The Ninth International Symposium on Operations Research and Its Applications (ISORA'10)
Chengdu-Jiuzhaigou, China, August 19–23, 2010

Approximation Algorithms for the Multiply Constrained Assignment Problem

Daisuke Yokoya Takeo Yamada

Department of Computer Science, The National Defense Academy
Yokosuka, Kanagawa 239-8686, Japan

Abstract We consider the multiply constrained assignment problem (MCAP), which is a variation of the linear assignment problem with m additional constraints. First, we derive upper and lower bounds respectively by heuristic and relaxation methods. The latter also implies the Lagrangian relaxation, as well. Next, we introduce the pegging test to MCAP, and further a virtual pegging test to reduce MCAP into a binary integer problem of smaller size. By solving the reduced problem, we obtain an approximate solution, which is often proved optimal. The results of numerical experiments are also given.

Keywords Assignment problem, Side constraints, Pegging test, MIP solver

1 Introduction

Assignment problem (AP for short) [1] is a classical combinatorial optimization problem that can be solved efficiently by polynomial time algorithms [8]. AP has been applied in many practical situations, such as personnel assignment, parallel machine scheduling and matching of moving objects among others. However, many real-life AP models are complicated due to existence of side constraints. Thus, we are concerned with the following *multiply constrained assignment problem* (MCAP).

$$\textbf{MCAP}: \quad \text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

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

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

$$\sum_{i}^{n}\sum_{j}^{n} r_{ij}^{k}x_{ij} \le b_k, \quad k \in M \tag{4}$$

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

where $N = \{1,2,\cdots,n\}$ and $M = \{1,2,\cdots,m\}$.

The case of $m = 1$ is the *singly constrained assignment problem* (SCAP), which can be solved by a branch-and-bound algorithm of Lieshout *et al.* [9]. The purpose of this paper is to present algorithms to obtain approximate solutions for problems with $m \geq 2$.

2 Lower and upper bounds

2.1 Lower bound: continuous relaxation

Let C(MCAP) denote the *continuous relaxation* of MCAP where (5) is replaced with $x_{ij} \geq 0$. This is an LP problem with $2n+m$ constraints and n^2 variables. For $n = 1000$ and $m = 100$, this is an LP with 2100 rows and 1 million columns. However, with the power of computers and solvers [6] of today, we can solve problems of this size without much difficulty. The optimal objective value $\underline{z}_C$ gives a lower bound to MCAP, and by $(u^\dagger, v^\dagger, \lambda^\dagger)$ we denote the optimal dual variables corresponding to (2), (3) and (4) respectively.

2.2 Lower bound:Lagrangian relaxation

With nonnegative multipliers $\lambda = (\lambda_{ij})$ associated with (4), the *Lagrangian relaxation* [4] of MCAP is defined as

$$\begin{aligned} \text{LMCAP}(\lambda): \quad \text{Minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} + \sum_{k=1}^{m}\lambda_k r_{ij}^k)x_{ij} - \sum_{k=1}^{m}\lambda_k b_k \\ \text{subject to} \quad & \text{(2), (3) and (5).} \end{aligned} \tag{6}$$

For a fixed $\lambda \geq 0$, this is an AP. Let the optimal objective values to LMCAP(λ) be $\underline{z}(\lambda)$. Then, we have the following [16].

Theorem 1.

(i) *For an arbitrary $\lambda \geq 0$, $\underline{z}(\lambda)$ gives a lower bound to MCAP, i.e., the optimal objective value $z^\star$ to MCAP satisfies*

$$z^\star \geq \underline{z}(\lambda).$$

(ii) *$\underline{z}(\lambda)$ is a piecewise-linear, concave function of λ.*

(iii) *If $\underline{z}(\lambda)$ is differentiable at λ,*

$$\partial \underline{z}(\lambda)/\partial\lambda_k = \sum_i \sum_j r_{ij}^k x_{ij} - b_k, \; k = 1, 2, \cdots, m. \tag{7}$$

Consider the *Lagrangian dual*:

$$\text{Minimize} \;\; \underline{z}(\lambda) \;\; \text{subject to} \;\; \lambda \geq 0. \tag{8}$$

The optimal objective value to this problem, denoted as $\underline{z}_L$, gives the lower bound by the Lagrangian relaxation. Then, we have the following [16].

Theorem 2. *$\lambda^\dagger$ gives an optimal solution to the Lagrangian dual, That is, $\underline{z}_L = \underline{z}(\lambda^\dagger)$. Moreover, this coincides with the lower bound by continuous relaxation, i.e., $\underline{z}_L = \underline{z}_C$.*

Thus, in what follows we write $\underline{z}$ instead of $\underline{z}_C$ (or $\underline{z}_L$).

