Manuscript Number: EJOR-D-09-01755

Title: A Reduction Approach to the Repeated Assignment Problem

Article Type: Theory and Methodology Paper

Section/Category: Combinatorial optimization

Keywords: repeated assignment problem; combinatorial optimization; relaxation; pegging test.

Corresponding Author: Ph. D. Student Daisuke Yokoya, M.S.

Corresponding Author's Institution: National Defense Academy

First Author: Daisuke Yokoya, M.S.

Order of Authors: Daisuke Yokoya, M.S.; Takeo Yamada, Ph.D.

Abstract: We formulate and solve the repeated assignment problem, which is a K-fold repetition of the $n \times n$ assignment problem, with the additional requirement that no assignments can be repeated more than once. We present a repeated Hungarian method to derive an upper bound, and continuous and Lagrangian relaxations to obtain lower bounds. The Lagrangian relaxation problem decomposes into a set of K independent assignment problems, and by applying the pegging test to each of the decomposed problem, the RAP is (often significantly) reduced in size and can be solved using MIP solvers. In addition, the pegging test is simplified for the assignment problem, and we present a `virtual' pegging test for further reduction of computation. Through numerical experiments, we examine the performance of the developed method to solve RAP.

A Reduction Approach to the Repeated Assignment Problem

Daisuke Yokoya and Takeo Yamada*

Department of Computer Science, The National Defense Academy,

Yokosuka, Kanagawa 239-8686, Japan

Abstract

We formulate and solve the repeated assignment problem, which is a K-fold repetition of the $n \times n$ assignment problem, with the additional requirement that no assignments can be repeated more than once. We present a repeated Hungarian method to derive an upper bound, and continuous and Lagrangian relaxations to obtain lower bounds. The Lagrangian relaxation problem decomposes into a set of K independent assignment problems, and by applying the pegging test to each of the decomposed problem, the RAP is (often significantly) reduced in size and can be solved using MIP solvers. In addition, the pegging test is simplified for the assignment problem, and we present a 'virtual' pegging test for further reduction of computation. Through numerical experiments, we examine the performance of the developed method to solve RAP.

Keywords: Repeated assignment problem, Combinatorial optimization, Relaxation, Pegging test.

1 Introduction

We are concerned with the *repeated assignment problem* (RAP), which is a K-fold repetition of the $n \times n$ *assignment problem* (AP, [1]), with the additional requirement that no assignments can be repeated more than once. Mathematically, the problem is formulated as the following *binary integer program* (BIP [13]).

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

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

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

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

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

*Corresponding author, E-mail: yamada@nda.ac.jp, Fax: +81-46-844-5911

Here, c_{ij}^k is the *cost* of assigning i to j at the k-th repetition, and x_{ij}^k is the decision variable such that $x_{ij}^k = 1$ in the above assignment, and $x_{ij}^k = 0$ otherwise.

The case of $K = 1$ is the standard assignment problem, which can be solved in polynomial time using, e.g., the *Hungarian method* [1, 7]. Another special case of RAP where c_{ij}^k is constant for all k, i.e., $c_{ij}^k \equiv c_{ij}$, can be reduced to the solution of a minimum cost flow problem and $K - 1$ maximum flow problems, and thus solvable in polynomial time [14]. However, it is not clear (to the authors) whether the general RAP is $\mathcal{NP}$-hard [5] or not. Since RAP is a linear 0-1 programming problem, small instances can be solved using commercial or free *mixed integer program* (MIP) solvers. The purpose of this paper is to present a procedure that specializes in solving larger RAPs.

The organization of the paper is as follows. In section 2, we present a greedy heuristic termed as the *repeated Hungarian method*, and prove that this works correctly to give an upper bound to RAP. Next, we derive a lower bound by continuously relaxing the constraint (5). To solve the resulting *linear program* (LP), which is often of a huge size, we present a *delayed-inclusion* method. The same lower bound can be obtained by the Lagrangian relaxation [4] of (4), provided that we use the 'correct' Lagrangian multipliers. From the result of continuous-relaxation, we obtain such a set of multipliers, and the Lagrangian relaxation problem is decomposed into K independent assignment problems.

Next, in Section 3, we explain how the *pegging test* (to reduce the size of general BIP, [8]) can be tailored for RAP. By applying this test, some variables are fixed either at 0 or 1, and removing these fixed variables RAP is reduced (often significantly) in size. We can solve the reduced problem using commercial or free MIP solvers.

To apply the pegging test to each of the decomposed assignment problem, however, we need a *feasible canonical form* (FCF, [2, 9]) of the problem in optimality. Usually, assignment problem is solved using algorithms such as the Hungarian method, but these efficient methods do not produce optimal FCFs. We show how such an FCF can be reconstructed from the result of the Hungarian method. Furthermore, from the *unimodularity* [13, 10] of the assignment matrix, the pegging test can be simplified in this case. Through the improved pegging procedure, we obtain the same pegging result with much smaller amount of computation.

Finally, in Section 4 we introduce the *virtual pegging test* [16] to further reduce the amount of computation, and examine the performance of the developed method to solve RAP in a series of numerical tests for randomly generated instances of various statistical characteristics.

2 Upper and lower bounds

Here we discuss upper and lower bounds to RAP. Continuous and Lagrangian relaxations give the same lower bound. We introduce the continuous relaxation to find the optimal Lagrangian multipliers, and the latter is substantial in reducing the size of the problem in later sections.

