The Sixth International Symposium on Operations Research and Its Applications (ISORA'06)
Xinjiang, China, August 8–12, 2006
, pp. 118–127

# An Improved Algorithm for the Bilateral Assignment Problem

Veerayuth Nomsiri* Takeo Yamada†

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

**Abstract** We are concerned with a variation of the assignment problem, where the assignment costs differ between assigners and assignees. We formulate this as a min-max optimization problem, present a surrogate relaxation approach to derive lower and upper bounds quickly, and show that the pegging test can be successfully applied to this problem. Next, we make use of the special structure of the assignment problem to shorten the computation time for pegging significantly. Finally, through numerical experiments we show that the developed method solves larger instances to optimality faster than conventional methods.

**Keywords** bilateral assignment problem; mini-max optimization; pegging test

## 1 Introduction

The *assignment problem* is a classical combinatorial optimization problem that can be solved efficiently by polynomial time algorithms [7]. Here we assign, for example, $n$ workers to $n$ jobs and want to minimize the total *cost* of assignment. In this article, we are concerned with a variation of this problem, where the perceived cost of assignment differs between workers and job managers. Let $p^1_{ij}$ and $p^2_{ij}$ denote, respectively, the worker's and manager's cost of assigning worker $i$ to job $j$, and $x_{ij}$ is the decision variable that takes value 1 in this assignment and 0 otherwise. Then, for $k = 1, 2$

$$z^k(x) := \sum_{i=1}^{n} \sum_{j=1}^{n} p^k_{ij} x_{ij} \tag{1}$$

is the workers' (resp., managers') total cost of assignment $x = (x_{ij})$, and thus we have two objective functions to minimize. Such a bi-objective assignment problem has been investigated under the framework of *stable marriage problem* [5] or *assigmnent game* [10].

*Currently with Chulachomklao Royal Military Academy, Thailand
†Corresponding author. E-mail: yamada@nda.ac.jp

An alternative approach is the *mini-max* optimization, where we minimize the maximum of $z^k(x),\ k=1,2$. Thus, we formulate the *bilateral assignment problem* as

**BAP:**

$$\text{minimize} \qquad z(x) := \max\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^1 x_{ij}, \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^2 x_{ij}\right\} \tag{2}$$

$$\text{subject to} \qquad \sum_{j=1}^{n} x_{ij} = 1,\ i=1,\ldots,n, \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j=1,\ldots,n, \tag{4}$$

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

Rewriting this as the following linear 0-1 programming problem, we may solve small instances using free or commercial IP solvers.

**BAP$^\flat$:**

$$\text{minimize} \qquad v \tag{6}$$

$$\text{subject to} \qquad \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^k x_{ij} \le v,\ k=1,2, \tag{7}$$

$$(3),\ (4),\ (5).$$

BAP was originally formulated by Kouvelis and Yu [6] as a *robust* optimization problem with $K \ge 2$ objective functions. They also proved that the problem is $\mathscr{NP}$-hard, constructed a branch-and-boud algorithm based on a surrogate relaxation method, and solved some test problems with $n \le 40$ and $K \le 30$. A heuristic algorithm for BAP was given by Sakakibara and Nakamori [9], where problems with $n \le 200$ were solved approximately.

In this paper, we also take the surrogate relaxation approach to derive lower and upper bounds quickly. Then, making use of these bounds we apply the *pegging test* [8] to reduce the size of the problem. For BAP we can further improve the pegging test and reduce the computing time significantly. Combining these, we are often able to solve problems with $n \le 1000$ exactly in reasonable CPU time.

## 2 Lower and upper bounds

This section derives an *upper bound* by applying the *surrogate relaxation* [4] to BAP. At the same time, we obtain an approximate solution and thus a *lower bound* to BAP.

