考えるが，前者は大規模な線形計画問題となることが多いので，これを効率的に解くため，**行，列の遅延取込み法**を提案する．

2.1 反復ハンガリー法

上界値の算出法として 反復ハンガリー法を提案する．これはコスト行列 $C^k = (c_{ij}^k)$ の割当問題を，$k = 1, 2, \cdots, K$ の順に逐次解くものであるが，第 k 回目の割当においては，それ以前までの反復で割当済みの組は割当が禁止される．具体的には次のようになる．$F \subseteq \{(i,j) \mid 1 \le i, j \le n\}$ を割当が禁止されたペアとする．最初これは空集合である．$\mathrm{AP}^k(F)$ を $n \times n$ の割当問題で，割当 (i,j) のコストが

$$\bar{c}_{ij}^k = \begin{cases} c_{ij}^k, & (i,j) \notin F \\ \infty, & (i,j) \in F \end{cases} \tag{6}$$

のものとする．ここに，F は「割当て禁止枝」の集合であり，$k = 1$ のときは $F = \emptyset$ とする．$\mathrm{AP}^k(F)$ に対応する 2 部グラフを $H^k(F)$ とし，ハンガリー法 [1, 5] により得られる**完全マッチング**を $M^k(F)$ とする．$M^k(F)$ の枝は $F := F \cup M^k(F)$ と更新することにより以降の割当てが禁止される．反復ハンガリー法のアルゴリズムは次のようになる．

Algorithm 反復ハンガリー法

Step 1. $k := 0$, $F := \emptyset$ とする．

Step 2. $\mathrm{AP}^k(F)$ をハンガリー法で解き，最適完全マッチング $M^k(F)$ を得る．

Step 3. $k \ge K$ なら終了．そうでなければ $k := k + 1$ and $F := F \cup M^k(F)$ として Step 2 へ．

このアルゴリズムは K 回の反復で終了して RAP の実行可能解が得られる．この方法に関して次の事実が成り立つ．

補題 1 任意の $k \le n$ に対して，$H^k(F)$ の中に完全マッチングが存在する．

このことから次が得られる．

定理 1 反復ハンガリー法は常に RAP の実行可能解を与える．

この実行可能解の目的関数値は RAP の一つの上界値となるので，これを z_{RAP} と記す．

2.2 連続緩和

(5) を $x_{ij}^k \ge 0$ に置き換えた RAP の連続緩和問題を C(RAP) とする．これは $2nK + n^2$ 制約式，n^2K 変数の線形計画（LP）問題であるが，n, k の値の増加とともに急激に大規模な問題となる．そこで，このようなサイズの大きい LP 問題に対処するために，**行，列の遅延取込み法** [13] を提案する．

2.2.1 行，列の遅延取込み法

次の LP 問題を考える．

$$\mathrm{P}: \text{maximize } c^T x \quad \text{subject to } Ax \le b,\ x \ge 0,$$

問題 P のすべての変数とすべての制約式の集合をそれぞれ，$\bar{C}$，$\bar{R}$ とする．P の最適解 x^* において $x_j^* = 0$ のとき行列 A の第 j 列は**ゼロ列**であるといい，第 i 番目の制約式が等号で成立していれば，第 i 行は**活性**であるという．

問題 P を部分集合 $R \subseteq \bar{R}$, $C \subseteq \bar{C}$ により次のようにブロック分けして表記し，R, C に対応する部分問題 P(R, C) を考える．

	C	C'	const
R	A_{00}	A_{01}	b_0
R'	A_{10}	A_{11}	b_1
obj	c_0^T	c_1^T	0

すると，P(R, C) は次のように書ける．

$$\mathrm{P}(R, C): \text{ maximize } c_0^T x \quad \text{subject to } A_{00}x \le b_0,\ x \ge 0.$$

P(R, C) とその双対問題の最適解をそれぞれ $x^*(R, C)$, $y^*(R, C)$ とすると次の定理が成り立つ．

定理 2 $x^*(R, C)$, $y^*(R, C)$ が

(i) $A_{10}x^*(R, C) \le b_1$ (主実行可能性)

(ii) $y^*(R, C)^T A_{01} \ge c_1^T$ (双対実行可能性)