2.1 Repeated Hungarian method

We propose the repeated Hungarian method to find a feasible solution and correspondingly an upper bound to RAP. Let $F \subseteq \{(i, j) \mid 1 \leq i, j \leq n\}$ be the set of $forbidden$ pairs of assignment, which is initially empty, and $\mathrm{AP}^k(F)$ denotes the assignment problem with the modified cost matrix $(\tilde{c}^k_{ij})$ defined as

$$\tilde{c}^k_{ij} = \begin{cases} c^k_{ij}, & (i,j) \notin F, \\ \infty, & (i,j) \in F. \end{cases}$$

The *bipartite graph* [1] associated with this assignment problem is denoted as $H^k(F)$, and applying the Hungarian method, we obtain a *complete matching* $M^k(F)$ in $H^k(F)$. The edges in $M^k(F)$ are then removed from the subsequent assignments by putting $F := F \cup M^k(F)$. The repeated Hungarian method is formally given as follows.

```
Algorithm REPEATED-HUNGARIAN.

Step 1. Set $k := 1$ and $F := \emptyset$.

Step 2. Using the Hungarian method solve AP$^k(F)$ and obtain an optimal bipartite
        matching $M^k(F)$.

Step 3. If $k = K$ stop. Otherwise, put $k := k+1$, $F := F \cup M^k(F)$, and go back to Step 2.
```

This algorithm stops at the K-th iteration with a feasible solution to RAP, and the corresponding upper bound is denoted as $\bar{z}_{RAP}$. The correctness of this method is shown through the following.

Lemma 1 *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 necessarily *vice versa*. Thus, we have $|U| \leq |N(U)|$, and by Hall's theorem [11] the proof is complete. ∎

From this the following is straightforward.

Theorem 1 REPEATED-HUNGARIAN *gives a feasible solution to RAP.*

Example 1 *We consider a RAP with $n = 4, K = 2$ and the following cost matrices.*

$$C^1 = \begin{pmatrix} 48 & 8 & \underline{6} & 60 \\ 125 & 48 & 41 & \underline{33} \\ \underline{18} & 97 & 58 & 88 \\ 59 & \underline{23} & 46 & 56 \end{pmatrix}, \quad C^2 = \begin{pmatrix} 73 & \underline{24} & 19 & 79 \\ 89 & 6 & \underline{9} & 25 \\ 44 & 72 & 108 & \underline{30} \\ \underline{101} & 24 & 64 & 103 \end{pmatrix}.$$

Underlined in these matrices indicate a feasible solution obtained by the repeated Hungarian method, with the corresponding upper bound $\bar{z}_{RAP} = 244$.

2.2 Lower bound by continuous relaxation

Let C(RAP) be the *continuous relaxation* of RAP where (5) is replaced with $x_{ij}^k \geq 0$. This is an LP problem with $2nK+n^2$ constraints and n^2K variables. Thus, for $n = 1000$ and $K = 10$ we have an LP problem with 1,020,000 rows and 10,000,000 columns, which is hard to solve on ordinary personal computers using available solvers. Here, we present a delayed-inclusion approach to solve such a large LP problem.

2.2.1 Delayed-inclusion method

Let us consider an LP problem

$$\text{P : maximize } c^T x \quad \text{subject to } \; Ax \leq b, \; x \geq 0,$$

and let $\bar{C}$ and $\bar{R}$ denote, respectively, the sets of all constraints and all variables of P. In matrix A, column j is said to be *zero* (with respect to an optimal solution x^* of P) if $x_j^* = 0$. Also row i is *active* if equality holds in the ith constraint of P at x^*.

Corresponding to an arbitrary pair of subsets $C \subseteq \bar{C}$ and $R \subseteq \bar{R}$, we introduce an LP problem P(R, C) as the restriction of P to this part. Partitioning the matrix as

$$\begin{array}{r|cc|c|}
 & C & C' & \text{const} \\ \hline
R & A_{00} & A_{01} & b_0 \\
R' & A_{10} & A_{11} & b_1 \\ \hline
\text{obj} & c_0^T & c_1^T & 0 \\ \hline
\end{array}$$

we have P(R, C) explicitly written as

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

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

Theorem 2 *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 remaining parts C' and R', i.e., $x^* := (x^*(R, C), 0)$ and $y^* := (y^*(R, C), 0)$, are optimal to P and its dual D, respectively.

Proof: Straightforward from the duality of LP problems. ∎

Here, if we know correct partition (R, C) *a priori* so that (i) and (ii) in Theorem 2 are satisfied, we can obtain an optimal solution to P by solving (usually) a much smaller problem P(R, C). Unfortunately, we do not know exactly which rows/columns can be thus eliminated until we completely solve P. In the delayed-inclusion 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 in optimality. Then, after solving the reduced problem, if some feasibility conditions are not satisfied, we revive 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. Solve $P(R, C)$, and obtain $x^* = x^*(R, C)$ and $y^* = y^*(R, C)$.

