A Computational Study of Global Algorithms for Linear Bilevel Programming

Carlos Henrique Medeiros de Sabóia *
COPPE/Sistemas, Universidade Federal do Rio de Janeiro - UFRJ.

Manoel Campêlo †
Dep. Estatística e Mat. Aplicada, Universidade Federal do Ceará - UFC,
CEP 60455-760, Fortaleza-CE, Brazil, mcampelo@ufc.br.

Susana Scheimberg ‡
DCC/IM, COPPE/Sistemas, Universidade Federal do Rio de Janeiro - UFRJ,
C.P. 68511, CEP 21945-970, Rio de Janeiro-RJ, Brazil, susana@cos.ufrj.br.

Abstract

In this paper we consider a linear bilevel program (LBP). We analyse two global algorithms that solve the problem. The first one is a recent algorithm built on a new concept of equilibrium point and a modified version of the outer approximation method. The second one is an efficient branch-and-bound algorithm known in the literature. Based on computational results we propose some modifications in both algorithms in order to improve their computational performance. A significant number of experiments is carried out and a comparative study with the algorithms is presented. The modified procedures show a better performance than the original versions.

Keywords: bilevel linear programming, global optimization, branch-and-bound, exact penalty methods.

1 Introduction

Bilevel programming problems arise when a two-level hierarchical system is modeled. In this system there is a leader that makes the decision first and a follower that makes his decision based on the leader's. This kind of problem has a wide field of applications including structural design [16], urban transportation [11], traffic assignment [22], pollution control [2], economics [14], etc. In this article the linear bilevel program is studied. It can be formulated as follows:

$$
\begin{array}{llllll}
\text{(LBP)} & \max_{x,y} & c_1^T x + c_2^T y & & & (1)\\
& \text{s.t.} & & & & \\
& & B_1 x + B_2 y \le b, & & & (2)\\
& & x \ge 0, y \text{ solves:} & & & (3)\\
& & & \max_y & d^T y & (4)\\
& & & \text{s.t.} & A_1 x + A_2 y \le a, & (5)\\
& & & & y \ge 0, & (6)
\end{array}
$$

*Partially supported by CAPES at COPPE/UFRJ.
†Partially supported by CNPq, grant 300251/00-9(RE).
‡Partially supported by CNPq, grant 302393/85-4(RN).

where $c_1, x \in \mathcal{R}^{n_1}$, $c_2, d, y \in \mathcal{R}^{n_2}$, $a \in \mathcal{R}^{m_2}$, $b \in \mathcal{R}^{m_1}$, $B_1 \in \mathcal{R}^{m_1 \times n_1}$, $B_2 \in \mathcal{R}^{m_1 \times n_2}$, $A_1 \in \mathcal{R}^{m_2 \times n_1}$ and $A_2 \in \mathcal{R}^{m_2 \times n_2}$. Programs (1)-(3) and (4)-(6) are called the leader's and the follower's problems where as x and y are the leader's and the follower's variables respectively.

Problem (LBP) has attracted much attention in the last years which may be confirmed by looking at a broad of algorithms that have been proposed to solve it. Hansen *et al.* [12] suggest a nonexclusive classification of the existing algorithms according to the methods used. The main groups and some of their representatives are:

- Extreme point methods [6, 9];
- Linear complementarity problems [10, 15];
- Branch-and-bound [3, 12];
- Penalty function methods [1, 21];
- Reverse convex programming [18, 19].

We recall that (LBP) is a nonconvex problem, since its feasible set may be nonconvex. Actually, this set is composed by faces of the polyhedron given by (2)-(3) and (5)-(6). Hence, problem (LBP) may have local solutions. Also, it can be shown to be a strongly NP-Hard problem [12]. Such characteristics are proper even if constraints (2) do not exist. This is the most studied linear bilevel problem which will be denoted by (LBPP).

The text is organized as follows. In section 2 we describe the algorithms given in [9] and [12]. In section 3 we present two modified methods. In the first one we replace the outer approximation procedure considered in [9] by an enumerative method. In the second one we modify the branch-and-bound algorithm given in [12] by embedding the equilibrium point procedure on it. In section 4 we give computational results for these two modified algorithms and compare their performance with the original branch-and-bound algorithm. Finally, in section 5 we have the conclusions.

2 Preliminaries

In this section we consider a recent penalty algorithm developed by Campêlo and Scheimberg [9] and a branch-and-bound algorithm proposed by Hansen *et al.* [12]. The first one seeks feasible extreme points using a penalty approach. The second one applies necessary optimality conditions expressed in terms of tightness of the follower's constraints to develop a branch-and-bound strategy.

2.1 The penalty algorithm

This algorithm solves problem (LBPP). One of its characteristics is that the assumption of the set $\{(x, y) \in R^{n_1 \times n_2} : A_1x + A_2y \leq a, x \geq 0, y \geq 0\}$ to be non empty and bounded is not required. This is a restrictive assumption usually considered in the literature. We remark that the algorithm works on problems with optimal solution but where the mentioned set may be unbounded. Also, it is able to identify whether problem (LBPP) is infeasible or unbounded. A brief description of the algorithm is provided below.

Let us consider two auxiliary problems of (LBPP):

$$
\begin{array}{lllll}
(P) & & \qquad & P(M) & \\
\max & c_1^T x + c_2^T y & & \max & c_1^T x + c_2^T y - M(u^T w + v^T y) \\
\text{s.t.} & A_1 x + A_2 y + w = a, & & \text{s.t.} & A_1 x + A_2 y + w = a, \\
 & x \geq 0, y \geq 0, w \geq 0, & & & x \geq 0, y \geq 0, w \geq 0, \\
 & A_2^T u - v = d, & & & A_2^T u - v = d, \\
 & u \geq 0, v \geq 0, & & & u \geq 0, v \geq 0, \\
 & u^T w + v^T y = 0, & & &
\end{array}
$$

where $w \in \mathcal{R}^{m_2}$, $u \in \mathcal{R}^{m_2}$ and $v \in \mathcal{R}^{n_2}$.

The first problem (P) is obtained by replacing the follower's problem by its KKT conditions. The second one $P(M)$ is a penalized problem that is obtained from (P) by moving its complementarity constraint into the leader's objective function with a penalty parameter $M \geq 0$. It is worth noting that problem $P(M)$ is attractive from a computational point of view, due to the fact that all of its constraints are linear.

Let us use a compact notation for these two problems. Two block matrices $A = [A_1\, A_2\, I_{m_2}] \in \mathcal{R}^{m_2 \times n}$ and $D = [0\ -I_{n_2}\ A_2^T] \in \mathcal{R}^{n_2 \times n}$ and the vector $c^T = (c_1^T, c_2^T, 0) \in \mathcal{R}^n$ are considered, where $n = n_1 + n_2 + m_2$, I_k is the $(k \times k)$-identity matrix and 0 is a null matrix of appropriate dimension for each case. Also, primal and dual feasible sets are defined as $Z = \{z \in \mathcal{R}^n : Az = a, z^T = (x^T, y^T, w^T) \geq 0\}$ and $S = \{s \in \mathcal{R}^n : Ds = d, s^T = (0, v^T, u^T) \geq 0\}$. Therefore, the auxiliary problems (P) and $P(M)$ may be rewritten as:

$$
\begin{array}{llllll}
(P) & \max & F(z,s) = c^T z & \qquad P(M) & \max & F_M(z,s) = c^T z - M s^T z \\
 & \text{s.t.} & z \in Z, s \in S, & & \text{s.t.} & z \in Z, s \in S. \\
 & & s^T z = 0, & & &
\end{array}
$$

Campêlo and Scheimberg [9, Theorems 5 and 6] establish the following equivalence between the original problem (LBPP) and the auxiliary problems (P) and $P(M)$:

Theorem 1 *Problems (LBPP), (P) and $P(M)$ are infeasible or unbounded for all $M \geq 0$, or else they will have solutions for some M_0. In this case, z^* is a global solution of (LBPP) if, and* *only if, there is s^* such that (z^*, s^*) is a global solution of (P) and $P(M)$ for all $M > M^*$, for some $M^* > M_0$.*

Next we enunciate the concept of equilibrium point (EqP) of the penalized problem given in [9, Definition 7].