を満たすとする．これらに C', R' に対応する部分に 0 を付加した解 $x^* := (x^*(R, C), 0)$, $y^* := (y^*(R, C), 0)$ は主問題とその双対問題の最適解である．

あらかじめ活性行の集合 R と非ゼロ列の集合 C が正確にわかっていた場合は，縮小された問題 P(R, C) を解くことによって元問題の最適解が得られる．しかしながら，実際には正しい区分けは P を完全に解くまではわからない．そこで，以下に述べる遅延取込み法では，活性である可能性が高い行の集合を R_0, 非ゼロである可能性が高い列の集合を C_0 として選択し，これらが定理 2 の条件 (i), (ii) を満足するように，逐次修正して行く．具体的なアルゴリズムは次のようになる．

Algorithm 行，列の遅延取込み法

Step 1. 適当な R_0, C_0 の組を取り，$(R, C) := (R_0, C_0)$ とする．

Step 2. $P(R, C)$ を解き，$x^* = x^*(R, C)$, $y^* = y^*(R, C)$ を求める．

Step 3. $A_{10}x^* \le b_1$ に反する「違反行」があれば，それらを R に加える．

Step 4. $y^*A_{01} \ge c_1$ に反する「違反列」があれば，それらを C に加える．

Step 5. 違反行, 違反列がなければ x^*, y^* (に適当に 0 成分を補ったもの) が P の最適解であるので，これらを出力して終了．そうでないときは Step 2 へ戻る．

2.2.2 RAP への適用

行，列の遅延取込法を C(RAP) に適用する場合，最初の (R_0, C_0) をどう選択するかが問題となる．R_0 としては常に活性な (2), (3) の制約式を選べばよい．一方，C_0 の選択はそれほど明らかではないが，反復ハンガリー法による実行可能解で $x_{ij}^k = 1$ となった変数は，最適解において非ゼロである可能性が高いと考えられる．そこで，このような変数の集合を C_0 として選択する．C(RAP) を解いて得られる下界値を z_C と以下では記す．

2.3 ラグランジュ緩和

RAP において，(4) 式を目的関数値に取り込んだラグランジュ緩和問題 [3] は次のようになる．

$$\mathrm{LRAP}(\gamma): \quad \text{minimize} \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}^k+\gamma_{ij})x_{ij}^k-\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}$$
$$\text{subject to (2), (3) and (5).}$$

$\gamma=(\gamma_{ij})$ は制約式 (4) に対応するラグランジュ乗数である．$\gamma\geq 0$ を固定すると LRAP(γ) は次の K 個の独立した割当問題に分解される．

$$\mathrm{AP}^k(\gamma): \quad \text{minimize} \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}^k+\gamma_{ij})x_{ij}^k \tag{7}$$
$$\text{subject to} \sum_{j=1}^{n}x_{ij}^k=1, \quad \forall i, \tag{8}$$
$$\sum_{i=1}^{n}x_{ij}^k=1, \quad \forall j, \tag{9}$$
$$x_{ij}^k\in\{0,1\}, \quad \forall i,j. \tag{10}$$

LRAP(γ)，$\mathrm{AP}^k(\gamma)$ の最適目的関数値をそれぞれ $\underline{z}(\gamma)$，$\underline{z}^k(\gamma)$ とする．LRAP(γ) の**ラグランジュ双対問題**を考えると，

$$\text{minimize } \underline{z}(\gamma) \text{ subject to } \gamma\geq 0.$$

で，この問題の最適目的関数値 $\underline{z}_L$ は，ラグランジュ緩和による下界値となる．

ラグランジュ双対問題の解は連続緩和解から以下のように求められる．$(u^\dagger,v^\dagger,\gamma^\dagger)\in R^{nK}\times R^{nK}\times R_+^{n^2}$ を C(RAP) の双対問題の最適解とする．ここで，$u^\dagger$, $v^\dagger$, $\gamma^\dagger$ はそれぞれ (2), (3), (4) に対応する双対変数である．すると次のことがわかる．

定理 3 $\gamma^\dagger$ はラグランジュ双対問題の最適解となる．すなわち，$\underline{z}_L=\underline{z}(\gamma^\dagger)$．さらに，ラグランジュ緩和による下界値と連続緩和による下界値は一致する．つまり，$\underline{z}_L=\underline{z}_C$.