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 neither violating rows nor columns exist, $x^*$ and $y^*$ (supplemented with ap-
        propriate 0 elements) solve P and D, respectively. Thus, output these and stop.
        Otherwise, go back to Step 2.
```

In Step 2 above, P(R, C) may be solved from scratch each time as a new LP problem. Or, better than that, we can add the violated rows and columns to the optimal simplex tableau at the previous iteration and solve the augmented P(R, C) more quickly. If P(R, C) is always feasible in DELAYED-INCLUSION, this clearly solves P. Specifically, if P$(C_0, \bar{R})$ is feasible, it is easily proved that P(R, C) is always feasible, since $C_0 \subseteq C$ and $\bar{R} \supseteq R$.

2.2.2 Application to RAP

In applying DELAYED-INCLUSION to C(RAP), 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 C(RAP). Contrary to this, most of the constraints (4) are usually inactive.

On the other hand, appropriate choice of starting C_0 is not so clear. To determine this, we make use of the solution obtained by REPEATED-HUNGARIAN. Indeed, with the solution $\bar{x} = (\bar{x}_{ij}^k)$ from the algorithm we define the initial set of columns C_0 as

$$C_0 := \{(i, j, k) \mid \bar{x}_{ij}^k = 1, \ \forall i, j, k\}.$$

Clearly, in this case P$(\bar{R}, C_0)$ is feasible, and therefore DELAYED-INCLUSION solves C(RAP) 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 lower bound obtained by solving C(RAP) this way is denoted as $\underline{z}_C$.

Example 2 *Applying DELAYED-INCLUSION to the RAP of Example 1, we obtain a lower bound* $\underline{z}_C = 238.5$ *after solving 7 LP problems of at most* 18×21*. In Table 1, we show the numbers of rows and columns, as well as the objective value at each iteration. The same lower bound is obtained by solving C(RAP) directly.*

Table 1: Behavior of DELAYED-INCLUSION.

Cycle	*#Row*	*#Col*	$\underline{z}_C$
1	14	8	244.0
2	14	12	236.0
3	15	14	216.0
4	16	15	237.0
5	18	16	244.0
6	18	19	238.5
7	18	21	238.5

2.3 Lagrangian relaxation

With nonnegative multipliers $\gamma = (\gamma_{ij})$ associated with (4), the Lagrangian relaxation of RAP is defined as

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

For a fixed $\gamma \geq 0$, this can be decomposed into K independent assignment problems.

$$\begin{aligned} \text{AP}^k(\gamma): \quad \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}^k + \gamma_{ij})x_{ij}^k & (6) \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij}^k = 1, \quad \forall i, & (7) \\ & \sum_{i=1}^{n} x_{ij}^k = 1, \quad \forall j, & (8) \\ & x_{ij}^k \in \{0,1\}, \quad \forall i, j. & (9) \end{aligned}$$

Let the optimal objective values to LRAP(γ) and AP^k(γ) be $\underline{z}(\gamma)$ and $\underline{z}^k(\gamma)$ respectively, and consider the *Lagrangian dual*:

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

The optimal objective value to this problem, denoted as $\underline{z}_L$, gives the lower bound by the Lagrangian relaxation.

Let $(u^\dagger, v^\dagger, \gamma^\dagger) \in R^{nK} \times R^{nK} \times R_+^{n^2}$ be an optimal solution to the *dual* of C(RAP), where $u^\dagger$, $v^\dagger$ and $\gamma^\dagger$ correspond to (2), (3) and (4), respectively. Then, we have the following.

Theorem 3 *$\gamma^\dagger$ gives an optimal solution to the Lagrangian dual, That is, $\underline{z}_L = \underline{z}(\gamma^\dagger)$. Moreover, this coincides with the lower bound obtained by the continuous relaxation, i.e., $\underline{z}_L = \underline{z}_C$.*

Proof: Since LRAP($\gamma^\dagger$) is K independent assignment problems, all extreme points of the feasible region of (2), (3), (5) are integral. Then this theorem follows from Theorem 10.3 of Wolsey [13]. ∎

Thus, in what follows we write $\underline{z}_{RAP}$, $\underline{z}^k$ and AP^k instead of $\underline{z}_C$ (or $\underline{z}_L$), $\underline{z}^k(\gamma^\dagger)$, and AP$^k(\gamma^\dagger)$, respectively. The lower bound is then given as

$$\underline{z}_{RAP} = \sum_{k=1}^{K} \underline{z}^k - \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij}^\dagger. \tag{10}$$

Example 3 *From the solution to C(RAP), we obtain the optimal $\gamma^\dagger$ as*

$$\gamma^\dagger = \begin{pmatrix} 0.0 & 0.0 & 2.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 8.0 \\ 13.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 22.5 & 0.0 & 0.0 \end{pmatrix}.$$

For $k = 1$, we solve AP^1 with cost matrix