Definition 2 *A point $(\hat{z}, \hat{s}) \in Z \times S$ is an equilibrium point of the penalized problem if there is $M_* \geq 0$ such that, for each $M \geq M_*$, it holds:*

$$\max\{F_M(\hat{z}, s) : s \in S\} = F_M(\hat{z}, \hat{s}) = \max\{F_M(z, \hat{s}) : z \in Z\}.$$

We can show that an (EqP) $(\hat{z}, \hat{s})$ satisfies $\hat{s}^T \hat{z} = 0$ and so it is feasible to (P) [8, Proposition 3.3.1]. Therefore, it is a local solution of (P) [9, Theorem 9]. Moreover, we remark that, among all points $z \in Z$ satisfying $\hat{s}^T z = 0$, $\hat{z}$ is the one that provides the maximum value of the leader's objective function. In other words $\hat{z}$ is the best feasible point of (LBPP) in the face $\{z \in Z : \hat{s}^T z = 0\}$.

In [8] Campêlo introduces an algorithm to find an (EqP) or to conclude that (LBPP) is unbounded by solving two linear programs. Assuming that $S \neq \emptyset$ and given an initial primal feasible point $z^0 \in Z$, it works in the following steps:

Basic Equilibrium Point Algorithm **(BEPA)**

Step 1: Solve the problem $\max\{F_M(z^0, s) : s \in S\}$ by the simplex method, obtain a solution $\hat{s}$;

Step 2: Attempt to solve the problem $\max\{F_M(z, \hat{s}) : z \in Z\}$ by the big-M simplex method. If this parametric problem is unbounded, then $P(M)$ is unbounded for all $M \geq 0$. In this case, (LPPP) is unbounded. Otherwise, obtain a solution $\hat{z}$. The point $(\hat{z}, \hat{s})$ is an (EqP).

A proof of the correctness of algorithm (BEPA) is given below:

Proposition 3 *Algorithm (BEPA) verifies that (LBPP) is unbounded or finds an (EqP).*

Proof. The case of unboundedness follows by Theorem 1. Now, let $(\hat{z}, \hat{s})$ be the point obtained by (BEPA). By step 1, it readly follows that $\hat{s} \in \arg\min\{(z^0)^T s : s \in S\}$. By step 2, $\hat{z}$ is a solution of the parametric linear program $\max\{F_M(z, \hat{s}) : z \in Z\}$ for dominating values of M, that is there is $M_* \geq 0$ such that $\max\{F_M(z, \hat{s}) : z \in Z\} = F_M(\hat{z}, \hat{s})$ for all $M \geq M_*$. Then, we can still deduce that $\hat{z} \in \arg\min\{\hat{s}^T z : z \in Z\}$. By Proposition 1 given in [9], we have that $\hat{s}^T \hat{z} = 0$. Thus, $\max\{F_M(\hat{z}, s) : s \in S\} = F_M(\hat{z}, \hat{s})$ for all $M \geq 0$. Hence, by definition 2, $(\hat{z}, \hat{s})$ is an equilibrium point. ■

Procedure (BEPA) is used within the penalty algorithm to find, at the first time, an initial (EqP) $(\hat{z}, \hat{s})$ which is a local solution of (P) and may give a local solution $\hat{z}$ of (LBPP) [8, Theorem

3.3.3]. At this point, a cut is added so that set Z is updated to $\tilde{Z} = Z \cap \{z \in \mathcal{R}^n : c^T z \geq c^T \hat{z} + \varepsilon\}$, where $\varepsilon > 0$. So, problem $P(M)$ is replaced by $\tilde{P}(M) \equiv \max\{c^T z - M s^T z : z \in \tilde{Z}, s \in S\}$, which is a penalized problem corresponding to an (LBP) where the first level constraint is the cut $c^T z \geq c^T \hat{z} + \varepsilon$.

After the cut, (BEPA) may not work properly, i.e. the returned point is not assured to be an (EqP) for $\tilde{P}(M)$. So, the General Equilibrium Point Algorithm (GEPA) is applied instead. It is formally described in [9]. Basically, it repeats steps 1 and 2 of (BEPA) restarting with $z^0 = \hat{z}$ until an equilibrium is found. This iterative algorithm is well-defined. Indeed, after a finite number of iterations it finds an (EqP) or verifies that $\tilde{P}(M)$ is unbounded for all $M \geq 0$ [9, Propositions 9 and 10]. Unlike (BEPA), the obtained (EqP) may not satisfy the complementarity condition. In this case an outer approximation procedure [13] is used.

A schematic description of the algorithm is provided below.

Penalty Algorithm

Step 1: If $S = \emptyset$ then go to **Step 6** else set $\tilde{Z} = Z$ and $j = 0$;

Step 2: If $\tilde{Z} = \emptyset$ then go to **Step 6** else find $z^0 \in \tilde{Z}$;

Step 3: Apply (BEPA) if $j = 0$, or (GEPA) otherwise. If $\tilde{P}(M)$ is unbounded for all $M \geq 0$, stop: (LBPP) is unbounded. Otherwise, let $(\hat{z}, \hat{s})$ be the obtained (EqP);

Step 4: If $\hat{z}^T \hat{s} = 0$ then set $z_{opt} = \hat{z}$ and update $\tilde{Z} = Z \cap \{z \in \mathcal{R}^n : c^T z \geq c^T z_{opt} + \varepsilon\}$. Set $j = j + 1$ and go to **Step 2**;

Step 5: If $\hat{z}^T \hat{s} > 0$ apply a modified version of the outer approximation algorithm developed in [8] to the problem $\min\{s^T z : (z, s) \in Z' \times S\}$, where $Z' = \{z \in Z : c^T z = c^T z_{opt} + \varepsilon\}$. Let (z^*, s^*) be the obtained solution. If ${z^*}^T s^* < \hat{z}^T \hat{s}$ then restart $z^0 = z^*$ and go to **Step 3**;

Step 6: Stop. If $j = 0$, (LBPP) is infeasible. Otherwise, z_{opt} is an ε-global solution of (LBPP).

We remark that at step 5 the outer approximation procedure looks for a feasible point of (LBPP) just on the hyperplane supporting the cut. Since the feasible set of (LBPP) is connected [5], there is a feasible point of (LBPP) in $\tilde{Z}$ if, and only if, there is a feasible point of (LBPP) in $Z' \subset \tilde{Z}$. Also, using set Z' instead of set $\tilde{Z}$ reduces the problem to be considered.

The penalty algorithm stops in a finite number of steps and finds an ε-global solution to (LBPP) or verifies that the problem is unbounded or infeasible [9, Propositions 12 and 13].

2.2 The branch-and-bound algorithm

The second studied algorithm uses a branch-and-bound strategy developed by Hansen *et al.* [12]. In [12] it is compared with two other efficient algorithms developed by Bard and Moore [3] and Judice and Faustino [15]. It always outperformed Bard and Moore's algorithm. Also it got computational times similar to those of Júdice and Faustino, even using a less powerful machine. Therefore, it is a competitive algorithm known in the literature.

Unlike the penalty algorithm, this method was designed to work on problem (LBP). On the other hand, it assumes that the sets $\{(x, y) \in R^{n_1 \times n_2} : A_1x + A_2y \leq a, x \geq 0, y \geq 0\}$ and $\{(x, y) \in R^{n_1 \times n_2} : A_1x + A_2y \leq a, B_1x + B_2y \leq b, x \geq 0, y \geq 0\}$ are non empty and bounded. A brief description of the algorithm is provided below.

The basic idea is to exploit necessary optimality conditions expressed in terms of subsets of the follower's constraints where at least one constraint must be tight at an optimum. The tightness of ith follower's constraint is represented by a binary variable α_i which takes value 1 if the constraint must be tight and 0 otherwise. Hansen *et al.* [12] show that the following set (R) of relations is valid at any feasible solution of (LBP):

$$\sum_{i|A_{2ij}>0} \alpha_i \geq 1, \qquad if\ d_j > 0$$

$$\alpha_{m_2+j} + \sum_{i|A_{2ij}<0} \alpha_i \geq 1, \qquad if\ d_j < 0$$

for $j = 1, 2, \cdots, n_2$.