## 2.1 Surrogate relaxation

Let $\lambda$ be an arbitrary value satisfying $0 \leq \lambda \leq 1$. By replacing (7) with a single constraint

$$\lambda \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}^1 x_{ij} + (1-\lambda) \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}^2 x_{ij} \leq v$$

and eliminating $v$ in BAP$^p$ we obtain the *surrogate relaxation problem* [4]

**SRP**($\lambda$):

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} \bar{p}_{ij}(\lambda) x_{ij} \\ \text{subject to} & (3),\ (4),\ (5), \end{array} \tag{8}$$

where

$$\bar{p}_{ij}(\lambda) := \lambda p_{ij}^1 + (1-\lambda) p_{ij}^2. \tag{9}$$

Note that for a fixed $\lambda \in [0,1]$, SRP($\lambda$) is a standard assignment problem which can be efficiently solved by, e.g., the *Hungarian method* [7]. By $\underline{x}(\lambda) = (\underline{x}_{ij}(\lambda))$ we denote an optimal solution to SRP($\lambda$) with the corresponding objective value $\underline{z}(\lambda) := z(\underline{x}(\lambda))$. Then, analogous to the case of Lagrangian relaxation [2], the followings can be easily shown.

**Proposition 1.**

(i) *For an arbitrary* $\lambda \in [0,1]$, $\underline{z}(\lambda)$ *gives a* lower bound *to* BAP*, i.e., the optimal objective value* $z^\star$ *to* BAP *satisfies*

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

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

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

$$d\underline{z}(\lambda)/d\lambda = z^1(\underline{x}(\lambda)) - z^2(\underline{x}(\lambda)). \tag{10}$$

To find a lower bound with $\underline{z}(\lambda)$ as large as possible, we solve the following *surrogate dual problem* [4]

**SDP**:

$$\begin{array}{ll} \text{maximize} & \underline{z}(\lambda) \\ \text{subject to} & \lambda \in [0,1]. \end{array}$$

Let the solution to this problem be $\lambda^\dagger$ with the corresponding *optimal* lower bound $\underline{z} := \underline{z}(\lambda\dagger)$. Using (10) this can be easily obtained by the standard *bisection method.*

### 2.2 Upper bound

Solving SRP($\lambda$) at an arbitrary $\lambda \in [0,1]$ to obtain the lower bound $\underline{z}(\lambda)$, we get an *upper bound*

$$\bar{z}(\lambda) := \max\{z^1(\underline{x}(\lambda)), z^2(\underline{x}(\lambda))\} \tag{11}$$

simultaneously, since the solution $\underline{x}(\lambda)$ is also feasible to BAP. The smallest $\bar{z}(\lambda)$ encountered in the bisection process gives the *best* upper bound $\bar{z}$, and together with $\underline{z}$ these are fed to the *reduction* algorithm of Section 3.

## 3 Reduction of BAP

### 3.1 Pegging test for 0-1 programming problem

Here we briefly summarize some basic results on *pegging test* [8] for readers' convenience. For simplicity of notation, let us consider the following 0-1 programming problem.

**P:**

$$\text{minimize} \qquad z(x) := \sum_{j=1}^{s} c_j x_j \tag{12}$$

$$\text{subject to} \qquad \sum_{j=1}^{s} a_{ij} x_j = b_j,\ i = 1,\ldots,r, \tag{13}$$

$$x_j \in \{0,1\},\ j = 1,\ldots,s. \tag{14}$$

Let $x^\star = (x_j^\star) \in R^s$ be an optimal solution to P with the objective value $z^\star := z(x^\star)$. First, we relax P by replacing (14) with

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

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 $\bar{x} \in R^s$ to P. This gives an upper bound $\bar{z} := z(\bar{x})$. Thus we have