$$\bar{C}^1 := C^1 + \gamma^\dagger = \begin{pmatrix} 48.0 & \underline{8.0} & 8.5 & 60.0 \\ 125.0 & 48.0 & 41.0 & \underline{41.0} \\ \underline{31.5} & 97.0 & 58.0 & 88.0 \\ 59.0 & 45.5 & \underline{46.0} & 56.0 \end{pmatrix},$$

and obtain the optimal matching as underlined above, with the corresponding objective value $\underline{z}^1 = 126.5$. Similarly, we have $\underline{z}^2 = 158.5$, and from (10) obtain $\underline{z} = 238.5$ again.

3 Pegging test

Let δ be either 0 or 1, and define $Gap := \bar{z}_{RAP} - \underline{z}_{RAP}$ for the upper and lower bounds obtained previously. By RAP$(x_{ij}^k = \delta)$ we mean RAP with one more constraint $x_{ij}^k = \delta$ added, and $\underline{z}(x_{ij}^k = \delta)$ denotes the lower bound to this modified problem. Then, if $\underline{z}(x_{ij}^k = \delta) \geq \bar{z}_{RAP}$, no better solution than $\bar{z}_{RAP}$ can be expected by fixing $x_{ij}^k = \delta$, and thus we conclude that $x_{ij}^k = \delta'$ in optimality, where $\delta' := 1 - \delta$.

Then, to evaluate $\underline{z}(x_{ij}^k = \delta)$ we introduce an assignment problem AP$^k(x_{ij}^k = \delta)$ as AP^k with $x_{ij}^k = \delta$ added, and its optimal objective value is denoted as $\underline{z}^k(x_{ij}^k = \delta)$. Then, from (10) we have

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

Thus, we can fix x_{ij}^k at δ' if

$$\underline{z}^k(x_{ij}^k = \delta) - \underline{z}^k \geq gap. \tag{11}$$

To determine if (11) is satisfied quickly, we consider the *pegging test* for general BIP first.

3.1 Pegging test for general BIP

Here we briefly summarize some basic results on the pegging test [8, 15] for readers' convenience. For simplicity of notation, let us consider the following BIP.

$$\text{Q: minimize } z(x) := c^T x \quad \text{subject to } Ax = b,\ x_j \in \{0, 1\},\ \forall j.$$

Let $x^\star = (x_j^\star) \in R^n$ be an optimal solution to Q with the objective value $z^\star := z(x^\star)$. First, we relax the 0-1 constraints to the continuous

$$0 \leq x_j \leq 1, \quad \forall j.$$

The resulting LP is denoted as C(Q). Solving this yields an optimal solution $\underline{x}$ with the corresponding objective value $\underline{z} := z(\underline{x})$, which gives a lower bound to Q. Next, assume that we have a feasible solution $\overline{x} \in R^n$ to Q, and corresponding upper bound $\overline{z} := z(\overline{x})$. Thus we have

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

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

$$\overline{b_i} = x_{B(i)} + \sum_{j \in N} \alpha_{ij} x_j, \tag{12}$$

$$z = \underline{z} + \sum_{j \in N} \alpha_{0j} x_j, \tag{13}$$

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

$$\begin{aligned} \alpha_{0j} \geq 0, &\qquad \forall j \in N, \\ 0 \leq \overline{b_i} \leq 1, &\qquad \forall i. \end{aligned}$$

For each i we define

$$PU_i := \min\{-\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} < 0\}(1 - \overline{b_i}), \tag{14}$$

$$PL_i := \min\{\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} > 0\}\overline{b_i}. \tag{15}$$

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

Theorem 4 [8]

(i) *For basic variable $x_{B(i)}$ in (12),*

$$PU_i > \overline{z} - \underline{z} \Rightarrow x^\star_{B(i)} = 0, \tag{16}$$

$$PL_i > \overline{z} - \underline{z} \Rightarrow x^\star_{B(i)} = 1. \tag{17}$$

(ii) *For non-basic variable x_j ($j \in N$) in (13),*

$$\alpha_{0j} > \overline{z} - \underline{z} \Rightarrow x^\star_j = 0. \tag{18}$$

3.2 Pegging test for RAP

By solving an assignment problem AP^k using the Hungarian method, we obtain an optimal assignment $x^\dagger = (x^\dagger_{ij})$ as well as an optimal solution $(u^\dagger, v^\dagger)$ to the dual of AP (superscript k is dropped in the remainder of this section), which is given as

$$\text{DAP:} \quad \text{maximize} \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{19}$$

$$\text{subject to} \quad u_i + v_j \leq \overline{c}_{ij}, \ \forall i, j. \tag{20}$$

Here, we write $\overline{c}_{ij} := c_{ij} + \gamma^{\dagger}_{ij}$, and these satisfy the *complementary slackness condition* [2]:

$$x^{\dagger}_{ij} = 1 \Rightarrow u^{\dagger}_{i} + v^{\dagger}_{j} = \overline{c}_{ij}. \tag{21}$$

However, to apply Theorem 4 to AP^k and see if (16) - (18) are satisfied, we need an optimal FCF of the problem. Here we explain how such an FCF can be reconstructed from the output of the Hungarian method.