The branching process at each node is carried out by using one of the following two strategies. In the first one a single variable α_i is chosen, fixed at 1 and then at 0. In the second one a relation of the set (R) is selected, for example $\alpha_{i1} + \alpha_{i2} + \alpha_{i3} + \cdots \geq 1$, and the branch follows the rule $\alpha_{i1} = 1$ or $(\alpha_{i1} = 0$ and $\alpha_{i2} = 1)$ or $(\alpha_{i1} = \alpha_{i2} = 0$ and $\alpha_{i3} = 1)$ and so on.

When α_i is fixed at 1, the ith constraint becomes an equality. This allows to eliminate one of the follower's variables thus generating a subproblem with less variables. When α_i is fixed at 0, the ith constraint becomes a strict inequality. According to the complementary slackness theorem, the follower's dual subproblem with its ith variable fixed at zero is considered instead of the primal.

Let us denote by (RLBP) the relaxed problem obtained from (LBP) by omitting the follower's objective function (4). In order to simplify the notation, we also consider (RLBP) as being the relaxed problem of the subproblems (in which some follower's variables have been eliminated).

An upper bound for each subproblem is obtained by the optimal solution of (RLBP). A lower bound for the problem is given by the value of the best incumbent solution.

At each node a new set of relations (R) is found and the remaining relations are updated appropriately. Hansen *et al.* [12] also use the concept of penalties similar to those used in mixed integer linear programming to improve their pruning rules. Given a relation $r_k \in (R)$, say $r_k \equiv \sum_{i \in I_k} \alpha_i \geq 1$, a penalty p_i $(i \in I_k)$ for setting the ith slack variable to zero is calculated. A better comprehension of this algorithm is attained in the short description below:

Branch-and-bound Algorithm

Step A: Initialize (x_{opt}, y_{opt}) and $f_{opt} = c_1{}^T x_{opt} + c_2{}^T y_{opt}$ with a heuristic solution. If there is no heuristic available set $f_{opt} = -\infty$. Consider free all variables α_i and set $(R) = \emptyset$;

Step B: Solve (RLBP). Let (x_L, y_L) be an optimal solution; If $f_L = c_1{}^T x_L + c_2{}^T y_L \leq f_{opt}$ then go to **Step M**;

Step C: If the follower's dual subproblem is infeasible then go to **Step M**;

Step D: If (x_L, y_L) is infeasible to the current subproblem go to **Step F**;

Step E: If (x_L, y_L) is feasible to the initial problem, update the incumbent solution (if $c_1{}^T x_L + c_2{}^T y_L > f_{opt}$) and go to **Step M**;

Step F: Compute penalties p_i associated with the optimal tableau of (RLBP). For all $r_k \in (R)$, compute $\pi_k = \min\{p_i : i \in I_k\}$. Set $\Pi = \max_k \ \pi_k$. If $f_{opt} \geq f_L - \Pi$ then go to **Step M**;

Step G: If (RLBP) is infeasible then go to **Step M**;

Step H: Consider again the penalties calculated at step F. For all i such that $f_{opt} \geq f_L - p_i$, fix α_i at zero and update set (R) appropriately;

Step I: If set (R) contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$ then go to **Step M**;

Step J: For all remaining y_j appearing in the subproblem, add to set (R) the logical relations on the α_i, if they are nonredundant. Eliminate from (R) those relations that have become redundant.

Step K: If (R) contains a relation with all $\alpha_i = 0$ except for one index i, then set the corresponding $\alpha_i = 1$, eliminate a variable y_j remaining in the ith constraint of the follower's subproblem such that fill-in is minimum and return to **Step B**;

Step L: Apply the selected branching rule to choose a free variable α_i or a relation $r_k \in (R)$ for which all variables α_i with $i \in I_k$ are free. In the first case branch by fixing α_i at 1. In the second case branch by fixing the first variable α_i at 1. Eliminate a variable y_j remaining in the ith constraint of the follower's subproblem such that fill-in is minimum and return to **Step B**;

Step M: Backtracking according to the branching rule applied to the node.

In our implementation we use the branching rule (BR6) due to the fact that it presented the best computational results [12]. It is performed as follows:

- select the relation $r_k \in (R)$ with two variables α_i and both the corresponding follower's variables z_i in basis, such that $\sum z_i s_i$ is maximum, where s_i is the related dual variable.
- If there is no such relation, branch on the variable with largest penalty p_i among those for which $z_i s_i > 0$;

3 Proposed modifications

The aim of this section is to introduce modifications in the studied algorithms in order to improve their computational performance. The main difficulties in each algorithm were identified from computational experience. We propose some modifications in each algorithm as described in the following subsections.

3.1 The modified penalty algorithm

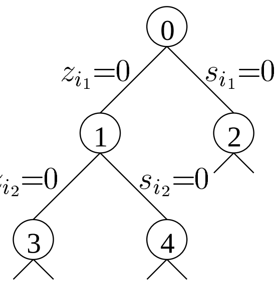
In tests with the original penalty algorithm using test problems to be described in section 4 we found that some of them could not be solved in a limit time of 600s. We observed that the most time consuming step of the penalty algorithm is the outer approximation procedure as already pointed out in [9].

Such a procedure, applied at step 5, attempts to solve the following bilinear problem:

$$(P') \quad \min\{g(z,s) = s^T z : (z,s) \in Z' \times S\},$$

where $Z' = \{z \in Z : c^T z = c^T z_{opt} + \varepsilon\}$. Actually, the objective is to verify if there is a point $(z,s) \in Z' \times S$ such that $z^T s = 0$.

Instead of the outer approximation procedure we are going to use a simple enumerative method that attempts to find a complementary point (z,s) in $Z' \times S$. The enumerative process can be visualized by the following binary tree:



where z_j and s_j are the jth variables of z and s respectively for $j \in \{1, 2, \ldots, n\}$.

To each node k of the tree we associate subsets $Z'_k \subseteq Z'$ and $S_k \subseteq S$. Also, we find a point $(z^k, s^k) \in Z'_k \times S_k$, if these sets are nonempty. At the initial node ($k = 0$), we set $Z'_0 = Z'$ and $S_0 = S$. Now, consider a node $k \geq 1$ which is generated from a node p by fixing z_i or s_i at 0. If $z_i = 0$ then $Z'_k = Z'_p \cap \{z \in \mathcal{R}^n : z_i = 0\}$ and $S_k = S_p$. If $s_i = 0$ then $S_k = S_p \cap \{s \in \mathcal{R}^n : s_i = 0\}$ and $Z'_k = Z'_p$. For example in the tree above we have $Z'_1 = Z'_0 \cap \{z \in \mathcal{R}^n : z_{i_1} = 0\}$ or still $S_4 = S_1 \cap \{s \in \mathcal{R}^n : s_{i_2} = 0\}$.

Besides, at each node k where $Z'_k \times S_k \neq \emptyset$ we apply a procedure similar to the *Mountain Climbing Algorithm* [17] starting from z^k or s^k. This procedure called Adapted Equilibrium Point Algorithm (AEPA) is a simplification of (GEPA) that aims to find a local solution of problem (P'). A description of (AEPA) starting from a point $\bar{z} := z^k$ is given below. A similar procedure can be designed starting from a point $\bar{s} := s^k$.

Adapted Equilibrium Point Algorithm (AEPA)

Step 1: Set $z^0 = \bar{z}$ and $l = 0$.

Step 2: Solve the problem $\min\{(z^0)^T s : s \in S\}$ obtaining a solution s^l.

Step 3: Solve the problem $\min\{(s^l)^T z : z \in Z'\}$ obtaining a solution z^{l+1}. If $(s^l)^T z^l = (s^l)^T z^{l+1}$ then STOP and return $(\tilde{z}, \tilde{s}) = (z^l, s^l)$.

Step 4: Solve the problem $\min\{(z^{l+1})^T s : s \in S\}$ obtaining a solution s^{l+1}. If $(s^l)^T z^{l+1} = (s^{l+1})^T z^{l+1}$ then STOP and return $(\tilde{z}, \tilde{s}) = (z^{l+1}, s^l)$. Otherwise set $l = l + 1$ and go to step 3.

The algorithm halts in a finite number of iterations, since each iteration moves from one vertex of $Z' \times S$ to another and the value of function $g(z, s) = z^T s$ never increase. Also, it finds a stationary point of problem (P') as shows the following proposition.