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

Let an optimal *feasible canonical form* (FCF) of C(P) be

$$\bar{b}_i = x_{B(i)} + \sum_{j \in N} \alpha_{ij} x_j, \tag{15}$$

$$z = \underline{z} + \sum_{j \in N} \alpha_{0j} x_j, \tag{16}$$

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

$$\alpha_{0j} \ge 0, \qquad \forall j \in N, \tag{17}$$

$$0 \le \bar{b}_i \le 1, \qquad i = 1,\cdots,r. \tag{18}$$

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

$$\begin{aligned} PU_i &:= \min\{-\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} < 0\}(1-\bar{b}_i), \\ PL_i &:= \min\{\alpha_{0j}/\alpha_{ij} \mid j \in N, \alpha_{ij} > 0\}\bar{b}_i. \end{aligned}$$

Here, if the defining set is empty, we set $\min\{\cdot \mid \emptyset\} := \infty$. Then, we have [8]

**Proposition 2.**

(i) *For basic variable* $x_{B(i)}$ $(i = 1, \ldots, r)$ *in* (15),

$$\begin{aligned} PU_i > \bar{z} - \underline{z} &\Rightarrow x^\star_{B(i)} = 0, \\ PL_i > \bar{z} - \underline{z} &\Rightarrow x^\star_{B(i)} = 1. \end{aligned}$$

(ii) *For non-basic variable* $x_j$ $(j \in N)$ *in* (16),

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

## 3.2 Pegging test for BAP

Since SRP($\lambda^\dagger$) is an assignment problem, without loss of generality, we can relax the 0-1 constraint (5) to non-negativity requirement, and thus the problem can be regarded as a *linear programming* (LP) problem. Then, although SRP($\lambda^\dagger$) is not just a continuous relaxation of the original BAP as opposed to C(P) (it is actually a surrogate relaxation *plus* continuous relaxation of BAP), Proposition 2 is still valid, and we can make use of this to reduce the problem size, provided that we have an optimal FCF for SRP($\lambda^\dagger$). However, in solving SRP($\lambda^\dagger$) we usually use such efficient algorithm as the Hungarian method, because it is much faster than LP algorithms such as the *revised simplex method*. Then, to apply Proposition 2, we need to *reconstruct* an optimal FCF associated with the optimal solution to SRP($\lambda^\dagger$). This can be accomplished in the following way.

In the Hungarian method we solve the following *dual* of SRP($\lambda^\dagger$), where for simplicity we write $\bar{p}_{ij} := \bar{p}_{ij}(\lambda^\dagger)$.

**DSR($\lambda^\dagger$):**

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

$$\text{subject to} \qquad u_i + v_j \le \bar{p}_{ij}, \ \forall i, j. \tag{20}$$

For an arbitrary feasible solution $(u, v)$ to DSR($\lambda^\dagger$), $H(u, v)$ denotes the undirected *bipartite graph* 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 $A(u, v) = \{(l_i, r_j) \in L \times R \mid u_i + v_j = \bar{p}_{ij}\}$. By the Hungarian method, we obtain an optimal solution $x^\dagger = \underline{x}(\lambda^\dagger)$ to SRP($\lambda^\dagger$), as well as an optimal solution $(u^\dagger, u^\dagger)$ to DSR($\lambda^\dagger$). These satisfy the following *complementary slackness condition*

$$x^\dagger_{ij} = 1 \Rightarrow u^\dagger_i + v^\dagger_j = \bar{p}_{ij}, \tag{21}$$

and include a *perfect matching* $M = \{(l_i, r_j) \in L \times R \mid x^\dagger_{ij} = 1\}$ in $H(u^\dagger, v^\dagger)$.

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.