Corresponding to $(u^{\dagger}, v^{\dagger})$, we introduce an undirected *bipartite graph* $H(u^{\dagger}, v^{\dagger})$ consisting of the left and right sets of nodes $L = \{l_1, l_2, \ldots, l_n\}$, $R = \{r_1, r_2, \ldots, r_n\}$ and the set of arcs $A(u^{\dagger}, v^{\dagger}) = \{(l_i, r_j) \in L \times R \mid u^{\dagger}_{i} + v^{\dagger}_{j} = \overline{c}_{ij}\}$. This includes a *perfect matching* $M = \{(l_i, r_j) \in L \times R \mid x^{\dagger}_{ij} = 1\}$ in $H(u^{\dagger}, v^{\dagger})$. Now, if $H(u^{\dagger}, v^{\dagger})$ is unconnected, we can modify $(u^{\dagger}, v^{\dagger})$ by executing the following steps repeatedly until finally the graph is connected. Let us consider the *connected components* of $H(u^{\dagger}, v^{\dagger})$, and by comp$(\cdot)$ we denote the component to which node $\cdot$ belongs.

Procedure DUAL-UPDATING

Step 1. Decompose $H(u^{\dagger}, v^{\dagger})$ into connected components.

Step 2. Find $\alpha := \min\{\overline{c}_{ij} - u^{\dagger}_{i} - v^{\dagger}_{j} \mid (l_i, r_j) \in L \times R,\ \text{comp}(l_i) \neq \text{comp}(r_j)\}$, and let $(l_{\underline{i}}, r_{\underline{j}})$ be a pair where the minimum is attained.

Step 3. Modify $(u^{\dagger}, v^{\dagger})$ according to:

$$u^{\dagger}_{i} \leftarrow u^{\dagger}_{i} - \alpha,\ \text{for all } l_i \in \text{comp}(r_{\underline{j}}),$$
$$v^{\dagger}_{j} \leftarrow v^{\dagger}_{j} + \alpha,\ \text{for all } r_j \in \text{comp}(r_{\underline{j}}).$$

Note that each component of $H(u^{\dagger}, v^{\dagger})$ includes identical number of left and right nodes, due to the existence of the perfect matching M. Then, after DUAL_UPDATING the objective value (19) remains unchanged, and (20) and complementary slackness condition (21) continue to be satisfied. Thus, the updated $(u^{\dagger}, v^{\dagger})$ is still optimal to DAP, and the number of arcs in $H(u^{\dagger}, v^{\dagger})$ is increased at least by 1. After repeating DUAL-UPDATING at most $n-1$ times, we obtain a connected $H(u^{\dagger}, v^{\dagger})$.

Here, let T be a *spanning tree* of $H(u^{\dagger}, v^{\dagger})$. Without loss of generality, we assume that the perfect matching M is included in T. Now, the assignment problem AP can be written as

$$\begin{aligned} Bx_B + Nx_n &= 1, \\ c_B x_B + c_N x_N &= z, \end{aligned}$$

where B and N are the columns corresponding to basic and non-basic variables, which are denoted as x_B and x_N respectively. Here we take the variables corresponding to the arcs of T as x_B, and the remaining variables represents x_N. Then, the *incidence matrix* of AP is correspondingly partitioned as (B, N), and the cost vector $(\overline{c}_{ij})$ is written as (c_B, c_N). If we arrange the rows and columns of the tableau in the order of nodes and arcs as encountered in the *breadth-first traverse* [12] of T starting from node l_1, B necessarily becomes an *upper triangular matrix*, which is easily inverted. Thus from the optimal matching obtained by the

Hungarian method we have reconstructed an optimal FCF for AP as

$$\begin{aligned} x_B \quad + B^{-1}N x_N &= B^{-1}\mathbf{1}, \\ (c_N - c_B B^{-1}N)x_N &= z - c_B B^{-1}\mathbf{1}. \end{aligned}$$

Example 4 *For AP^1 with cost matrix $\bar{C}^1$ given in Example 3, by the Hungarian method we obtain the solution depicted in Fig. 1, where $u_i^\dagger$ and $v_j^\dagger$ are shown at each node, and thick lines show the matching M. Here $H(u^\dagger, v^\dagger)$ consists of only one connected component, and a spanning tree is depicted with (thick and thin) solid lines. Also, attached at each node in*

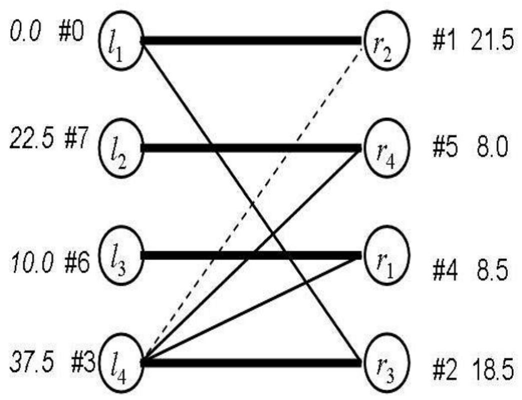

Figure 1: Optimal assignment for $k = 1$.

the figure is the number of steps as encountered in the breadth-first walk [12] from node l_1. Then, rearranging rows and columns in AP^1, the basic and non-basic parts of the problem are given as Tables 2 - 3. The objective line comes from $\bar{C}^1$.

Table 2: Upper triangular matrix B in the initial simplex tableau.

node	x_{12}	x_{13}	x_{43}	x_{41}	x_{44}	x_{31}	x_{24}
r_2	1						
r_3		1	1				
l_4			1	1	1		
r_1				1		1	
r_4					1		1
l_3						1	
l_2							1
min	8.0	8.5	46.0	59.0	56.0	31.5	41.0

Table 3: Non-basic part N of the initial tableau.

x_{11}	x_{14}	x_{21}	x_{22}	x_{23}	x_{32}	x_{33}	x_{34}	x_{42}	const
			1		1			1	1
				1		1			1
								1	1
1		1							1
	1						1		1
					1	1	1		1
		1	1	1					1
48.0	60.0	125.0	48.0	41.0	97.0	58.0	88.0	45.5	0.0

The matrix B of Table 2 can be easily inverted, and we obtain the non-basic part of the optimal simplex tableau as shown in Table 4.

Table 4: Non-basic part $B^{-1}N$ of the optimal FCF.

x_{11}	x_{14}	x_{21}	x_{22}	x_{23}	x_{32}	x_{33}	x_{34}	x_{42}	const