Proposition 4 *The point $(\tilde{z}, \tilde{s})$ returned by (AEPA) satisfies:* **(1)** $\tilde{s} \in \arg\min\{\tilde{z}^T s : s \in S\}$ *and* $\tilde{z} \in \arg\min\{\tilde{s}^T z : z \in Z'\}$, **(2)** *it is a KKT point of problem (P'),* **(3)** *if $\tilde{s} = \arg\min\{\tilde{z}^T s : s \in S\}$ or $\tilde{z} = \arg\min\{\tilde{s}^T z : z \in Z'\}$, it is a strict local solution of problem (P').*

Proof. **(1)** trivially follows by the definition of (AEPA).

(2) By (1), $\tilde{s}$ and $\tilde{z}$ are KKT points for the linear programs $\min\{\tilde{z}^T s : s \in S\}$ and $\min\{\tilde{s}^T z : z \in Z'\}$, respectively. Moreover, the KKT conditions for (P') at $(\tilde{z}, \tilde{s})$ are exactly the union of the KKT conditions for the linear programs. Hence, the result follows.

(3) By (1), we have that $\tilde{s}^T(z - \tilde{z}) \geq 0$ for all $z \in Z'$ and $\tilde{z}^T(s - \tilde{s}) \geq 0$ for all $s \in S$. In addition, by the hypothesis, it follows that $\nabla g(\tilde{z}, \tilde{s})^T \begin{pmatrix} z - \tilde{z} \\ s - \tilde{s} \end{pmatrix} = \tilde{s}^T(z - \tilde{z}) + \tilde{z}^T(s - \tilde{s}) > 0$ for all $(z, s) \in Z' \times S$. As $Z' \times S$ is a convex set, this means that the directional derivative of g at $(\tilde{z}, \tilde{s})$ with respect to any feasible direction is positive. Therefore, $(\tilde{z}, \tilde{s})$ is a strict local solution of (P'). ∎

Without the condition considered in Proposition 4(3), we can still get a weaker local optimality condition. We say that a basic feasible solution $(\tilde{z}, \tilde{s})$ of (P') is a local star minimum if $g(z, s) \geq g(\tilde{z}, \tilde{s})$ for all adjacent basic feasible solution (z, s) [15].

Proposition 5 *The point returned by (AEPA) is a local star minimum of problem* (P').

Proof. Let $(\tilde{z}, \tilde{s})$ be the basic feasible solution returned by (AEPA). Let (z, s) be an adjacent basic feasible solution. Then, either $z = \tilde{z}$ or $s = \tilde{s}$. In any case, by Proposition 4(1), it follows that $z^T s \geq \tilde{z}^T \tilde{s}$. Therefore, $(\tilde{z}, \tilde{s})$ is a local star minimum. ∎

Now we are able to present the enumerative algorithm that seeks a complementary point in $Z' \times S$. Let us consider the sets I_z and I_s of indices i of the variables z_i and s_i fixed at 0 respectively.

Enumerative Algorithm

Step 1: Set $I_z = \emptyset$, $I_s = \emptyset$ and $g_{opt} = \infty$. Set $k = 0$.

Step 2: Update $Z'_k = Z' \cap \{z \in \mathcal{R}^n : z_i = 0, \forall i \in I_z\}$ and $S_k = S \cap \{s \in \mathcal{R}^n : s_i = 0, \forall i \in I_s\}$. If $Z'_k = \emptyset$ or $S_k = \emptyset$ then go to step 6. Otherwise let $\bar{z}$ and $\bar{s}$ be points of Z'_k and S_k respectively, obtained by phase 1 of simplex method.

Step 3: Apply (AEPA) from $\bar{z}$ obtaining a point $(\tilde{z}, \tilde{s})$. If $g_{opt} \geq \tilde{z}^T \tilde{s}$ then set $g_{opt} = \tilde{z}^T \tilde{s}$ and $(z^*, s^*) = (\tilde{z}, \tilde{s})$. Apply (AEPA) from $\bar{s}$ obtaining a point $(\tilde{z}, \tilde{s})$. If $g_{opt} \geq \tilde{z}^T \tilde{s}$ then set $g_{opt} = \tilde{z}^T \tilde{s}$ and $(z^*, s^*) = (\tilde{z}, \tilde{s})$.

Step 4: If $g_{opt} = 0$ then STOP.

Step 5: Select a variable z_i such that the product $\bar{z}_i \bar{s}_i$ is maximum. Fix this variable at 0 and set $I_z = I_z \cup \{i\}$. Set $k = k + 1$ and go to step 2.

Step 6: Attempt to find the last variable z_i branched upon and fixed at 0. If there is no such variable, STOP. Otherwise, free z_i, update $I_z = I_z \backslash \{i\}$, fix variable s_i at 0 and update $I_s = I_s \cup \{i\}$. Also, free all variables s_j for $j \in I_s^i$, where $I_s^i \subset I_s$ is the subset of indices j of the variables s_j fixed at 0 after z_i was fixed at 0, update $I_s = I_s \backslash I_s^i$, set $k = k + 1$ and go to step 2.

The modified penalty algorithm we propose replaces the original **Step 5** by the following:

New Step 5: If $\hat{z}^T \hat{s} > 0$ apply the Enumerative Algorithm obtaining the point (z^*, s^*). If ${z^*}^T s^* = 0$ then restart $z^0 = z^*$ and go to **Step 3**;

We remark that this enumerative method is similar to that one used in [15] but with some diferences related with the branching and prunning processes.

3.2 The modified branch-and-bound algorithm

In the branch-and-bound described in subsection 2.2 lower and upper bound are used as follows. The lower bound is given by the leader's objective function value of the best feasible point found. The upper bound at each node is given by the optimum value of the corresponding relaxed subproblem. If the upper bound is less than the lower bound, the subproblem does not need to be fathomed and a backtracking is carried out. Therefore, it is easy to perceive that as soon as a better feasible solution is found, the lower bound is enhanced and consequently less nodes may be explored in the branch-and-bound tree.

Attempting to provide strong lower bounds, we introduce some modifications in the branch-and-bound algorithm. Meanwhile, the computational effort of the original version is maintained. The idea is to find a new feasible solution at each node. Actually, we propose a merged method in which the basic equilibrium point procedure (BEPA) replaces steps D and E of the branch-and-bound algorithm. The method is designed for the problem with no constraints in the first level (LBPP) so that a feasible equilibrium point is attained in just two linear problems.

The modified branch-and-bound algorithm comprises steps A to M as before, except for steps D and E that are altered as below. First, let us remember the definitions of vectors $c^T = (c_1^T, c_2^T, 0)$, $z^T = (x^T, y^T, w^T)$ and $s^T = (0, v^T, u^T)$, the block matrices $A = [A_1\, A_2\, I_m]$ and $D = [0 - I_{n_2}\, A_2^T]$, and the sets $Z = \{z \in \mathcal{R}^n : Az = a, z \geq 0\}$ and $S = \{s \in \mathcal{R}^n : Ds = d, s \geq 0\}$.

New Step D

Step D.1: Let (x_L, y_L) be a solution of the current relaxed subproblem and set $(z^0)^T = (x_L^T, y_L^T, w_L^T) \in Z$.

Step D.2: Solve $\max\{c^T z^0 - M(z^0)^T s : s \in S\}$ using de simplex method, obtain a solution $\hat{s}$;

New Step E

Step E.1: Solve $\max\{c^T z - M\hat{s}^T z : z \in Z\}$ using the simplex big-M method [4], obtain a feasible point $\hat{z}$ to (LBPP). If $f_{opt} < c^T\hat{z}$ then set $f_{opt} = c^T\hat{z}$ and update the incumbent solution;

Step E.2: If ${c_1}^T x_L + {c_2}^T y_L \leq f_{opt}$ then go to **Step M**;

Let us note that steps D.2 and E.1 comprise procedure (BEPA). We remember that the equilibrium point algorithm seeks, among those points $z \in Z$ satisfying $(s^0)^T z = 0$, that one which maximizes the leader's objective function. Thus, the algorithm provides strong lower bounds in the branch-and-bound tree.

It is important to observe that the same number of linear programs are considered at the original and new steps D and E. Therefore, the computational effort is expected to be similar. Also, as the optimal tableaux of the linear problems solved at steps D.2 and E.1 are stored, a post optimization procedure is carried out at each node. Let us observe that we are just replacing the objective function in each tableau.