DUAL_UPDATING

a) Decompose $H(u^\dagger, v^\dagger)$ into connected components.
b) Find $\alpha := \min\{\bar{p}_{ij} - u^\dagger_i - v^\dagger_j \mid (l_i, r_j) \in L \times R,\ \mathrm{comp}(l_i) \neq \mathrm{comp}(r_j)\}$, and let $(l_{\underline{i}}, r_{\underline{j}})$ be a pair where the minimum is attained.
c) Modify $(u^\dagger, v^\dagger)$ according to:

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

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) are kept satisfied. Thus, the updated $(u^\dagger, v^\dagger)$ is still optimal to DSR$(\lambda^\dagger)$, 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, consider the *simplex tableau* corresponding to SRP$(\lambda^\dagger)$ which can be written as

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

We can take the set of optimal *basic variables* as those corresponding to the arcs of $T$, and $x_B$ and $x_N$ represent, respectively, those basic and non-basic variables. Here the *incidence matrix* of SRP$(\lambda^\dagger)$ is correspondingly partitioned as $(B, N)$, and the cost vector $(\bar{p}_{ij})$ 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* 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 SRP$(\lambda^\dagger)$ as

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

### 3.3 An improved reduction method for BAP

A difficulty with the pegging test of section 3.2 is the reconstruction of an optimal FCF for SPR($\lambda^\dagger$). For the problem of size $n$, after obtaining the basis matrix $B$ and its inverse $B^{-1}$, computing all the elements of $B^{-1}N$ is almost prohibitive 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.

For simplicity, let us consider the optimal FCF given in (15) and (16). In addition to (17), from the *unimodularity* of the coefficient matrix $(B,N)$ in the assignment problem SRP($\lambda^\dagger$) we have the following for $i = 1,2,\ldots,r$.

$$\alpha_{ij} \in \{-1,0,1\},\ \forall j \in N, \tag{22}$$

$$\bar{b}_i \in \{0,1\}. \tag{23}$$

Let

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

Then, we have

**Theorem 3.**

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

**Proof.** (i) From (22), $PL_i = \min\{\alpha_{0j} \mid \alpha_{ij} = 1, j \in N\} = \min\{PL_i^+, PL_i^-\}$, 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 Proposition 2, (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 1 is far more faster than the direct application of Proposition 2.

## 4 Numerical Experiments

### 4.1 Design of experiments

For BAP with $n = 200 \sim 1000$, we evaluate the performance of the 'surrogate relaxation + pegging' approach stated in previous sections. We follow [6] in preparing test problems as follows. First, *nominal* assignment cost $p^0_{ij}$ is determined as a uniform random integer over [1,1000]. Then, the assignment cost is

$$p^k_{ij} : \text{uniformly random integer over } [(1-\delta)p^0_{ij}, (1+\delta)p^0_{ij}],$$

for $k = 1, 2$, where $\delta$ is a parameter to control the degree of *correlation* between different scenarios. Note that assignment costs are more strongly correlated for smaller $\delta$, and we examine the cases of $\delta = 0.3,\ 0.6$ and 0.9.

To compute upper and lower bounds as well as for doing pegging test, we implemented the algorithm in ANSI C language on an IBM RS/6000 SP 44 Model 270 workstation (CPU: POWER 3-II, 375Mhz), and to solve the reduced problem XPRESS-IVE Ver. 1.13.30 [1] was run on an DELL 8400 computer (Pentium (R), 3.40GHz).

## 4.2 Bounds and reduction

Table 1 gives the results of computation of the upper and lower bounds, as well as that of pegging test. In addition to the bounds $\underline{z}$ and $\bar{z}$, the table includes

gap : $= \bar{z} - \underline{z}$,
assgn : the number of the Hungarian methods solved to obtain the bounds,
rerr : relative error given by $100 \cdot (\bar{z} - \underline{z})/\underline{z}\ (\%)$,
$\text{fix}_1$ : the number of decision variables fixed at 1,
$n'$ : the number of unfixed variables (out of $n^2$ variables),
reduc : the ratio of unfixed variables defined as $100 \cdot n'/n^2\ (\%)$,
$\text{CPU}_1$ : the time in seconds to evaluate the bounds,
$\text{CPU}_2$ : the time to reconstruct an optimal FCF and carry out the pegging test.

Each row is the average over 10 randomly generated instances.