以下ではこれらの下界値を $\underline{z}_{RAP}$，$\mathrm{AP}^k(\gamma^\dagger)$ の最適目的関数値を $\underline{z}^k$，$\mathrm{AP}^k(\gamma^\dagger)$ を AP^k と記す．すると，RAP の下界値は次のように書ける．

$$\underline{z}_{RAP}=\sum_{k=1}^{K}\underline{z}^k-\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}^\dagger. \tag{11}$$

3 釘付けテスト

$\delta\in\{0,1\}$, $gap:=\overline{z}_{RAP}-\underline{z}_{RAP}$ とし，$x_{ij}^k=\delta$ と固定した問題を $\mathrm{RAP}(x_{ij}^k=\delta)$，この問題の下界値を $\underline{z}(x_{ij}^k=\delta)$ で表す．もし $\underline{z}(x_{ij}^k=\delta)\geq\overline{z}_{RAP}$ であれば，$x_{ij}^k=\delta$ と固定しても，$\overline{z}_{RAP}$ より良い解が得られないことになるので，$x_{ij}^k=\delta'$ ($\delta':=1-\delta$) と結論できる．そこで，$\underline{z}(x_{ij}^k=\delta)$ の評価をするために AP^k において $x_{ij}^k=\delta$ と固定した割当問題 $\mathrm{AP}^k(x_{ij}^k=\delta)$ を導入し，この問題の最適目的関数値を $\underline{z}^k(x_{ij}^k=\delta)$ とする．(11) より

$$\underline{z}(x_{ij}^k=\delta)=\underline{z}_{RAP}+\underline{z}^k(x_{ij}^k=\delta)-\underline{z}^k(\gamma).$$

なので，

$$\underline{z}^k(x_{ij}^k=\delta)-\underline{z}^k\geq gap \tag{12}$$

であれば x_{ij}^k を δ' に固定できる．

0-1 計画問題に対する釘付けテスト [7, 14] には最適単体表 [2, 8] を利用する．本報告では，ラグランジュ緩和によって分解された割当問題をハンガリー法を用いて解くが，一般にハンガリー法による解からは最適単体表は得られない．そこで，ハンガリー法による解から最適単体表を復元する必要があるが，紙数の都合上，詳しくは省略する．

4 仮想釘付けテスト

釘付けテストの効果は上下界値の差 (gap) に依存する．上下界値の差が大きすぎると，釘付けによる問題縮小効果は小さくなるが，予備的な実験では反復ハンガリー法による上界値はそれほど良い値とは言えず，上下界値の差が大きい．そのため，期待していた問題縮小効果が得られないことが多かった．そこでより高い問題縮小効果を得るために，仮想釘付けテストを提案する．

4.1 原理

釘付けテストは上下界値を用いて行うが，仮想釘付けテストでは，仮の上界値 $l \in [\underline{z}, \overline{z}]$ （**試行値**と呼ぶ）を用いて釘付けテストを行う．実行可能解の集合を $X = \{x \in R^n | Ax = b, x \geq 0\}$ とする．$\underline{z}$ と l を用いて釘付けテストを行うと，いくつかの変数 x_j が 0 または 1 に固定される．しかし，l は実際に上界値になっているかはわからないので，この釘付けが正しく行われている保証はない．この仮想釘付けテストで 0，1 に固定された変数の集合をそれぞれ $F_0(l)$，$F_1(l)$ とすると，次の問題が得られる．

縮小問題. $\quad$ Q(l): minimize $c^T x$ subject to $Ax = b, x \geq 0$, *and*

$$x_j = 1, \text{if } j \in F_1(l),\ x_j = 0, \text{if } j \in F_0(l).$$

この縮小問題の最適目的関数値を $z_l^\star$ と表し，試行値 l に対する**実現値**と呼ぶ．Q(l) が実行不可能の場合は $z_l^\star := \infty$ とすると，次の定理が成り立つ．

定理 4 [14] 任意の試行値 $l \geq \underline{z}$ とその実現値 $z_l^\star$ に対して，

(i) $l \geq z^\star \Rightarrow z_l^\star = z^\star$,

(ii) $l < z^\star \Rightarrow z_l^\star \geq z^\star$,

(iii) $l \geq z_l^\star \Rightarrow z_l^\star = z^\star$.