4 Computational Results

The algorithms were coded in Pascal and compiled in DELPHI. They were ran on a Pentium II 333 MHz. All of them used the same simplex procedure also coded in Pascal. We remark that for the penalty algorithm the value of ε was fixed at 0.013 and for the original branch-and-bound algorithm the heuristic proposed by autors [12] to be used in absence of first level constraints was applied.

The test problems were generated randomly using a method developed in [7]. The method allows us to control the density and the conditioning number of the constraint matrix, and the number of global and local optima of the problem. Actually, it first generates a separable linear bilevel problem with the desired dimensions and number of optima. Then, it disguises the separability with a matrix transformation $HD^{-1}H$ while keeping the number of optima. The invertible diagonal matrix D and the block-diagonal Householder matrix H are randomly generated so that 10^{δ} is the conditioning number of D and τ is the percentual of nonzero entries in the basis vector of H, where δ and τ are given parameters (see [7] for details). In accordance with the author's suggestion some modifications were introduced in the method to reduce the number of constraints. Also, nonnegativeness constraints for all variables were added to each generated instance.

We consider 384 tests problems that were generated with the following characteristics:

1. The total number of variables $n_1 + n_2$ ranges from 50 up to 200, and either 25% or 75% of them were attributed to the leader;
2. The number of constraints m is around 30% of the total number of variables, and no constraints were attributed to the leader;
3. For a given number of variables, three values of m_2 were considered. The ith problem ($i = 1, 2, 3$), related to the ith value of m_2, has 2^i local solutions (one of them is a global one) before considering $x \geq 0$ and $y \geq 0$;
4. The parameter δ that controls the conditioning number of the constraint matrix was set to 1.
5. The parameter τ that influences the density of the constraint matrix was set to 10%, 40%, 60% and 80%.

The generated problems were divided in four groups according to the values of τ. Each group contains 96 test problems. Table 1 presents the mean density (μ_d), the standard deviation (σ_d), and the minimum ($\min_d$) and maximum ($\max_d$) density in each the group.

τ (%)	μ_d (%)	σ_d (%)	$\min_d$ (%)	$\max_d$ (%)
10	4.5	1.45	2.3	8.4
40	18.8	2.10	16	26
60	41.0	2.4	33.0	45.5
80	63.5	3.5	50.0	72.0

Table 1: Summary statistics for densities of the groups of problems

The four described algorithms were applied to each test problem. The computational results for the modified penalty algorithm, the original branch-and-bound algorithm and the modified branch-and-bound algorithm are presented in Tables 3a, 3b, 3c and 3d. Each of these tables describes a group of problems with the same value of τ. The numbers for the original penalty algorithm are not presented since it was always considerably outperformed by its modified version. Also, it was not able to solve the majority of the problems with more than 100 variables in a limit time of 600s.

A row of each Table 3a to 3d comprises three problems with the same number of leader's and follower's variables. The ith problem ($i = 1, 2, 3$) relates to columns m_i, T_i, N_i and S_i which respectively denote the number of constraints and the elapsed time in seconds, for the three algorithms studied, the number of nodes and generated subproblems, for the branch-and-bound algorithms. We remark that the number of generated subproblems gives us a better ideia of the computational effort than the number of nodes, since more than one subproblem may be solved at each node.

Table 2 presents a summary of the spending times given in Tables 3a to 3d. For each group of problems and each algorithm, the table provides the mean elapsed time (μ_T) and the standard deviation (σ_T) in seconds. The calculation does not include 22 problems whose solution is the solution of the relaxed problem. These problems correspond to entries in the tables where no subproblems were generated by the original branch-and-bound algorithm.

From Table 2 it seems that:

- The original branch-and-bound algorithm outperforms the penalty algorithm in sparser problems. However, they are competitive when applied to denser problems.
- The modification introduced in the branch-and-bound algorithm leads to an improvement and the gain increases while the density increases. From the detailed tables, we can also infer that the improvement is greater when comparing the number of generated nodes or subproblems.
- The modified branch-and-bound algorithm outruns the modified penalty algorithm and the difference is quite similar in all groups of problems.
- The standard deviations increase as the density increases for the branch-and-bound algorithm but they remain similar for the penalty algorithm. Also, the insertion of the equilibrium point procedure in the branch-and-bound algorithm seems to reduce the standard deviation of the spending times.

Finnaly, we present the percentage of problems in which a certain algorithm was better than or comparable (time difference less than 0.05s) to another. The original branch-and-bound algorithm outran and was comparable to the penalty algorithm in 49% and 12% of the 384 problems respectively. However, we remark that these numbers could have been lower if the

	Modified Penalty Algorithm		Original Branch-and-Bound		Modified Branch-and-Bound	
τ (%)	μ_T (s)	σ_T (s)	μ_T (s)	σ_T (s)	μ_T (s)	σ_T (s)
10	3.29	5.23	2.53	2.66	2.30	2.52
40	3.01	5.16	2.56	2.82	2.01	2.56
60	3.51	5.29	2.82	3.83	2.07	2.48
80	3.63	4.56	3.74	5.99	2.42	3.01

Table 2: Summary statistics for elapsed times of the groups of problems

penalty algorithm had been started at the solution of the relaxed problem. When comparing the branch-and-bound algorithms, the modified version had a better and a comparable performance in 61% and 32% of all the problems respectively.

5 Concluding remarks

The orginal penalty algorithm may be very time consuming, since the number of vertices enumerated by the outer approximation method may be really large. However, the substitution of the outer appoximation by a simple enumerative procedure could considerably improve the overall performance. Such a result indicates that the equilibrium point procedure can be efficiently used together with an appropriate global optimization method.

We recall that the used enumerative procedure applies the LIFO branching strategy and prunes only nodes related to infeasible subproblems. The use of heuristics to establish criteria for choosing the node to be branched and the application of prunning rules based on bounds may lead to a greater improvement. These claims deserve a detailed study.

Regarding the branch-and-bound algorithm, the computational results ratified its efficiency. Moreover, the insertion of the equilibrium point procedure could make its performance even better. Again, it seems that the repeated use of the equilibrium point procedure in an attempt to find good feasible solutions can be worthwhile.

In the computational experiments the branch-and-bound algorithm outperformed on average the penalty algorithm. Nevertheless, it is worth noting that the test problems used satisfy the compactness assumptions assumed by the branch-and-bound algorithm. For problems where these assumptions are not valid, the comparison between these two algorithms may lead to different results. In fact, the penalty algorithm is designed to work with no compactness assumption. So, its behavior is not expected to change very much. On the other hand, the prunning rules of the branch-and-bound algorithm are strongly based on the solution value of the relaxed problem. Thus, the spending times may increase. Anyway, this is a matter for future work.

References

[1] M. Amouzegar and K. Moshirvaziri (1998). A penalty method for linear bilevel programming problems. In: A. Migdalas, P. Pardalos and P. Värbrand (eds). Multilevel optimization: algorithms and applications. *Nonconvex Optimization and its Applications* 20, 251–271, Kluwer.

[2] M. Amouzegar and K. Moshirvaziri (1999). Determining optimal pollution control policies: an application of bilevel programming. *European J. of Operational Research* 119, 100-120.

[3] J. Bard and J. Moore (1990). A branch and bound algorithm for the bilevel programming problem. *SIAM J. Scientific and Stat. Computing* 11, 281-292.

[4] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali (1990). *Linear Programming and Network Flows*, Wiley, 2nd edition, New York.

[5] H. Benson (1989). On the structure and properties of a linear multilevel programming problem. *Journal of Optimization Theory and Applications* 60, 353–373.

[6] W.F. Bialas and M.H. Karwan (1984). Two-level linear programming. *Manegement Science* 30(8), 1004-1020.

[7] P. Calamai and L. Vicente (1993). Generating linear and linear-quadratic bilevel programming problems. *SIAM J. Scientific and Stat. Computing* 14, 770-782.

[8] M. Campêlo (1999). *Programação Linear em Dois Níveis: uma abordagem teórica e computacional.* PhD Thesis, Univ. Federal do Rio de Janeiro, COPPE/Engenharia de Sistemas e Computação. Written in Portuguese.

[9] M. Campêlo and S. Scheimberg (2001). Theoretical and computational results for a linear bilevel problem. In: N. Hadjisavvas and P. Pardalos (eds). Advances in Convex Analysis and Global Optimization. *Nonconvex Optimization and its Applications* 54, 269-281, Kluwer.