2.3 Upper bounds

2.3.1 Lagrangian heuristics

From (7), it is expected that there exists $\lambda^\sharp$ satisfying

$$\frac{\partial \underline{z}(\lambda^\sharp)}{\partial \lambda} \leq 0,\ \lambda^\sharp \geq \lambda^\dagger \tag{9}$$

in the neighborhood of $\lambda^\dagger$. Although this is not true in general, in most actual computations we find such a $\lambda^\sharp$. Then, we have a feasible solution $x(\lambda^\sharp)$, and correspondingly an upper bound $\overline{z}^\sharp = z(x(\lambda^\sharp))$. We search for such a $\lambda^\sharp$ by the following.

Algorithm Lagrangian_Heuristic

Step 1. Set $t = 0$ and $\lambda^t = \lambda^\dagger$.
Step 2. Solve LMCAP(λ^t) and obtain $x(\lambda^t)$ and $\underline{z}(\lambda^t)$.
Step 3. If $\partial \underline{z}(\lambda^t)/\partial \lambda \leq 0$, output $\lambda^\sharp := \lambda^t$ and $z^\sharp := \underline{z}(x(\lambda^t))$, and stop.
Step 4. If $\partial \underline{z}(\lambda^t)/\partial \lambda_k > 0$ for some k, update $\lambda_k^{t+1} \leftarrow \max\{1.2\lambda_k^t, 0.5\}$, $t \leftarrow t+1$, and go back to Step 2.

2.4 Local search

We may improve the solution $(x^\sharp, z^\sharp)$ obtained previously by the following *local search* method. Let $x = (x_{ij})$ be a feasible solution to MCAP with the objective value $z = z(x)$. We say that $y = (y_{ij})$ is in the *2-opt neighborhood* of x, if it is obtained from x by replacing assignment of two items, say i_1 and i_2. By $N(x)$ we denote the set of 2-opt neighbors of x. Then, the local search algorithm is quite standard with this neighborhood, i.e., from a solution x we move to an improved solution $y \in N(x)$, as long as we find such a better solution in $N(x)$.

3 Pegging and virtual pegging approaches

3.1 Pegging test for BIP

Since MCAP is a *binary integer programming* (BIP) [12] problem, we can apply the *pegging test* for BIP to MCAP as well. Here we briefly summarize the pegging test for readers' convenience.

Let us consider the following BIP.

$$\text{P:}\quad \text{Minimize}\quad cx \quad \text{subject to}\ \ Ax = b,\ x \in \{0,1\}^n.$$

Let $x^\star = (x_j^\star) \in R^n$ be an optimal solution to P with the objective value $z^\star := z(x^\star)$. First, we relax the 0-1 constraints to the continuous $0 \leq x_j \leq 1, \forall j$. The resulting linear programming problem is denoted as C(P). Solving this yields an optimal solution $\underline{x}$ with the corresponding objective value $\underline{z} := z(\underline{x})$, which gives a *lower bound* to P. Next, assume that we have a feasible solution $\bar{x} \in R^n$ to P. This gives an *upper bound* $\bar{z} := z(\bar{x})$. Thus we have

$$\underline{z} \leq z^\star \leq \bar{z}.$$

Let an optimal *feasible canonical form* (FCF, [2], [3], [11]) of C(P) be

$$\bar{b}_i = x_{B(i)} + \sum_{j \in N} \alpha_{ij} x_j, \tag{10}$$

$$z = \underline{z} + \sum_{j \in N} \alpha_{0j} x_j, \tag{11}$$

where N is the index set of *non-basic variables*, and $B(i)$ denotes the index of the ith *basic variable*. From optimality of this form we have

$$\begin{aligned} \alpha_{0j} &\geq 0, \qquad \forall j \in N, \\ 0 \leq \bar{b}_i &\leq 1, \qquad i = 1, 2, \cdots, r. \end{aligned}$$

For $i = 1, 2, \cdots, r$ we define

$$PU_i := \min\{-\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} < 0\}(1 - \bar{b}_i), \tag{12}$$

$$PL_i := \min\{\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} > 0\}\bar{b}_i. \tag{13}$$

Here, if the defining set is empty, we set $\min\{\cdot \mid \emptyset\} := \infty$. Then, we have [10]

Theorem 3.