すなわち，試行値 l に対して実現値 $z_l^\star$ がそれ以下であるならば，$z_l^\star$ は最適値 z^* と一致して RAP が正確に解かれたことになる．

4.2 アルゴリズム

仮想釘付けテストを RAP に適用するにあたり，$\alpha < gap := \overline{z}_{RAP} - \underline{z}_{RAP}$ となる仮想ギャップ $\alpha > 0$ を用意する．仮想釘付けテストのアルゴリズムは次のようになる．

Procedure 仮想釘付けテスト.

Input: $\bar{z}_{RAP}$, $\underline{z}_{RAP}$, AP^k $(k = 1, \ldots, K)$ の最適単体表.

Parameter: 仮想ギャップ α.

Step 1. 各 AP^k に試行値 $\underline{z}_{RAP} + \alpha$ を用いて釘付けテストを行う.

Step 2. RAP から固定された変数を取り除き，縮小された 0-1 計画問題を得る.

Step 3. MIP ソルバーで，縮小された 0-1 計画問題を解き，RAP の実行可能解 $\tilde{x}$ と対応する目的関数値 $\tilde{z}$ を得る．これらを出力して終了.

$\tilde{z}$ は試行値 $\underline{z}_{RAP} + \alpha$ に対する実現値になっている．定理 4 より，$\tilde{z} \leq \underline{z}_{RAP} + \alpha$ であれば，得られた解は最適解である．$\tilde{z} > \underline{z}_{RAP} + \alpha$ となって最適解の保証が得られなかった場合は α を増加させて，もう一度仮想釘付けテストを行う．最適解の保証が得られるまでこれを繰り返すが，実際には適当なところで（すなわち，近似解を得て）計算を打ち切ることもある.

5 数値実験

前節までのアルゴリズムを ANSI C 言語で実装し，Dell DIMENSION 8400 (CPU : Pentium(R)4 CPU 3.40GHz, メモリー : 2.00GB) 上で MIP ソルバー ILOG CPLEX 11.1 [4] を用いて実験を行った．例題は次のように作成した．基準コスト c^0_{ij} を [1,1000] の一様乱数で生成し，k 回目のコスト c^k_{ij} は，[Floor, Ceil] の一様乱数で生成する．ここで,

$$\mathrm{Floor} := \max\{C^0_{ij} - 1000(1-\sigma), 1\},\ \mathrm{Ceil} := \min\{c^0_{ij} + 1000(1-\sigma), 1000\}.$$

で，σ は相関の度合いを表すパラメータである．以下では $\sigma = 0.0$ の無相関，$\sigma = 0.3$ の弱相関，$\sigma = 0.6$ の強相関のケースについて実験を行った.

5.1 実験結果

表 1 は上下界値を求めた計算結果であり，各行は 10 例題の平均を表す．#col, #row は遅延取込み法における，列数，行数の最大値であり，#cycl は Step 2 - 5 の反復回数を表す．CPU_1 は上界値 $\bar{z}$ と下界値 $\underline{z}$，そして $gap := \bar{z} - \underline{z}$ の計算にかかった時間の秒表示である.

表 2 は表 1 と同じ例題に対し $\alpha = 5$ で仮想釘付けテストを行い，縮小問題を CPLEX で解いた結果である．0，1 に固定された変数の数を#fix0，#fix1, 縮小された問題の列数，行数を#col，#row で表す．$z^\star$ は最適目的関数値であり，CPU は総計算時間 (CPU_1 + 縮小された問題を解いた時間) を秒表示している．#int は遅延取込み法（連続緩和問題）において 0-1 解が得られた回数（10 回中）で，この場合は #fix1 = nK, #fix0 = $n^2K - nK$，#col=#row=0 として平均を出している．また，#solved は最適性の保証 ($\alpha > \tilde{z} - \underline{z}_{RAP}$) が得られた回数（10 回中）である.

これらの表から次のことが言える.