[10] W. Candler and R. Townsley (1982). A linear two-level programming problem. *Computers and Operations Research* 9, 56-76.

[11] J. Clegg, M. Smith, Y. Xiang and R. Yarrow (2001). Bilevel programming applied to optimising urban transportation *Transportation Research Part B* 35, 41-70.

[12] P. Hansen, B. Jaumard and G. Savard (1992). New branch-and-bound rules for linear bilevel programming. *SIAM J. Scientific and Statistical Computing* 13(5), 1194-1217.

[13] R. Horst and H. Tuy (1990), *Global Optimization: Deterministic Approaches*, Spring-Verlag, Berlin.

[14] S.M.N. Islam (1997). Sustainable Economic Developments in the Australian Energy Sector: Findings of the Australian Energy Planning System Optimization Model (AEPSOM). *Renewable and Sustainable Energy Reviews* 1(3), 229-238.

[15] J.J. Júdice and A.M. Faustino (1992). A sequential LCP method for bilevel linear programming. *Annals of Operations Research* 34, 89-106.

[16] U. Kirsch (1997). Two-level optimization of prestressed structures. *Engineering Structures* 19(4), 309–317.

[17] H. Konno (1976). A cutting plane algorithm for solving bilinear programs. *Mathematical Programming* 11, 14-27.

[18] H. Tuy and S. Ghannadan (1998). A new branch-and-bound method for bilevel linear programs. In: A. Migdalas, P. Pardalos and P. Värbrand (eds). Multilevel optimization: algorithms and applications. *Nonconvex Optimization and its Applications* 20, 231–249, Kluwer.

[19] H. Tuy, A. Migdalas and P. Värbrand (1993). A global optimization approach for the linear two-level program. *J. of Global Optimization* 3, 1-23.

[20] L.N. Vicente and P.H. Calamai (1994). Bilevel and multilevel programming: a bibliography review. *J. of Global Optimization* 5, 291-306.

[21] D.J. White and G. Anandalingam (1993). A penalty function approach for solving bi-level linear programs. *J. of Global Optimization* 3, 397-419.

[22] H. Yang and S. Yagar (1995). Traffic Assigment and signal control in saturated road networks. *Transportation Research A* 29A(2), 125-139.

					Modified Penalty			Original B&B Algorithm									Modified B&B Algorithm								
n_1+n_2	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.06	0.05	0.11	0.00	0	12	0.05	2	13	0.06	6	18	0.05	0	12	0.06	2	13	0.05	6	15
50	38	15	17	19	0.05	0.05	0.28	0.05	2	12	0.06	2	14	0.11	6	19	0.06	0	11	0.11	2	14	0.05	6	17
60	15	18	20	22	0.05	0.11	0.27	0.05	0	13	0.05	2	16	0.11	6	21	0.06	0	13	0.11	2	15	0.11	6	18
60	45	18	20	22	0.11	0.27	0.33	0.05	2	13	0.17	2	17	0.16	8	23	0.11	0	12	0.11	2	15	0.17	2	17
70	17	20	22	24	0.11	0.16	0.49	0.11	2	16	0.11	2	16	0.16	6	23	0.11	0	15	0.11	2	15	0.17	6	20
70	53	20	22	24	1.04	0.22	0.71	0.22	2	17	0.21	2	19	0.28	6	24	0.17	0	16	0.17	2	17	0.22	2	20
80	20	23	25	27	0.11	0.27	0.5	0.17	0	18	0.22	2	20	0.27	6	26	0.22	0	18	0.22	2	18	0.33	6	25
80	60	23	25	27	0.22	0.22	0.55	0.28	2	18	0.33	2	22	0.38	4	20	0.33	0	17	0.33	2	20	0.33	2	17
90	22	25	27	29	0.17	0.33	0.49	0.33	0	22	0.39	4	25	0.44	6	28	0.33	0	22	0.33	2	24	0.44	6	28
90	68	25	27	29	0.33	3.57	1.81	0.44	2	22	0.55	2	22	0.66	6	28	0.44	0	21	0.49	2	21	0.55	6	25
100	25	28	20	32	0.28	0.77	0.77	0.55	2	26	0.66	6	29	0.61	2	25	0.44	0	25	0.55	6	29	0.55	2	25
100	75	28	20	32	0.71	1.93	1.26	0.82	4	30	0.82	6	29	0.76	2	21	0.72	0	25	0.82	6	29	0.28	0	1
110	27	30	32	34	0.39	1.21	0.99	0.66	2	26	0.82	6	29	0.82	2	27	0.66	0	25	0.77	2	27	0.77	2	27
110	83	30	32	34	0.60	2.25	2.04	1.05	2	27	1.21	6	31	1.32	8	30	1.04	2	27	1.04	2	29	1.26	8	30
120	30	33	35	37	0.44	1.10	0.82	0.99	0	28	1.20	6	32	1.15	2	30	0.93	0	28	1.05	2	30	1.1	2	30
120	90	33	35	37	2.20	3.30	15.93	1.65	6	34	1.70	6	34	1.81	8	33	0.60	0	5	1.54	2	32	1.76	8	33
130	32	35	37	39	0.43	1.32	0.88	1.32	0	32	1.48	6	33	1.54	2	33	1.27	0	32	1.37	2	31	1.37	2	33
130	98	35	37	39	1.04	4.12	21.37	2.03	2	32	2.14	6	36	2.58	14	41	1.87	0	31	2.09	6	36	2.25	10	37
140	35	38	40	42	0.55	1.76	1.37	1.76	0	33	2.03	6	36	1.92	2	33	1.59	0	33	1.76	2	34	1.82	2	33
140	105	38	40	42	1.26	6.15	3.08	2.69	2	36	2.91	6	39	3.08	8	37	2.48	0	35	2.64	2	37	2.96	8	37
150	37	40	42	44	0.77	2.36	1.87	2.14	2	34	2.41	6	38	2.47	2	36	1.92	0	33	2.31	6	38	2.25	2	36
150	113	40	42	44	28.95	7.58	26.2	3.19	2	35	3.57	6	41	4.01	14	43	3.07	2	35	3.35	6	41	3.85	14	43
160	40	43	45	47	0.83	5.22	1.98	2.81	2	37	3.02	6	40	3.13	2	38	2.52	0	36	2.80	6	40	2.91	2	38
160	120	43	45	47	2.48	8.79	5.44	4.06	2	38	4.62	6	44	4.34	2	35	3.84	0	37	4.17	2	42	4.01	2	34
170	42	45	47	49	0.94	2.09	2.75	3.46	2	40	3.51	4	40	3.52	2	36	3.08	0	39	3.18	2	39	3.29	2	37
170	128	45	47	49	2.14	4.28	4.83	5.33	2	43	5.49	2	44	5.76	8	43	4.94	0	42	5.00	2	42	5.61	8	43
180	45	48	50	52	1.21	5.61	3.35	4.40	0	42	4.56	4	42	4.72	2	42	4.01	0	42	4.17	2	41	4.29	2	41
180	135	48	50	52	2.59	13.40	6.65	6.48	2	43	7.03	6	49	7.03	2	41	6.09	0	42	6.59	6	49	6.7	2	41
190	47	50	52	54	1.59	6.15	4.23	5.10	2	43	5.39	6	45	5.43	2	42	4.51	0	42	4.94	6	45	4.94	2	42
190	143	50	52	54	8.40	7.47	17.14	7.91	6	47	8.13	2	49	9.22	6	49	7.36	6	47	7.64	2	49	9.45	6	46
200	50	53	55	57	2.30	6.04	4.94	6.54	2	46	6.70	4	46	6.54	2	43	5.88	0	45	1.26	0	5	6.1	2	44
200	150	53	55	57	3.46	8.68	9.5	9.67	2	51	10.1	2	52	10.4	2	52	9.01	0	50	9.45	2	52	11.09	2	52