			1		1			1	1
1	1		−1		−1			−1	
−1	−1		1	1	1	1		1	1
1		1			−1	−1	−1		
	1	−1	−1	−1			1		
					1	1	1		1
		1	1	1					1
26.5	41.5	81.0	17.5	10.0	79.0	39.5	59.5	0.0	-126.5

Having an optimal FCF for AP, we can now apply the pegging test (Theorem 4) and fix some variables as follows.

Example 5 *With gap* $= \bar{z}_{RAP} - \underline{z}_{RAP} = 5.5$, *we see all non-basic variables except for* x^1_{42} *are fixed at 0. For basic variables, from (i) of Theorem 4, we fix variables as:* $x^1_{41} = x^1_{44} = 0, x^1_{31} = x^1_{24} = 1$, *and we have only 4 variables unfixed for* $k = 1$. *Similarly, from* $k = 2$ *we have 7 unfixed variables, and removing the fixed parts, we have a BIP with 11 variables and 21 constraints. Solving this, we obtain an optimal solution to RAP with* $z^* = 241$, *which is identical to the value obtained by solving the original RAP (with 50 variables and 45 constraints) directly.*

3.3 An improved reduction method for RAP

A difficulty with the pegging test of section 3.2 is the reconstruction of an optimal FCF for AP. For an assignment problem of size $n \times n$, after obtaining the basis matrix B and its inverse B^{-1}, computing all the elements of $B^{-1}N$ can be quite expensive for problems with large n, since N is a matrix of $(2n-1) \times (n^2 - 2n + 1)$. However, we show here that a very small portion of $B^{-1}N$ suffices to carry out the pegging test, which enables us a drastic speed-up of computation.

Let us consider the optimal FCF given by (12) and (13). From the *unimodularity* [1] of the coefficient matrix (B, N) in the assignment problem we have the following for all i and j.

$$\alpha_{ij} \in \{-1, 0, 1\}, \ \forall i, j \in N, \tag{22}$$
$$\bar{b}_i \in \{0, 1\}, \ \forall i. \tag{23}$$

Let

$$N^+ := \{j \in N \mid \alpha_{0j} > \bar{z} - \underline{z}\}, \ N^- := \{j \in N \mid \alpha_{0j} \le \bar{z} - \underline{z}\}. \tag{24}$$

Then, we have

Theorem 5

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

Proof: (i) From (15), $PL_i = \min\{\alpha_{0j} \mid \alpha_{ij} = 1, j \in N\} = \min\{PL_i^+, PL_i^-\}$, where $PL_i^\pm := \min\{\alpha_{0j} \mid \alpha_{ij} = 1, j \in N^\pm\}$, respectively. By definition of N^+, we have $PL_i^+ > \bar{z} - \underline{z}$ and $PL_i^- \le \bar{z} - \underline{z}$. Then, $PL_i > \bar{z} - \underline{z} \Leftrightarrow \{j \in N^- \mid \alpha_{ij} = 1\} = \emptyset$; hence from Theorem 4, (i) is proved. (ii) is proved analogously. ∎

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 5 is far more faster than the direct application of Theorem 4. For the example of Section 3.2, we only need the column of x_{42} to obtain the same pegging result.

4 Virtual pegging approach

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, since the size of the problem will not be reduced much in such a case. In the present section, we introduce a *virtual* pegging test in order to cope with this difficulty.

4.1 The principle

For the problem Q of Section 3.1, pegging test works with a pair of upper and lower bounds $\bar{z}$ and $\underline{z}$ satisfying

$$\underline{z} \ \le \ z^\star \ \le \ \bar{z}. \tag{25}$$

However, we may perform 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_j's will be fixed either at 0 or 1. But this pegging is not guaranteed to be correct because l is not necessarily a true upper bound. Let the index sets of variables, which are 'fixed' at 0 and 1 by the above procedure, be $F_0(l)$ and $F_1(l)$ respectively. Then, we have the following *reduced problem*.

$$\text{Q}(l): \quad \text{minimize } c^T x \text{ subject to } Ax = b, x_j \in \{0, 1\}, \textit{ and}$$
$$x_j = 1, \text{if } j \in F_1(l),\ x_j = 0, \text{if } j \in F_0(l).$$

The optimal objective value to this problem will be denoted as $z_l^\star$, and is referred to as the *realization* for the trial value l. If Q(l) is infeasible, we put $z_l^\star := \infty$. Then, we have

Theorem 6 [16] *For an arbitrary trial value $l \geq \underline{z}$ and its realization $z_l^\star$, the followings hold.*

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

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

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

4.2 A virtual pegging procedure

To apply the virtual pegging method to RAP, we prepare a parameter $\alpha > 0$, which we call the *virtual gap*, such that $\alpha < gap := \overline{z}_{RAP} - \underline{z}_{RAP}$. We present a procedure which reduces RAP to a small BIP, and by solving it we often obtain an exact solution to the original problem. The procedure is as follows.