Table 1: Lower and upper bounds with pegging test.

| $\delta$ | $n$ | $\underline{z}$ | $\bar{z}$ | gap | assgn | rerr | $\text{fix}_1$ | $n'$ | reduc | $\text{CPU}_1$ | $\text{CPU}_2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.3 | 200 | 1459.1 | 1464.0 | 4.9 | 6.1 | 0.34 | 89.8 | 355.8 | 0.89 | 0.1 | 0.2 |
| | 400 | 1408.1 | 1411.5 | 3.4 | 5.7 | 0.24 | 140.5 | 954.0 | 0.60 | 0.5 | 0.6 |
| | 600 | 1328.4 | 1330.6 | 2.2 | 5.9 | 0.17 | 167.3 | 1427.8 | 0.40 | 1.4 | 2.1 |
| | 800 | 1236.1 | 1237.9 | 1.8 | 5.2 | 0.15 | 327.9 | 1806.6 | 0.28 | 2.4 | 4.2 |
| | 1000 | 1132.9 | 1135.3 | 2.4 | 5.4 | 0.21 | 275.4 | 3627.9 | 0.36 | 4.0 | 10.2 |
| 0.6 | 200 | 1384.2 | 1393.2 | 9.0 | 6.1 | 0.65 | 63.2 | 581.8 | 1.45 | 0.1 | 0.2 |
| | 400 | 1329.1 | 1333.6 | 4.5 | 7.2 | 0.34 | 92.2 | 1221.0 | 0.76 | 0.6 | 0.9 |
| | 600 | 1252.9 | 1255.5 | 2.6 | 6.5 | 0.21 | 301.6 | 1381.9 | 0.38 | 1.6 | 0.9 |
| | 800 | 1146.3 | 1149.6 | 3.3 | 6.7 | 0.29 | 213.6 | 3047.2 | 0.48 | 2.9 | 6.6 |
| | 1000 | 1036.5 | 1039.8 | 3.3 | 6.2 | 0.32 | 332.8 | 4546.5 | 0.45 | 4.7 | 5.2 |
| 0.9 | 200 | 1195.8 | 1204.9 | 9.1 | 7.5 | 0.76 | 47.8 | 679.4 | 1.70 | 0.1 | 0.2 |
| | 400 | 1100.5 | 1103.7 | 3.2 | 7.2 | 0.29 | 112.6 | 1047.9 | 0.65 | 0.6 | 0.8 |
| | 600 | 989.7 | 999.4 | 9.7 | 6.9 | 0.98 | 48.5 | 5207.0 | 1.47 | 1.6 | 2.5 |
| | 800 | 881.9 | 887.0 | 5.1 | 6.5 | 0.58 | 97.2 | 5099.3 | 0.80 | 2.9 | 6.0 |
| | 1000 | 789.0 | 793.3 | 4.3 | 7.4 | 0.55 | 229.1 | 6445.3 | 0.64 | 5.4 | 10.2 |

From these tables, we observe the following.

1. By the surrogate relaxation method we get a heuristic solution and a lower bound in reasonably small CPU time. The relative error between these bounds is usually less than 1.0%.
2. By the pegging test, problem is reduced remarkably in size. The reduced problem is usually less than 2% in size of the original.
3. CPU time to compute lower and upper bounds ($\mathrm{CPU}_1$) increases with $n$, but this is rather insensitive to $\delta$.
4. CPU time for the reconstruction of optimal FCF and for pegging test ($\mathrm{CPU}_2$) increases with $n$, but increase of this with $\delta$ is relatively small.
5. As $n$ increases, reconstruction of optimal FCF grows to be a dominant part of CPU time.