Table 3a: Computational results for problems with $\tau = 10\%$

					Modified Penalty			Original B&B Algorithm									Modified B&B Algorithm								
n_1+n_2	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.00	0.17	0.11	0.06	2	10	0.05	2	7	0.05	2	9	0.00	0	0	0.00	0	6	0.06	2	9
50	38	15	17	19	0.05	0.11	0.39	0.11	6	13	0.11	6	13	0.82	92	180	0.06	6	13	0.05	6	13	0.66	82	153
60	15	18	20	22	0.06	0.06	0.17	0.06	0	10	0.05	2	10	0.11	2	11	0.05	0	10	0.00	0	0	0.05	0	10
60	45	18	20	22	0.17	0.11	0.22	0.11	2	11	0.16	6	17	0.16	8	16	0.06	0	10	0.17	6	17	0.17	8	16
70	17	20	22	24	0.06	0.11	0.49	0.06	2	10	0.11	2	12	0.11	4	12	0.05	0	1	0.11	0	11	0.11	2	10
70	53	20	22	24	0.22	1.70	0.83	0.22	6	18	0.44	24	45	2.58	308	324	0.22	6	18	0.50	24	45	0.66	48	58
80	20	23	25	27	0.11	0.17	0.44	0.17	2	12	0.17	2	12	0.17	4	12	0.00	0	1	0.16	0	11	0.16	2	10
80	60	23	25	27	0.38	0.49	6.04	0.33	6	16	1.16	80	92	2.58	200	209	0.33	6	16	0.33	6	18	2.47	194	203
90	22	25	27	29	0.11	0.28	0.49	0.22	2	13	0.28	2	15	0.33	6	16	0.05	0	1	0.27	0	14	0.28	2	13
90	68	25	27	29	0.44	0.55	6.76	0.66	12	25	0.55	6	18	2.63	138	151	0.50	6	19	0.55	6	18	2.64	132	145
100	25	28	20	32	0.22	0.66	0.65	0.39	2	17	0.38	2	15	0.50	4	17	0.38	0	16	0.11	0	1	0.11	0	1
100	75	28	20	32	0.66	1.10	5.50	0.77	6	21	0.99	6	23	4.23	150	165	0.77	6	21	0.94	6	23	3.46	122	137
110	27	30	32	34	0.49	0.99	0.38	0.50	0	16	0.71	10	25	0.66	2	17	0.44	0	16	0.77	10	25	0.11	0	1
110	83	30	32	34	0.71	21.20	2.80	0.88	2	16	1.48	14	31	1.59	14	29	0.88	2	16	1.48	14	31	1.60	14	29
120	30	33	35	37	0.71	1.05	0.28	0.83	2	19	0.72	2	18	0.77	2	16	0.33	0	5	0.66	0	17	0.11	0	0
120	90	33	35	37	0.66	2.20	2.59	1.49	6	23	1.70	6	23	2.31	18	34	1.37	6	23	1.65	6	23	1.53	6	22
130	32	35	37	39	0.50	1.15	0.33	1.04	2	21	1.10	2	22	1.04	2	19	0.88	0	20	0.99	0	21	0.22	0	1
130	98	35	37	39	1.76	1.21	10.33	1.65	2	20	2.03	6	26	2.75	14	34	1.59	2	20	1.98	6	26	2.74	14	34
140	35	38	40	42	0.60	0.77	1.65	1.32	0	22	1.43	2	23	1.43	2	22	1.26	0	22	0.22	0	0	1.26	0	21
140	105	38	40	42	0.99	2.69	31.41	2.75	6	27	3.46	12	33	3.74	14	34	2.63	6	27	2.75	6	27	3.57	12	33
150	37	40	42	44	2.04	0.49	2.64	0.49	0	4	1.81	2	25	2.37	12	33	0.49	0	4	0.22	0	0	2.19	8	31
150	113	40	42	44	1.98	4.12	5.55	3.13	6	26	5.44	20	40	7.41	38	57	3.08	6	26	3.30	6	26	4.78	18	37
160	40	43	45	47	2.25	1.81	2.03	2.03	0	24	2.36	2	27	2.26	2	24	1.93	0	24	2.20	2	27	0.38	0	1
160	120	43	45	47	2.42	3.68	4.06	4.12	6	26	3.96	2	22	6.42	24	44	3.84	6	26	0.82	0	0	4.12	6	26
170	42	45	47	49	1.04	2.20	2.14	2.36	0	23	2.85	2	29	2.80	2	26	2.20	0	23	2.59	2	29	0.33	0	0
170	128	45	47	49	1.70	22.13	2.74	5.05	6	29	7.75	16	40	5.43	6	29	4.83	6	29	7.03	16	40	5.28	6	29
180	45	48	50	52	0.77	2.30	1.48	0.33	0	0	3.51	2	28	3.35	2	26	0.38	0	0	0.39	0	0	0.55	0	1
180	135	48	50	52	3.46	6.98	10.05	6.81	6	33	7.52	12	35	9.56	20	46	6.43	6	33	7.20	12	35	7.19	6	33
190	47	50	52	54	1.37	3.25	3.29	4.06	0	31	3.90	2	28	4.56	4	32	3.85	0	31	3.62	2	28	0.72	0	2
190	143	50	52	54	5.55	14.39	3.74	7.31	6	31	8.84	12	40	7.86	6	31	6.81	6	31	8.40	12	40	7.52	6	31
200	50	53	55	57	2.69	3.57	1.92	4.83	0	31	5.06	2	32	5.21	2	31	4.51	0	31	4.61	0	31	0.77	0	1
200	150	53	55	57	11.81	8.40	18.78	10.6	8	41	11.8	14	45	11.3	14	38	10.27	8	41	10.99	14	45	9.56	6	31

Table 3b: Computational results for problems with $\tau = 40\%$

					Modified Penalty			Original B&B Algorithm									Modified B&B Algorithm								
n_1+n_2	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.05	0.11	0.05	0.00	2	4	0.05	2	5	0.06	2	3	0.00	0	0	0.00	0	4	0.00	0	0
50	38	15	17	19	0.05	0.17	1.16	0.11	12	17	0.55	142	147	0.83	164	178	0.06	6	11	0.16	16	21	0.93	164	178
60	15	18	20	22	0.06	0.05	0.05	0.00	0	0	0.05	2	9	0.06	2	6	0.00	0	0	0.06	0	8	0.00	0	1
60	45	18	20	22	0.16	0.55	0.44	0.11	6	12	0.27	22	27	1.82	288	294	0.11	6	12	0.17	10	15	0.27	26	32
70	17	20	22	24	0.06	0.11	0.17	0.00	0	0	0.06	2	11	0.11	2	8	0.05	0	0	0.11	0	10	0.00	0	0
70	53	20	22	24	0.22	0.44	0.61	0.22	6	15	0.22	6	13	0.66	36	43	0.22	6	15	0.22	6	13	0.55	30	37
80	20	23	25	27	0.28	0.11	0.28	0.11	2	8	0.05	0	1	0.11	2	8	0.11	0	7	0.00	0	0	0.11	0	2
80	60	23	25	27	0.33	0.50	1.87	0.33	6	14	2.75	202	208	0.99	42	50	0.33	6	14	0.38	10	16	0.99	42	50
90	22	25	27	29	0.11	0.16	0.28	0.00	0	0	0.22	2	11	0.22	2	10	0.05	0	0	0.16	2	11	0.11	0	1
90	68	25	27	29	0.55	0.66	8.52	0.60	12	21	0.60	12	20	1.48	38	46	0.44	6	15	0.50	6	14	1.54	38	46
100	25	28	20	32	0.44	0.61	0.55	0.22	0	8	0.33	2	11	0.38	2	13	0.28	0	8	0.33	2	11	0.11	0	0
100	75	28	20	32	0.54	2.31	6.10	0.88	12	20	1.43	22	29	6.81	208	215	0.61	6	14	1.26	20	27	6.48	208	215
110	27	30	32	34	0.49	0.50	0.33	0.38	0	11	0.11	0	0	0.44	2	11	0.33	0	11	0.11	0	0	0.11	0	0
110	83	30	32	34	0.71	5.00	3.74	0.99	6	15	6.43	156	166	7.14	132	141	0.94	6	15	3.57	62	72	2.25	28	37
120	30	33	35	37	0.93	0.93	0.66	0.55	2	11	0.66	2	14	0.66	2	13	0.11	0	0	0.22	0	3	0.61	0	12
120	90	33	35	37	0.93	1.59	1.97	1.32	6	15	1.81	12	21	1.37	6	14	1.21	6	15	1.76	12	21	1.38	6	14
130	32	35	37	39	1.26	1.54	11.97	0.72	0	14	0.77	2	14	1.10	10	21	0.66	0	14	0.71	0	13	0.88	6	17
130	98	35	37	39	1.48	1.59	1.81	1.87	6	18	1.81	6	17	1.81	8	17	1.81	6	18	1.76	6	17	1.82	6	16
140	35	38	40	42	0.60	0.93	3.84	1.04	2	17	0.99	2	15	1.48	8	23	0.99	0	16	0.94	2	15	1.32	6	20
140	105	38	40	42	2.58	2.20	7.03	2.36	6	19	3.29	14	26	24.0	290	302	2.25	6	19	2.97	14	26	2.91	12	24
150	37	40	42	44	0.44	0.71	2.37	0.16	0	0	1.37	2	18	1.54	4	19	0.22	0	0	1.32	0	17	1.48	4	19
150	113	40	42	44	2.47	9.72	6.97	3.07	6	20	4.72	20	35	2.97	6	19	2.97	6	20	3.73	14	27	2.86	6	19
160	40	43	45	47	3.40	1.42	2.31	2.09	10	21	1.71	2	19	1.54	2	15	2.03	10	21	0.33	0	0	0.38	0	0
160	120	43	45	47	9.40	32.19	2.96	4.23	6	25	4.06	6	21	3.79	6	19	4.06	6	25	3.90	6	21	3.79	6	19
170	42	45	47	49	1.26	7.80	3.29	0.49	0	2	2.03	2	14	2.91	14	25	0.50	0	2	0.94	0	4	2.86	14	25
170	128	45	47	49	3.74	23.78	13.79	4.67	6	24	7.31	18	36	9.39	36	48	4.44	6	24	6.97	18	36	28.45	226	241
180	45	48	50	52	0.83	2.03	2.36	0.33	0	0	2.64	2	21	2.14	2	14	0.38	0	0	2.47	0	20	0.44	0	0
180	135	48	50	52	2.03	10.28	5.83	6.37	6	26	7.69	16	32	9.23	20	39	6.10	6	26	6.26	10	26	6.64	6	26
190	47	50	52	54	0.93	3.52	2.52	0.33	0	0	3.07	2	20	4.06	12	27	0.44	0	0	2.80	0	19	3.85	12	27
190	143	50	52	54	5.11	9.89	6.76	6.32	6	22	8.73	12	31	15.0	44	60	6.04	6	22	8.40	12	31	7.63	12	28
200	50	53	55	57	9.39	1.70	5.66	3.29	2	19	3.79	2	22	3.62	2	19	2.97	0	18	0.55	0	0	12.79	64	85
200	150	53	55	57	2.81	18.45	13.95	7.80	6	23	10.6	14	31	12.9	24	40	7.19	6	23	10.27	14	31	7.75	6	22