(i) *For basic variable $x_{B(i)}$ in (10),*

$$PU_i > \bar{z} - \underline{z} \Rightarrow x^\star_{B(i)} = 0, \tag{14}$$

$$PL_i > \bar{z} - \underline{z} \Rightarrow x^\star_{B(i)} = 1. \tag{15}$$

(ii) *For non-basic variable x_j ($j \in N$) in (11),*

$$\alpha_{0j} > \bar{z} - \underline{z} \Rightarrow x^\star_j = 0. \tag{16}$$

3.2 An improved reduction method for MCAP

A difficulty with the pegging test for MCAP is that the size of this problem is too large, because with n^2 columns computing all elements of (10)-(11) can be quite expensive. However, this can be remedied as follows.

Let us consider the optimal FCF given by (10) and (11). From the *unimodularity* [1], [14] of the coefficient matrix of the assignment problem, we have the following.

$$\alpha_{ij} \in \{-1, 0, 1\}, \ \forall j \in N, \tag{17}$$

$$\bar{b}_i \in \{0, 1\}, \ \forall i. \tag{18}$$

Let

$$N^+ := \{j \in N \mid \alpha_{0j} > \bar{z} - \underline{z}\}, \ N^- := \{j \in N \mid \alpha_{0j} \leq \bar{z} - \underline{z}\}. \tag{19}$$

Then, we have

Theorem 4.

(i) $\bar{b}_i = 1$ and $\{j \in N^- \mid \alpha_{ij} = 1\} = \emptyset \Rightarrow x^\star_{B(i)} = 1$,
(ii) $\bar{b}_i = 0$ and $\{j \in N^- \mid \alpha_{ij} = -1\} = \emptyset \Rightarrow x^\star_{B(i)} = 0$.

An important implication of this theorem is that, in carrying out the pegging test, we only need columns in N^-, and see if $\{j \in N^- \mid \alpha_{ij} = \pm 1\} = \emptyset$ is satisfied. Frequently, $|N^-|$ is much smaller than $|N|$, and if this is the case pegging test by Theorem 4 is far more faster than the direct application of Theorem 3.

Then, removing fixed variables, we have a reduced BIP of smaller size, and solving this obtain an optimal solution with exact $z^\star$.

3.3 A virtual pegging procedure

The usefulness of the pegging test depends on the gap between the upper and lower bounds. If the gap is not small enough, the effectiveness of the method is limited. In such a case, we may carry out the pegging test using an arbitrary value l within $[\underline{z}, \bar{z}]$ as a hypothetical upper bound. Such an l is said to be a *trial value*. As the result of this pegging with $\underline{z}$ and l, some x_{ij}'s will be fixed either at 0 or 1, and we obtain a reduced problem. The optimal objective value obtained by solving this reduced problem will be denoted as $z^\star_l$, and is referred to as the *realization* for the trial value l. If the reduced problem is infeasible, we define $z^\star_l := \infty$. Then, we have [17]

Theorem 5. *For an arbitrary trial value $l \geq \underline{z}$ and its realization $z^\star_l$, the followings hold.*

(i) $l \geq z^\star \Rightarrow z^\star_l = z^\star$,
(ii) $l < z^\star \Rightarrow z^\star_l \geq z^\star$,
(iii) $l \geq z^\star_l \Rightarrow z^\star_l = z^\star$.

Thus, in case of (iii) the solution is proved, a posteriori, optimal. Otherwise, unless $z^\star_l = \infty$ we have an approximate solution. If $z^\star_l = \infty$, we may retry with an increased l, until we have a feasible solution to MCAP.

4 Numerical experiments

We implemented the algorithms stated in the previous sections in ANSI-C language and carried out numerical experiments on an Dell Precision T7400 computer (CPU: Xeon (R) X5482, 3.20GHz, 2.00GB RAM).

4.1 Design of experiments

Instances are prepared as follows. First, c_{ij} is uniformly and independently distributed over the integer interval [1, 1000]. For (r^k_{ij}) we consider three patterns:

Dense : Here r^k_{ij} is also uniformly random over [1, 1000], independent of c_{ij} and between k's.
Sparse : The first constraint ($k = 1$) is generated similarly. However, for $2 \leq k \leq m$ 75% of r^k_{ij} are set to 0, while the remaining 25% are distributed over [1, 1000].
Disjunctive : For $2 \leq k \leq m$, we pick up a pair of rows at random, and 30% of the elements of these rows are set to 1, while the remaining 70% are 0.

To make the instance feasible, we set b_k as follows.

$$b_k = \sum_i r_{ii}^k.$$

4.2 Result of experiments

In Table 1-3 we compare two approximation methods: the Lagrangian heuristic followed by the local search (Lagrangian + LS), and the virtual pegging. Columns stand for the followings, and each row is the average over 10 random instances.