### 4.3 Exact solution

Table 2 summerizes the result of computation of exact solutions. We compare here (i) the direct solution of BAP$^{\flat}$ using XPRESS-IVE, and (ii) application of the pegging test based on Theorem 1, followed by XPRESS-IVE to solve the reduced problem. The latter is henceforth referred to as the PEG-METHOD. For each values of $\delta$ and $n$, we computed the same 10 instances as in section 4.2, and the average of these are shown in these tables. We truncated XPRESS-IVE at the time limit of 600 seconds. All the rows, except for those marked $(-)$, are the average of 10 runs which are successfully completed within this time limit. In all cases, whenever solved by both methods, we obtained the identical objective value $z^{\star}$. In these tables, 'BBN' is the number of the branch-and-bound nodes generated by XPRESS-IVE, $\mathrm{CPU}_3$ is the time to solve the reduced problem (on a DELL computer), and $\mathrm{CPU}_T$ is the total time needed to solve problem exactly by our method, i.e., $\mathrm{CPU}_T = \mathrm{CPU}_1 + \mathrm{CPU}_2 + \mathrm{CPU}_3$.

From these tables, we conclude that PEG-METHOD is able to solve larger instances in considerably smaller CPU time. However, increase of parameter $\delta$ makes problem harder to solve.

## 5 Conclusion

We have introduced the surrogate relaxation method to BAP, and showed that this gives a lower bound as well as a heuristic solution of high accuracy in relatively small CPU time. Also, we have shown that the pegging test is effectively applied to reduce the size of the problem. In numerical experiments, we were able to solve larger problems in much smaller computation time than the direct application of a commercial IP solver.

## References

[1] Dash Associates, XPRESS-IVE, Ver. 1.13.30, 2002, `http://www.dashoptimization.com`.

[2] M. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 50, 1861–1871, 2004.

Table 2: Exact solutions.

| $\delta$ | $n$ | $z^\star$ | XPRESS-IVE | | PEG-METHOD | | |
|---|---|---|---|---|---|---|---|
| | | | BBN | CPU | BBN | $CPU_3$ | $CPU_T$ |
| 0.3 | 200 | 1460.5 | 29.9 | 1.72 | 5.8 | 0.0 | 0.3 |
| | 400 | 1409.2 | 36.4 | 16.16 | 4.5 | 0.0 | 1.1 |
| | 600 | 1329.1 | 21.0 | 31.94 | 12.4 | 0.1 | 3.6 |
| | 800 | 1236.5 | 20.7 | 77.92 | 1.6 | 0.1 | 6.7 |
| | 1000 | 1133.3 | 12.1 | 99.24 | 1.6 | 0.2 | 14.3 |
| 0.6 | 200 | 1385.8 | 49.3 | 3.63 | 44.0 | 0.2 | 0.5 |
| | 400 | 1330.1 | 62.8 | 29.23 | 41.7 | 0.3 | 1.9 |
| | 600 | 1253.5 | 48.7 | 54.66 | 11.7 | 3.0 | 0.4 |
| | 800 | 1146.8 | 29.3 | 82.10 | 27.5 | 1.6 | 11.0 |
| | 1000 | 1037.0 | – | | 25.9 | 4.4 | 14.3 |
| 0.9 | 200 | 1197.9 | 71.0 | 5.38 | 58.3 | 0.3 | 0.5 |
| | 400 | 1101.3 | 39.1 | 22.33 | 27.4 | 0.2 | 1.6 |
| | 600 | 990.3 | – | | 56.9 | 11.6 | 15.7 |
| | 800 | 882.3 | – | | 45.2 | 12.8 | 21.7 |
| | 1000 | 789.6 | – | | 18.9 | 3.2 | 18.8 |

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

[4] F. Glover. Surrogate constraint duality in mathematical programming. *Operations Research*, 23, 434–451, 1975.

[5] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*, The MIT Press, Cambridge, MA, 1989.

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

[7] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2, 83–97, 1955.

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

[9] S. Sakakibara and M. Nakamori. On assignment problems with vector costs. In *Proc. 2005 Fall Conf. OR Soc. Japan*, 2-D-5, 2005. (in Japanese)

[10] G. Shapley and M. Shubik. The assignment game I: the core. *Int. J. Game Theory*, 1, 111–130, 1972.