Procedure VIRTUAL-PEGGING.

Input: $\overline{z}_{RAP}, \underline{z}_{RAP}$ and optimal FCFs for AP^k ($k = 1, \ldots, K$).

Parameter: Virtual gap α.

Step 1. Apply the pegging test (Theorem 5) to each of AP^k with trial value $\underline{z}_{RAP} + \alpha$.

Step 2. Eliminate the fixed variables from RAP and obtain a reduced BIP.

Step 3. Solve the BIP using a MIP solver, and obtain a feasible solution $x^\sharp$ to RAP with the corresponding objective value $z^\sharp$. Output $(x^\sharp, z^\sharp)$ and stop.

Actually, $z^\sharp$ is the realization for the trial value $\underline{z}_{RAP} + \alpha$. Then, by Theorem 6, if $z^\sharp \leq \underline{z}_{RAP} + \alpha$ the solution is proved optimal. Even if $z^\sharp > \underline{z}_{RAP} + \alpha$, it is often the case that $z^\sharp < \overline{z}_{RAP}$, and then we have a better upper bound than $\overline{z}_{RAP}$. Or, we may retry VIRTUAL-PEGGING with α increased until optimality of the solution is proved.

5 Numerical experiments

This section gives the results of numerical experiments conducted to evaluate the performance of the proposed method. We implemented REPEATED-HUNGARIAN, DELAYED-INCLUSION and VIRTUAL-PEGGING procedures in ANSI C language, and carried out computation on an Dell DIMENSION 8250 computer (CPU: Pentium4, 3.40GHz, 2.00GB RAM). To solve LP and BIP problems, CPLEX 11.1 was used on the same computer.

5.1 Design of experiments

The instances were prepared in the following way. First, a *nominal* cost c_{ij}^0 is assumed to be uniformly random over the integer interval [1,1000], and the cost c_{ij}^k at k-th repetition is determined as a uniformly random integer over [Floor, Ceil], where

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

Here, σ denotes the parameter representing the degree of *correlation* between the costs over $k = 1, 2, \ldots, K$. We tried three cases:

- Uncorrelated (UNCOR): $\sigma = 0.0$.
- Weakly correlated (WEAK): $\sigma = 0.3$.
- Strongly correlated (STRONG): $\sigma = 0.6$.

In Figure 2 we plot (c_{ij}^1, c_{ij}^2) for each of these correlation types.

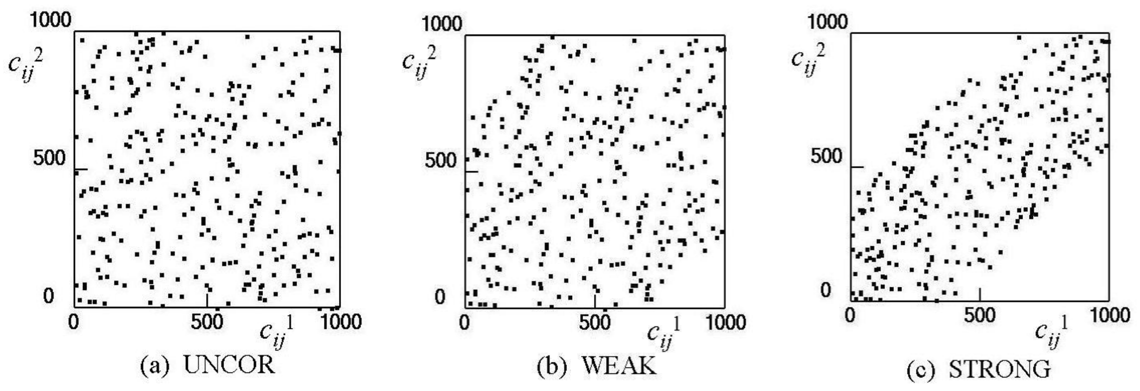

Figure 2: Correlation between (c_{ij}^1, c_{ij}^2).

5.2 A numerical example

For an instance with $n = 120, K = 12$ and $\delta = 0.3$, we obtained bounds as $\bar{z}_{RAP} = 27713$ and $\underline{z}_{RAP} = 27199.7$. Assuming the virtual gap of $\alpha = 5$, 103,560 (out of 173,800) variables were fixed at 0, and removing these we got a BIP with 69240 variables and 17147 constraints. Using CPLEX we solved this and obtained a solution with $z^\sharp = 27205$, which was much better than $\bar{z}_{RAP}$. However, this solution was not proved optimal, since $z^\sharp - \underline{z} > \alpha$. Then, we repeat VIRTUAL-PEGGING with α increased to $\alpha = 10$. We obtained $z^\sharp = 27205$ again, and this was proved to be optimal, since we have $z^\sharp - \underline{z} < \alpha$.

5.3 Result of experiments

Table 5 shows the result of computation of the upper and lower bounds ($\bar{z}$ and $\underline{z}_{RAP}$). Each row is the average over 10 randomly generated instances, and in addition to $\bar{z}$ and $\underline{z}$ the columns stand for the followings: '*#Col*,' '*#Row*' and '*#Cycle*' stand for the maximum number of columns and rows in DELAYED-INCLUSION, and the number of iterations (Steps 2 - 5) repeated therein. 'CPU_1' is the CPU time in seconds to compute $\bar{z}$ and $\underline{z}$, and $gap := \bar{z} - \underline{z}$.

Table 6 summarizes the result of VIRTUAL-PEGGING with $\alpha = 5$ for the same instances as in Table 5. We show here the numbers of variables fixed at 0 and 1 ('$\#Fix_0$' and '$\#Fix_1$', resp.), the size of the reduced BIP ('*#Col*' and '*#Row*', resp.), the optimal objective value ($z^\star$) and the total CPU time (CPU_1 + the time to solve the reduced BIP problem) in seconds. The column of '*#Int*' gives the number of instances (out of 10) where integer solutions were obtained in DELAYED-INCLUSION. In such a case, we put $\#Fix_1 = nK$, $\#Fix_0 = n^2K - nK$, and $\#Col$=$\#Row$=0 in computing the averages. Also, the column of '*#Opt*' gives the number of instances where the solution was proved optimal ($\alpha > z^\sharp - \underline{z}_{RAP}$).

Table 5: Upper and lower bounds

σ	n	K	$\bar{z}_{RAP}$	$\#Row$	$\#Col$	$\#Cycle$	$\underline{z}_{RAP}$	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

Table 6: Reduction and CPLEX