UB_1 : Upper bound obtained by Lagrangian + LS.
UB_2 : Upper bound by Virtual pegging.
LB : Lower bound.
Opt% : The percentage (out of 10 runs) that the Virtual pegging produced optimal solutions.

From these tables, we observe the followings.

- Virtual pegging gives approximate solutions which are very close to optimal. For Sparse and Disjunctive cases, the solutions is frequently proved optimal.
- Lagrangian + LS can be advantageous for Dense instances in CPU time. However, for Sparse and Disjunctive cases, the quality of the solution from this method is disastrous.

5 Conclusion

We have formulated MCAP, and presented two approximate algorithms to solve the problem. By the virtual pegging approach, we were able to solve MCAP of considerable size within a few minute. These were approximate solutions, but usually very close to optimal, and often proved optimal.

References

[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] G.B. Danzig: *Linear Programming and Extensions* (Princeton Univ. Press, Princeton, 1963).

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

[5] M.R. Garey and D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman and Company, San Francisco, 1979).

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

[7] P. Kouvelis and G. Yu: *Robust Discrete Optimization and Its Applications*, Kluwer, Dordrecht, 1997.

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

[9] P.M.D. Lieshout and A. Volgenant, "A branch-and-bound algorithm for the singly constrained assignment problem", *European Journal Operational Research*, Vol. 176, pp. 151-164. 2007.

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

Table 1: Numerical experiment (Dense)

		Lagrangian+LS		Virtual pegging		
n	k	UB1/LB	CPU_{sec}	UB2/LB	CPU_{sec}	Opt%
200	2	105.1	0.57	101.0	0.59	40
	3	106.7	0.69	101.4	0.87	10
	4	106.2	0.80	101.4	1.55	20
400	2	102.4	7.53	100.3	7.58	40
	3	103.7	11.14	100.5	11.64	20
	4	105.7	13.59	100.4	47.60	40
600	2	102.3	32.23	100.1	32.50	10
	3	104.4	43.82	100.2	51.32	10
	4	106.5	55.07	100.2	325.62	20

Table 2: Numerical experiment (Sparse)

		Lagrangian+LS		Virtual pegging		
n	k	UB1/LB	CPU_{sec}	UB2/LB	CPU_{sec}	Opt%
200	4	135.1	0.61	100.3	0.41	80
	16	214.2	0.70	100.6	0.58	70
	64	316.9	0.99	103.2	32.25	10
400	4	105.6	5.22	100.0	3.07	100
	16	174.6	5.71	100.1	3.72	100
	64	275.3	6.69	100.2	10.19	100
600	4	117.2	21.40	100.0	11.98	100
	16	152.3	23.88	100.0	14.21	100
	64	313.9	28.27	100.1	20.35	100
800	4	124.7	55.18	100.0	30.14	100
	16	160.1	60.83	100.0	33.15	100
	64	258.3	70.73	100.0	42.16	100
1000	4	100.4	56.15	100.0	58.34	100
	16	100.7	61.03	100.0	63.02	100
	64	178.1	77.85	100.0	77.80	100

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

[12] C.H. Papadimitriou and K. Steiglitz: *Combinatorial Optimization: Algorithms and Complexity* (Prentice Hall, Englewood Cliffs, 1982).

[13] S. Sakakibara and M. Nakamori: “On assignment problems with vector cost," (in Japanese),*Proc. 2005 Fall Conf. OR Soc. Japan*, 2-D-5, 2005.

[14] A. Schrijver, *Combinatorial Optimization*, Springer, 2003.

[15] R. Sedgewick: *Algorithms in C, Third Ed.* (Addison-Wesley, Reading, 1998).

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

[17] 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.

Table 3: Numerical experiment (Disjunctive)

		Lagrangian+LS		Virtual pegging		
n	k	UB1/LB	CPU_{sec}	UB2/LB	CPU_{sec}	Opt%
200	4	111.0	0.32	100.1	0.35	100
	16	143.5	0.38	100.1	0.35	100
	64	196.3	1.92	100.2	0.47	100
400	4	107.1	2.62	100.0	2.97	100
	16	212.6	3.01	100.0	3.20	100
	64	189.8	6.13	100.0	3.66	100
600	4	223.6	11.01	100.0	11.47	100
	16	280.9	11.88	100.0	11.29	100
	64	296.0	23.84	100.0	13.44	100
800	4	160.6	27.43	100.0	29.30	100
	16	221.7	28.59	100.0	29.92	100
	64	396.9	59.95	100.0	32.84	100
1000	4	100.3	55.62	100.0	57.63	100
	16	171.8	57.53	100.0	58.55	100
	64	178.0	84.80	100.0	69.33	100