The Eighth International Symposium on Operations Research and Its Applications (ISORA'09)
Zhangjiajie, China, September 20–22, 2009

Solution of Large-scale LP Problems Using MIP Solvers: Repeated Assignment Problem

Daisuke Yokoya Takeo Yamada

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

Abstract With the rapid increase in computing powers in these few decades, today we can solve LP problems with thousands of variables and constraints easily using free or commercial MIP solvers such as CPLEX and XPRESS-MP on conventional personal computers. However, we frequently encounter problems with more than a million variables/constraints, and such a problem is often intractable in our ordinary computing environment. In this paper we consider a repeated assignment problem (RAP) as an example of such a large-scale problem, and show how this can be treated by employing the delayed inclusion technique.

Keywords Linear programming, large-scale problems, MIP solver

1 Introduction

Linear programming (LP) [2] is one of the most successful OR techniques which has been used both in public and private sectors. Its root can be traced back to the work of G. Danzig [9] during WWII at the Combat Analysis Branch of the US Army's Air Corp, where they had to manage logistics of supply chains consisting of hundreds of thousands of items and people. He continued his work in the US Air Force, and there formulated the LP problem and proposed the *simplex method* to solve this in 1947 [3].

Since that time, LP has been used extensively in public sector as well as in industry. For example, an LP model was constructed to help a variety of logistics planning for the Berlin Airlift (June 1948 - Sept. 1949) [6]. Similar but far more sophisticated model was used as a part of a planning tool in the 1991 Gulf War [8]. In the cold war era, a network flow model was build to assess the transportation ability and vulnerability of the Soviet railroad system [7].

With the rapid increase in computing powers in these few decades, today we can solve LP problems with thousands of variables and constraints fairly easily using free or commercial LP (or MIP) solvers such as CPLEX [4] and XPRESS-MP [10] on conventional personal computers [5]. However, we frequently encounter problems with more than a million variables and/or constraints, and such a problem is intractable in our ordinary computing environment [12, 14, 13]. In this paper we consider a *repeated assignment problem* as an example of such a large-scale problem, and show how this can be treated by employing the *delayed inclusion* (of rows and columns) technique.

2 Repeated assignment problem

Repeated assignment problem (RAP) is defined as the K-fold repetition of the $n \times n$ assignment problem, with the additional requirement that no assignment can be repeated more than once. Mathematically, the problem is formulated as the following.

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

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

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

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

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

Here, c_{ij}^k is the *cost* of assigning i to j at k-th repetition.

The case of $K = 1$ is the standard assignment problem, which can be solved in polynomial time using, e.g., the *Hungarian method* [1]. Another special case of **RAP** where c_{ij}^k is constant over $k = 1,2,\ldots,K$ can be reduced to the solution of a minimum cost flow problem and $K-1$ maximum flow problems, and thus solvable in polynomial time [11]. However, it is not clear (to us) whether the general **RAP** is $\mathscr{NP}$-hard or not.

By $\textbf{RAP}_0$ we denote the *continuous relaxation* of this problem where (5) is replaced with $x_{ij}^k \ge 0$. Eliminating apparent redundancy in (2) and (3), $\textbf{RAP}_0$ is an LP problem with $(2n-1)K+n^2$ constraints and n^2K variables. Thus, for $n = 1000$ and $K = 10$ we have an LP problem of $1,019,990 \times 10,000,000$, which is hard to solve on ordinary personal computers using available solvers.

3 Delayed inclusion approach

Consider an LP problem

$$\textbf{P}: \quad \text{Maximize} \quad c^T x \tag{6}$$

$$\text{subject to} \quad Ax \le b,\ x \ge 0 \tag{7}$$

with an optimal solution $x^* = (x_j^*)$. We call column j *zero* (with respect to x^*) if $x_j^* = 0$. Otherwise, column j is *non-zero*. Also row i is said to be *active* if equality holds in the ith constraint of (7) at x^*. If it is not active, the row is *inactive*. Then, if we correctly know which columns are zero/non-zero, and which rows are active/inactive, we can derive a smaller LP problem by eliminating all the zero columns and inactive rows, and solving this obtain an optimal solution to **P**. Even if we do not have exact knowledge on this, often we can solve the original LP problem by solving much smaller problems as follows.