σ	n	K	$\#Int$	$\#Fix_0$	$\#Fix_1$	$\#Col$	$\#Row$	$z^\sharp$	CPU	$\#Opt$
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	10
		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

From these tables, we observe the followings.

1. For $K = 4$, we often obtain an optimal solution by solving C(RAP) via DELAYED-INCLUSION.

2. Although the gap between the upper and lower bounds increases with the problem size (n and K) and the degree of correlation (σ), the actual optimal value ($z^\star$) is usually very close to the lower bound ($\underline{z}_{RAP}$), and thus the virtual pegging is very effective to the instances tested here.

3. With $\alpha = 5$, VIRTUAL-PEGGING reduces problem size to the level which is tractable by MIP solvers such as CPLEX, and by solving the reduced BIP, except for some cases, we obtain optimal solutions within a few minutes. Even if the solution is not proved optimal, we usually have approximate solutions which are much better than the original repeated Hungarian solution.

5.4 Weakness of the pegging approach

VIRTUAL-PEGGING remains some weakness for instances with relatively small n and large K and σ. Table 7 compares VIRTUAL-PEGGING against the direct solution of RAP by CPLEX 11.1. Each row is again the average over 10 random instances, and 'gap' is again $\overline{z}_{RAP} - \underline{z}_{RAP}$. In CPLEX, computation was truncated at the time-limit of 1800 CPU seconds, and in such a case we took the incumbent objective value at that time as $z^\sharp$ and CPU=1800 in deriving averages. The findings from this table is as follows.

Table 7: Solution by CPLEX 11.1

σ	n	K	VIRTUAL-PEGGING				CPLEX	
			Gap	$z^\sharp$	CPU	$\#Opt$	$z^\star$	CPU
0.0	40	4	0.0	6390.7	0.01	10	6390.7	0.10
		8	89.5	13297.8	0.09	6	13294.7	4.52
		12	1164.5	20369.8	8.21	0	20293.2	112.35
	80	4	1.6	6764.5	0.05	9	6764.5	1.37
		8	28.9	13611.5	0.18	10	13611.5	4.08
		12	263.2	20688.0	6.08	7	20687.7	58.88
	120	4	0.0	6805.5	0.10	10	6805.5	5.78
		8	6.5	13796.5	0.32	10	13796.5	12.92
		12	106.2	20847.7	4.40	10	20847.7	54.90
0.3	40	4	0.0	8013.8	0.01	10	8013.8	0.10
		8	441.1	16715.9	0.13	4	16604.0	17.43
		12	1961.4	26778.1	3.33	0	25773.8	297.44
	80	4	0.0	8618.0	0.05	10	8618.0	1.01
		8	128.2	17372.2	2.51	6	17371.9	36.70
		12	747.9	26642.1	60.84	0	26635.2	301.27
	120	4	2.3	8781.3	0.12	10	8781.3	7.28
		8	69.9	17855.7	0.56	10	17855.7	23.39
		12	316.9	27021.1	70.54	1	27021.0	190.37
0.6	40	4	60.3	8679.1	0.03	8	8675.9	0.31
		8	1696.5	19728.3	0.30	0	18623.3	165.33
		12	6152.6	35406.5	2.07	0	31245.6	1808.62
	80	4	20.3	9138.0	0.07	10	9138.0	1.32
		8	465.1	18888.1	28.76	0	18872.8	204.40
		12	1850.2	30141.2	1090.83	0	29831.4	1816.65
	120	4	0.0	9503.2	0.12	10	9503.2	6.46
		8	208.4	19516.9	32.21	4	19516.6	253.80
		12	998.3	29983.2	1107.43	0	29976.9	1766.79

The findings from this table is as follows.

1. In these instances with $n \leq 120$, the gap between the optimal objective value and the lower bound ($z^\star - \underline{z}_{RAP}$) is often larger than $\alpha(= 5)$, and thus, VIRTUAL-PEGGING fails to produce a solution with optimality proved. If we retry VIRTUAL-PEGGING with an increased α, we may obtain optimal solutions.

2. Even in such a case, we obtain feasible solutions with $z^\sharp$ close to the optimal in shorter time than the direct application of CPLEX.

6 Conclusion

We formulated the repeated assignment problem, and presented an approach to solve this problem. The proposed methods include REPEATED-HUNGARIAN, DELAYED-INCLUSION, and VIRTUAL-PEGGING. Through numerical experiments, we found that the developed method was able to reduce the size of the problem, often significantly, and the reduced BIP can be solved using commercial or free MIP solvers. As the result, we were able to solve RAP with up to $n = 600$ and $K = 12$, often with the proof of optimality. The method remains some weakness for problems with relatively small n and large K, where direct application of MIP solvers can be competitive solution strategy.

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

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

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

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

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

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

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

[14] 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.

[15] 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.

[16] 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.