(i) $K = 4$ のときはほとんどのケースで遅延取込み方による C(RAP) の最適解が RAP の最適解となっている.

表 1: 上下界値

σ	n	K	$\bar{z}$	#col	#row	#cycl	$\underline{z}$	gap	CPU_1
0.0	200	4	7054.4	1604.2	2471.1	4.3	7047.1	7.3	0.27
		8	14199.6	3234.4	5001.7	5.6	14149.1	50.4	0.79
		12	21504.9	4891.6	7697.3	7.3	21354.0	150.8	1.67
	400	4	7430.6	3202.1	4882.4	4.3	7427.8	2.8	1.27
		8	14859.6	6428.5	9903.3	5.8	14826.4	33.1	3.58
		12	22282.7	9677.2	15013.8	6.3	22194.3	88.4	5.93
	600	4	7772.8	4801.9	7350.2	4.4	7769.5	3.3	3.00
		8	15589.4	9624.3	14806.8	5.8	15568.4	21.0	7.49
		12	23455.7	14472.1	22408.4	7.0	23397.9	57.7	14.13
0.3	200	4	9042.2	1606	2481.2	4.2	9026.6	15.6	0.29
		8	18187.2	3251.3	5108.1	6.3	18062.2	125.0	0.87
		12	27709.4	4944.3	7874.8	8.0	27355.7	353.6	2.21
	400	4	9477.5	3206.5	4951	4.6	9467.6	9.9	1.32
		8	18995.3	6446.8	10034.2	6.1	18926.1	69.1	3.42
		12	28593.8	9730.3	15197.3	7.5	28402.1	191.6	7.33
	600	4	9815.4	4806	7323.1	4.9	9809.5	5.9	3.04
		8	19674.4	9642.4	14976.3	6.2	19629.7	44.7	7.83
		12	29699.1	14523.2	22654.5	7.5	29591.3	107.7	18.38
0.6	200	4	9842.4	1618.4	2521.2	5.0	9807.0	35.4	0.32
		8	20135.7	3322.6	5292.1	8.1	19827.2	308.4	1.52
		12	31137.1	5158.7	8209.3	8.8	30284.4	852.6	8.05
	400	4	10216.9	3216.4	5025.3	5.2	10201.4	15.5	1.45
		8	20772.8	6509.6	10304.1	7.0	20613.2	159.5	5.30
		12	31402.7	9899.5	15747.1	8.2	31003.3	399.3	18.61
	600	4	10599.6	4814	7404.1	5.3	10585.6	14.0	3.13
		8	21326.1	9700	15136.6	7.5	21234.0	92.0	11.14
		12	32247.8	14672.4	23079.2	9.0	32006.2	241.5	29.84

表 2: 問題縮小効果と厳密解法の結果

σ	n	K	#int	#fix0	#fix1	#col	#row	z^*	CPU	solved
0.0	200	4	10	159200.0	800.0	0.0	0.0	7047.1	0.2	10
		8	7	317792.3	1246.9	960.8	1629.3	14149.4	1.1	10
		12	5	476051.1	1507.4	2441.5	4087.8	21354.5	2.5	10
	400	4	10	638400.0	1600.0	0.0	0.0	7427.8	1.2	10
		8	7	1274708.0	2368.4	2923.7	4512.2	14826.5	4.5	10
		12	9	1914132.0	4377.3	1491.2	2272.5	22194.3	6.5	10
	600	4	10	1437600.0	2400.0	0.0	0.0	7769.5	3.0	10
		8	9	2873661.0	4347.5	1991.6	2849.6	15568.4	8.2	10
		12	8	4308131.0	5944.0	6024.9	8511.9	23398.0	18.2	10
0.3	200	4	9	159118.8	749.8	134.4	238.0	9026.9	0.3	10
		8	7	317902.4	1275.9	821.7	1430.9	18062.5	0.9	10
		12	5	476232.3	1547.8	2219.9	3756.4	27356.2	4.1	10
	400	4	10	638400.0	1600.0	0.0	0.0	9467.6	1.3	10
		8	9	1276229.0	2934.8	836.7	1340.6	18926.2	3.6	10
		12	6	1911771.0	3220.2	5009	7913.3	28402.4	12.3	10
	600	4	10	1437600.0	2400.0	0.0	0.0	9809.5	3.0	1
		8	9	2874012.0	4371.3	1616.3	2429.5	19629.7	8.7	10
		12	4	4301969.0	3315.8	14715.3	21826.1	29591.5	30.3	10
0.6	200	4	9	159124.6	749.4	126	221.9	9807.0	0.3	10
		8	2	316977.4	690.2	2332.4	3958.1	19829.1	15.0	10
		12	0	474601.2	592.9	4805.9	7772.2	30303.4	726.8	0
	400	4	9	638136.7	1471.6	391.7	637.9	10201.4	1.4	10
		8	2	1272516.0	1130.4	6353.2	10136.9	20613.5	7.9	10
		12	0	1906622.0	809.9	12568.5	19437.9	31005.9	300.8	9
	600	4	10	1437600.0	2400.0	0.0	0.0	10585.6	3.1	10
		8	4	2868446.0	2251.0	9302.8	13956.6	21234.3	17.4	10
		12	2	4299009.0	2069.6	18921.3	27765.2	32007.1	139.3	10