Let $\bar{C}$ and $\bar{R}$ denote, respectively, the sets of all constraints and all variables of **P**, and corresponding to an arbitrary pair of subsets $C \subseteq \bar{C}$ and $R \subseteq \bar{R}$, we introduce an LP problem $\textbf{P}(R,C)$ as the restriction of **P** to this part. Partitioning the original simplex tableau as

	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

we have $\mathbf{P}(R,C)$ explicitly written as

$$\mathbf{P}(R,C): \quad \text{Maximize} \quad c_0^T x \tag{8}$$
$$\text{subject to} \quad A_{00}x \leq b_0 \tag{9}$$

Let a pair of primal and dual optimal solutions to this problem be $x^*(R,C)$, $y^*(R,C)$. Then, we have

Theorem 1: *If*

(i) $A_{10}x^*(R,C) \leq b_1$ (primal feasibility) *and*
(ii) $y^*(R,C)^T A_{01} \geq c_1^T$ (dual feasibility)

are both satisfied, then the vectors obtained by filling zeros to the parts of C' and R', i.e., $x^* := (x^*(R,C), \mathbf{0})$ and $y^* := (y^*(R,C), \mathbf{0})$, are optimal to **P** *and* **D***, respectively.*

Proof: Straightforward from the duality of LP problems. □

Note that in this theorem the constraints in R' are *inactive*, and the variables in C' are *zero* in optimality. If we know these inactive rows and zero columns *a priori*, we can obtain an optimal solution to **P** by solving a (usually) much smaller problem $\mathbf{P}(R,C)$. Unfortunately, we do not know exactly which rows/columns can be thus eliminated until we completely solve **P**. So, we propose the following approach. We start with a guess of these sets, i.e., we take R_0 as a set of *plausibly* active constraints, and C_0 a set of seemingly non-zero variables. Then, after solving the reduced problem, if some feasibility conditions are not satisfied, we include the violated rows/columns and repeat the process all over again. More precisely, the algorithm is as follows.

Algorithm DELAYED-INCLUSION.

Step 1. Take an arbitrary pair of R_0 and C_0, and put $(R,C) := (R_0, C_0)$.
Step 2. (Using MIP solver such as CPLEX) Solve $P(R,C)$, and obtain $x^* = x^*(R,C)$, $y^* = y^*(R,C)$.
Step 3. If there exist rows violating $A_{10}x^* \leq b_1$, add these rows to R.
Step 4. If there exists columns violating $y^*A_{01} \geq c_1$, add these columns to C.
Step 5. If there exist neither violating rows nor columns, x^* and y^* (supplemented with appropriate 0 elements) solve **P**. Thus, output these and stop. Otherwise, go back to Step 2.

In Step. 2 above, $\mathbf{P}(R,C)$ may be solved from scratch each time as a new problem. Or, better than that, we can add the violated rows and columns to the optimal simplex

tableau obtained at the previous iteration and solve the augmented $\mathbf{P}(R,C)$ more quickly. If $\mathbf{P}(R,C)$ is always feasible in DELAYED-INCLUSION, this clearly solves $\mathbf{P}$. Specifically, if $\mathbf{P}(C_0,\bar{R})$ is feasible, it is easily proved that $\mathbf{P}(R,C)$ is always feasible, since $C_0 \subseteq C$ and $\bar{R} \supseteq R$.

4 Application to RAP

In applying DELAYED-INCLUSION to the problem $\mathbf{RAP}_0$, we need to specify the starting pair (R_0,C_0) of the sets of rows and columns. As R_0 we take constraints (2) and (3), which are always active in any optimal solutions to $\mathbf{RAP}_0$. Contrary to this, most of the constraints (4) are expected to be inactive.