Table 3c: Computational results for problems with $\tau = 60\%$

					Modified Penalty			Original B&B Algorithm									Modified B&B Algorithm								
n_1+n_2	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.06	0.05	0.06	0.00	0	0	0.05	10	11	0.06	12	14	0.00	0	0	0.06	10	11	0.05	12	14
50	38	15	17	19	0.22	0.11	0.17	0.11	18	22	0.05	8	10	0.05	2	6	0.16	18	22	0.06	6	9	0.11	2	6
60	15	18	20	22	0.06	0.06	0.28	0.05	0	0	0.06	2	8	0.06	2	5	0.00	0	0	0.05	0	7	0.00	2	5
60	45	18	20	22	0.88	0.88	1.10	0.82	138	142	0.61	78	82	1.43	228	232	0.83	138	142	0.71	78	82	1.26	190	194
70	17	20	22	24	0.06	0.11	0.50	0.05	0	0	0.05	2	6	0.06	2	5	0.00	0	0	0.05	0	5	0.05	0	4
70	53	20	22	24	0.38	0.28	2.91	0.17	6	11	0.33	18	21	5.11	482	486	0.22	6	11	0.22	6	9	2.75	274	278
80	20	23	25	27	0.05	0.16	0.16	0.06	0	0	0.06	0	0	0.11	2	7	0.05	0	0	0.00	0	0	0.06	0	0
80	60	23	25	27	0.38	1.10	0.44	0.28	6	8	0.50	18	21	0.33	8	10	0.22	6	8	0.27	6	9	0.27	8	10
90	22	25	27	29	0.17	0.32	1.48	0.00	0	0	0.16	2	8	0.22	6	11	0.05	0	0	0.17	2	8	0.28	6	11
90	68	25	27	29	0.99	1.21	18.23	0.39	6	10	0.50	8	12	15.4	826	830	0.38	6	10	0.55	8	12	15.11	754	758
100	25	28	20	32	0.33	0.66	1.10	0.22	0	7	0.22	2	8	0.27	2	8	0.22	0	7	0.22	0	7	0.27	2	8
100	75	28	20	32	0.88	0.82	7.14	0.72	6	11	0.55	6	8	1.87	32	37	0.60	6	11	0.61	6	8	1.09	14	19
110	27	30	32	34	0.27	0.33	0.38	0.06	0	0	0.06	0	0	0.22	2	5	0.11	0	0	0.11	0	0	0.11	0	0
110	83	30	32	34	0.99	1.27	9.78	0.88	6	12	5.71	122	126	6.92	140	146	0.82	6	12	0.88	6	11	5.77	122	128
120	30	33	35	37	0.22	0.38	0.61	0.06	0	0	0.11	0	1	0.55	2	11	0.16	0	0	0.16	0	0	0.16	0	1
120	90	33	35	37	1.21	2.69	2.75	1.04	6	10	1.59	12	17	3.30	34	38	1.05	6	10	1.54	12	17	1.70	16	20
130	32	35	37	39	0.33	0.77	1.54	0.11	0	0	0.11	0	0	0.66	6	12	0.17	0	0	0.17	0	0	0.55	2	8
130	98	35	37	39	1.53	2.25	17.46	1.48	6	13	1.59	6	12	19.5	238	245	1.49	6	13	1.48	6	12	13.13	178	185
140	35	38	40	42	0.83	2.09	11.98	0.94	8	15	10.7	170	177	0.66	2	8	1.04	8	15	6.48	100	107	0.27	0	0
140	105	38	40	42	1.43	7.36	5.55	1.98	6	14	5.87	34	43	41.8	600	608	1.92	6	14	5.66	34	43	3.02	16	24
150	37	40	42	44	0.66	0.77	3.74	0.16	0	0	1.04	2	12	1.15	6	13	0.17	0	0	0.93	0	11	1.04	4	12
150	113	40	42	44	1.65	2.91	5.33	2.31	6	13	2.69	6	13	6.54	36	43	2.20	6	13	2.42	6	13	4.50	24	31
160	40	43	45	47	1.43	1.32	3.29	0.99	0	9	0.93	2	8	0.99	2	8	0.93	0	9	0.27	0	0	0.88	0	7
160	120	43	45	47	2.14	17.41	2.36	2.81	6	12	3.24	6	15	2.86	6	12	2.63	6	12	2.97	6	15	2.86	6	12
170	42	45	47	49	2.03	2.69	16.20	9.78	98	104	1.48	2	11	7.91	70	76	1.70	8	14	0.66	0	3	5.11	42	48
170	128	45	47	49	7.96	8.57	5.27	8.84	36	45	5.38	12	21	8.62	22	31	8.30	36	45	5.38	12	21	5.33	12	21
180	45	48	50	52	0.82	2.58	3.13	0.33	0	0	1.81	2	12	1.76	2	11	0.38	0	0	1.65	0	11	1.59	0	10
180	135	48	50	52	2.08	10.98	8.63	4.73	6	16	5.38	10	18	11.8	34	43	4.56	6	16	5.00	10	18	5.98	12	21
190	47	50	52	54	0.93	1.70	2.31	0.33	0	0	0.55	0	1	3.13	12	19	0.50	0	0	0.61	0	1	3.18	12	19
190	143	50	52	54	2.41	9.89	14.50	5.27	6	16	7.47	12	24	12.7	34	44	4.95	6	16	7.08	12	24	6.87	12	22
200	50	53	55	57	3.51	10.99	2.25	2.47	0	13	13.2	24	36	2.75	2	14	2.64	0	13	6.65	6	18	2.63	2	14
200	150	53	55	57	2.75	10.38	9.17	6.43	6	17	8.51	12	23	14.3	34	45	6.09	6	17	8.35	12	23	8.08	12	23

Table 3d: Computational results for problems with $\tau = 80\%$