(ii) 問題サイズ $(n,\ K)$ と相関 σ が大きくなるにつれて上界値と下界値の差も大きくなる．実際の最適目的関数値 z^* は下界値 z_{LRAP} に非常に近い値になっていることが多いため，仮想釘付けテストはこれらの例題に対して効果的なものになっている．

(iii) 仮想ギャップ $\alpha = 5$ での仮想釘付けテストにより，元問題は CPLEX で解ける程度のサイズにまでに縮小されている．いくつかの例題を除いて，1 分以内に最適解を得ている．最適解の保証が得られていない場合でも，反復ハンガリー法による実行可能解よりもはるかに良い近似解を得ている．

6 結論

本報告では反復割当て問題を定式化し，厳密解を求めるアルゴリズムを提示した．具体的には，上界値を求める方法としては反復ハンガリー法を，下界値を得る方法として遅延取込み法を，そして問題サイズを縮小させるための方法として，仮想釘付けテストを提案した．数値実験の結果，この方法により高い問題縮小効果が得られ，ほとんどの例題を市販の MIP ソルバーによってわずかな時間で解くことができた．その結果，$n = 600$，$K = 12$ 程度のサイズの問題に対しても最適性のある解を得た．

参考文献

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin: *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Englewood Criffs, 1993).

[2] V. Chvátal: *Linear Programming* (Freeman and Company, San Francisco, 1983).

[3] M. Fisher: The Lagrangian relaxation method for solving integer programming problems. *Management Science*, **50** (2004), 1861-1871.

[4] ILOG CPLEX 11.1, http://ilog.com/products/cplex, 2008.

[5] H.W. Kuhn: The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* **2** (1955), 83-97.

[6] 松井, 宮代: ここまで解ける整数計画． システム/ 情報/ 制御 50, pp. 363-368, 2006

[7] R.M. Nauss: *Parametric Integer Programming* (Univ. Missouri Press, Columbia, MI, 1979).

[8] M. Padberg, *Linear Optimization and Extensions, 2nd Ed.*, Springer, 1999.

[9] L.A. Wolsey: *Integer Programming* (John Wiley & Sons, New York, 1998).

[10] 矢島, 小坂: "繰返し授業が行われる場合のクラス編成問題", オペレーションズ・リサーチ, **40**(1995), 421-424.

[11] T. Yamada and Y. Nasu, "Heuristic and exact algorithms for the simultaneous assignment problem," *European Journal of Operational Research*, Vol. 123, pp. 531-542, 2000.

[12] 山下, 山田: 反復割当問題の解法について． 日本 OR 学会 2007 年春季研究発表会，2-E-3, pp. 188-189.

[13] 山下, 山田: 行, 列の遅延取込みによる大規模線形計画問題の一解法． 日本 OR 学会 2007 年秋季研究発表会，2-E-1, pp. 206-207.

[14] B.-J. You and T. Yamada, "A virtual pegging approach to the precedence constrained knapsack problem", *European Journal Operational Research*, Vol. 183, pp. 618-632. 2007.