On the other hand, appropriate choice of starting C_0 is not so clear. To determine this, we can make use of a feasible solution to **RAP**. Indeed, if $\bar{x} = (\bar{x}^k_{ij})$ is such a solution, we define the initial set of columns C_0 as

$$C_0 := \{(i,j,k) \mid \bar{x}^k_{ij} = 1, 1 \le i,j \le n, 1 \le k \le K\}. \tag{10}$$

Clearly, in this case $\mathbf{P}(\bar{R},C_0)$ is feasible, and therefore DELAYED-INCLUSION solves $\mathbf{RAP}_0$ correctly. Moreover, if $\bar{x}$ is a 'good' approximation, it is expected that most of the columns of C_0 are actually non-zero in optimality. The choice of (R_0,C_0) in DELAYED-INCLUSION as stated above is referred to as STRATEGY1.

Alternatively, we may take C_0 as the set with some other variables added to the C_0 of STRATEGY1. Natural candidates for this are the variables x^k_{ij} with small c^k_{ij}. In STRATEGY2, for each k we add to the above C_0 the variables of the smallest and second smallest costs in each row and column.

Then, the remaining question is how we obtain a good feasible solution $\bar{x}$ to **RAP**. We propose the *repeated Hungarian method* for this purpose. Let $F \subseteq \{(i,j) \mid 1 \le i,j \le n\}$ be the *forbidden* pair of assignment, which is initially empty, and $\mathrm{AP}^k(F)$ denotes the assignment problem with the modified cost matrix $\bar{c}^k = (\bar{c}^k_{ij})$ defined as

$$\bar{c}^k_{ij} = \begin{cases} c^k_{ij}, & (i,j) \notin X \\ \infty, & (i,j) \in X \end{cases} \tag{11}$$

The bipartite graph associated with this assignment problem is denoted as $H^k(F)$, and applying the Hungarian method to this problem, we obtain a complete matching $M^k(F)$ in $H^k(F)$. The edges in $H^k(F)$ are then excluded from the subsequent assignment by putting $F := F \cup H^k(F)$. The repeated Hungarian method is formally given as follows.

Algorithm REPEATED-HUNGARIAN.

Step 1. Set $k := 0$ and $F := \emptyset$.
Step 2. Using Hungarian method solve $\mathrm{AP}^k(F)$ and obtain an optimal bipartite matching $H^k(F)$.
Step 3. If $k \ge K$ stop. Otherwise, put $k := k+1$ and $F := F \cup M^k(F)$ and go to Step 2.

Figure 1 shows REPEATED-HUNGARIAN for an instance with $n = 5$ and $K = 3$, where thick lines represent $M^k(F)$ which is removed from the bipartite graph of the next step. Thus, we obtain a feasible solution to **RAP** with the objective value $z = 4921$.

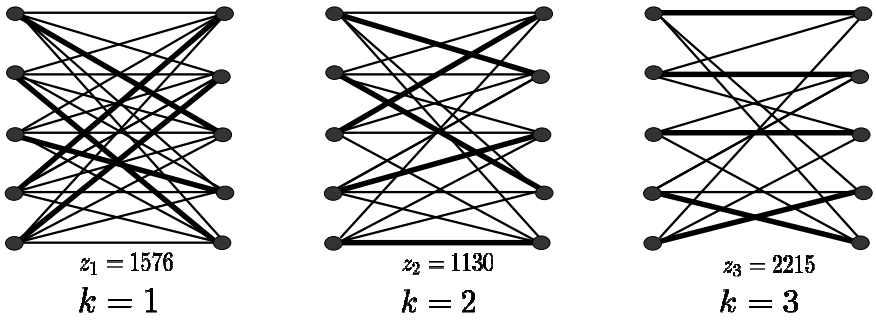

Figure 1: REPEATED-HUNGARIAN for $n = 5$ and $K = 3$.

The correctness of this method can be shown by considering the following.

Lemma: *For any $k \leq n$, there exists a complete matching in $H^k(F)$.*

Proof: Let V_1 and V_2 be the sets of left and right nodes of $H^k(F)$, respectively. For an arbitrary $U \subseteq V_1$, $N(U) \subseteq V_2$ is the set of nodes which are adjacent to U. Here, we note that the node-degree of each node of $H^k(F)$ is $n-k+1$. Then, the numbers of edges incident to U and $N(U)$ are, respectively, $(n-k+1)|U|$ and $(n-k+1)|N(U)|$. Also, all the edges incident to U is incident to $N(U)$, but not *vice versa*. Thus, we have $|U| \leq |N(U)|$, and by Hall's theorem [7] the proof is complete. □

From this the following is straightforward.

Theorem 2: REPEATED-HUNGARIAN *gives a feasible solution to* **RAP**.

5 A numerical example

Let us consider the case of $n = 100$ and $K = 10$. The size of the original $\mathbf{RAP}_0$ is $11,990 \times 100,000$, and solving this directly with CPLEX we obtain the optimal value as $z^* = 24481.9$.

Figure 1 shows the behavior of our algorithm with STRATEGY1. Here shown are the numbers of rows and columns, as well as the optimal objective value at each iteration. The size of the initial LP problem is 1990×1000, and solving this with CPLEX we obtain the optimal value 24288.0 to this problem. Next, we include the violated rows and columns and solve the augmented LP problem. Repeating this we obtain the same $z^\star = 23481.9$ in 14 iterations. The size of the final problem is 2154×4007, which is much smaller than the original problem.

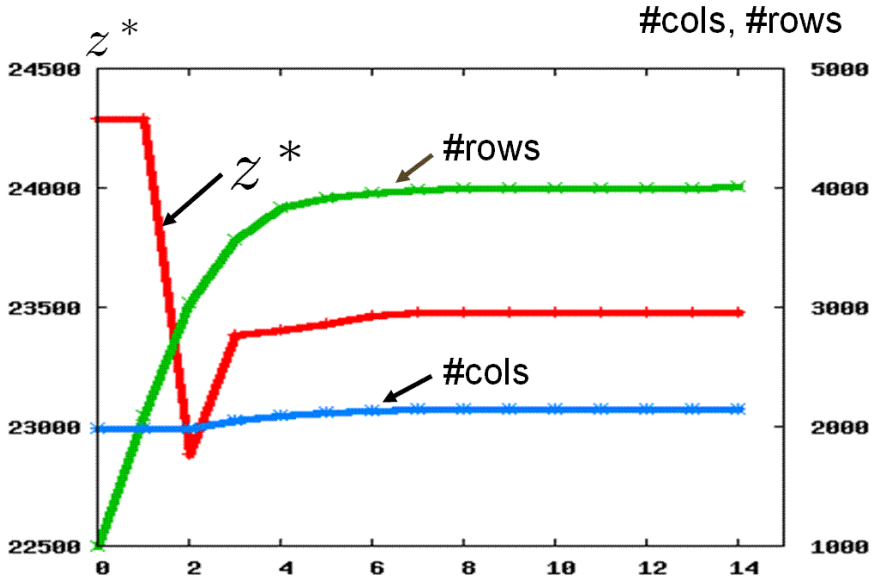

Figure 2: The behavior of DELAYED-INCLUSION.

6 Numerical experiments

This section gives the results of numerical experiments conducted to compare the strategies as well as to evaluate the performance of the proposed algorithm. We implemented DELAYED-INCLUSION algorithm in ANSI C language which solved $\mathbf{P}(R,C)$ by calling an LP library of CPLEX 10.1 [4], and carried out computation on an Dell DIMENSION 8400 computer (Pentium(R)4 CPU, 3.40GHz, 2.00GB RAM). The instances were prepared in the following way. First, a *nominal cost* c^0_{ij} is assumed to be uniformly random over the integer interval [1,1000], and the cost c^k_{ij} at k-th repetition is determined as a uniformly random integer over [Floor, Ceil], where

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

Here, σ denotes the parameter representing the degree of correlation between the costs over $k = 1,2,\ldots,K$. This means no correlation for $\sigma = 0.0$ and complete correlation for $\sigma = 1.0$. Figure 2 is the case of $\sigma = 0.4$.

Tables 1 and 2 summarize the results of STRATEGY1 and STRATEGY2 for instances of $n = 100$ with $K = 4/8/16$ and $\sigma = 0.0/0.4/0.8$, respectively. Here shown are the numbers of rows and columns at the initial and final stages, the CPU time in seconds, and 'cycle' which is the number of iterations of Steps 2 through 5 in DELAYED-INCLUSION before obtaining the optimal solution. Each row is the average over 10 randomly generated instances. We see that the numbers of rows and columns at the final stage usually increase with σ. However, to discuss advantages between two strategies, more extensive numerical experiments are required.

Table 3 gives the result of STRATEGY1 for larger problems with $n = 200/400/600$. The number of cycles increases with K, while it is almost insensitive to the increase of σ.

On the other hand, CPU time increases both with K and σ The increase of the numbers of rows (#R) between initial and final stages remains small for all cases tested, while the number of columns (#C) at the final stage is approximately four times larger than that at the initial stage.

Table 1: STRATEGY1 ($n = 100$)

K	σ	z^*	cycle	Initial		Final		CPUsec
				#R	#C	#R	#C	
4	0.0	6741.7	8.3	796	400	803.8	1513.4	0.23
	0.4	9137.8	8.0	796	400	806.6	1466.4	0.23
	0.8	9231.0	9.5	796	400	847.3	1644.3	0.38
8	0.0	13583.4	10.8	1592	800	1638.7	3032.2	1.09
	0.4	18495.5	11.5	1592	800	1678.5	3164.4	1.39
	0.8	19350.7	11.1	1592	800	1919.1	4084.0	5.32
16	0.0	27726.1	14.4	3184	1600	3438.8	6398.5	8.46
	0.4	38453.0	14.4	3184	1600	3666.5	7321.2	20.02
	0.8	44166.8	11.8	3184	1600	4405.3	10344.8	146.07

Table 2: STRATEGY2 ($n = 100$)

K	σ	z^*	cycle	Initial		Final		CPUsec
				#R	#C	#R	#C	
4	0.0	6741.7	5.9	796	1135.8	803.6	1554.4	0.22
	0.4	9137.8	6.0	796	1131.0	807.1	1554.1	0.23
	0.8	9231.0	7.9	796	1140.6	849.1	1567.6	0.38
8	0.0	13583.4	7.9	1592	2277.6	1640.2	3106.9	1.00
	0.4	18495.5	9.2	1592	2277.6	1678.5	3148.7	1.33
	0.8	19350.7	8.9	1592	2327.5	1924.4	3361.4	4.70
16	0.0	27726.1	11.6	3184	4574.3	3436.5	6337.8	8.15
	0.4	38453.0	12.4	3184	4609.3	3665.9	6520.7	20.09
	0.8	44166.8	9.6	3184	4853.1	4420.6	7715.3	112.41

7 Conclusion

We have proposed a delayed inclusion approach to solve large-scale LP problems using MIP solvers. As a future work, we plan to extend application of this method to other type of large-scale problems.

References

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

[2] G.B. Danzig, *Linear Programming and Extensions*, Princeton U. Press, 1963.

[3] S.I. Gass and A.A. Assad, *An Annotated Timeline of Operations Research*, Kluwer, 2005.

[4] ILOG CPLEX 10.0, http://ilog.com/products/cplex, 2007.

[5] R. Miyashiro and T. Matsui, “Evaluating the power of some recent MIP solvers" (in Japanese), *System/Information/Control*, Vol. 50, pp. 363-368, 2006AD

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

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

[8] Th. Schuppe, “OR goes to war," *OR/MS Today*, April, pp. 36-44, 1991.

[9] Wikipedia, *George Danzig*, 2007.

[10] XPRESS-MP 2007A, Dash Optimization, http://www.dash optimization.com, 2007.

[11] Y. Yajima and E. Kosaka, Multi-term class assignment problem" (in Japanese), *Communications of the OR Society of Japan*, Vol. 40, pp. 421-424,1995.

[12] 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.

[13] T. Yamada and T. Takeoka, “An exact algorithm for the fixed-charge multiple knapsack problem," to appear in *European Journal of Operational Research*, 2007.

[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.