AMO - Advanced Modeling and Optimization, Volume 13, Number 2, 2011

On the use of a tabu pivoting technique for solving the linear complementarity problem

Maria João Alves

Fac. Economia, Univ. Coimbra /INESC Coimbra, Portugal

mjalves@fe.uc.pt

Joaquim J. Júdice

Dep. Matemática, Univ. Coimbra /Inst. Telecomunicações, Coimbra, Portugal

joaquim.judice@co.it.pt

Abstract

This paper describes a pivoting heuristic based on tabu search and its integration into an enumerative framework for solving the Linear Complementarity Problem (LCP). The tabu pivoting heuristic works with basic solutions and performs pivot operations guided by two indicators, one concerned with the satisfaction of the complementarity conditions and the other with the feasibility of the solution. It incorporates the concept of tabu search employing a strategy that avoids the repetition of recent moves. The heuristic ends when a solution to the LCP is found or after a specified number of iterations. In the latter case, an enumerative algorithm is applied which integrates the tabu pivoting heuristic within a branching framework. Computational experience on test problems is reported to highlight the efficiency of the proposed methodology for solving the LCP.

Keywords: Linear complementarity problem, Tabu search, Pivoting techniques, Enumerative algorithms

1 Introduction

The Linear Complementarity Problem (LCP) consists of finding vectors $z \in \mathrm{R}^n$ and $w \in \mathrm{R}^n$ such that

$$w = Mz + q \tag{1}$$

$$z \geq 0, w \geq 0 \tag{2}$$

$$z^T w = 0 \tag{3}$$

where q is an n-dimensional vector and M is a square matrix of order n.

This problem has several applications, which include economic equilibrium analysis, game theory, portfolio selection and structural analysis, among others. A number of important optimization problems can be solved by finding a solution of its associated LCP or one of its generalizations [Murty, 1988; Cottle et al., 2009; Júdice, 1994].

It is well-known that the LCP is NP-hard, although it is polynomially solvable for some classes of matrices M, such as Positive-Semi-Definite (PSD) matrices [Murty, 1988]. One of the earliest methods, and probably the most famous procedure for solving the LCP, is the Lemke's algorithm [Lemke, 1968], which can solve some classes of LCP. There are other several direct and iterative algorithms, but these procedures also face the limitation of only being able to process the LCP when M has some special properties. Therefore, enumerative algorithms are the only ones that can solve the LCP without imposing conditions on the class of matrix M.

There are different alternative formulations of the LCP that have been exploited in the design of procedures for its solution. The reformulation of the LCP as a mixed integer linear program has led to an enumerative algorithm based on the so-called reformulation-linearization technique by [Sherali et al., 1998]. Reformulating the LCP as a nonconvex quadratic program (the minimization of (3) subject to (1) and (2)) has been used by [Al-Khayyal, 1987] and [Júdice et al., 2002] to develop enumerative algorithms for the LCP. [Júdice et al., 2002] discusses an enumerative algorithm (EMRG) for finding a global minimum of the quadratic formulation of the LCP, which uses a Modified Reduced-Gradient (MRG) method [Al-Khayyal, 1987] in each node generated by the procedure. The authors also present an enumerative sequential algorithm based on the bilinear formulation of the LCP. The results reported using subset sum problems (a special case of the knapsack problem) show that the enumerative algorithm based on the quadratic formulation performed in general better than the enumerative algorithm based on the bilinear formulation. An alternative enumerative algorithm (EAsET) for finding a global minimum of the same quadratic formulation of the LCP was tested by [Ribeiro, 2004] using the same instances. EAsET is similar to EMRG but uses an Active-Set method in each node to find a stationary point of the objective function on a set defined by the corresponding linear constraints (instead of an MRG).

The LCP can also be solved by exploiting its reformulation as a Mathematical Program with Equilibrium (complementarity) Constraints (MPEC) [Júdice et al., 2002]. A branch-and-bound algorithm (EMPEC) for solving this special MPEC has been proposed in [Júdice et al., 2006], which incorporates disjunctive

cuts for computing lower bounds and uses a complementarity active-set algorithm for computing upper bounds. Computational results with this EMPEC method for solving some LCP instances are reported in this latter paper. These results, as well as those of the EMRG and EAsET algorithms, are used in the present paper for comparison purposes.

We propose herein a procedure to solve the LCP that incorporates the concept of *tabu search*.

In the last decades many metaheuristic algorithms have been developed as viable alternatives for solving optimization problems. Tabu search is one such metaheuristic approach, which was firstly proposed by [Glover, 1986]. As described by Glover, "*the approach [tabu search] undertakes to transcend local optimality by a strategy of forbidding (or, more broadly, penalizing) certain moves. The purpose of classing a move forbidden – i.e. tabu – is chiefly to prevent cycling*". Although the LCP is not intrinsically an optimization problem, the penalization of recent moves fits the same primary purpose of avoiding revisiting solutions during the search.

Tabu search has been widely used on combinatorial optimization problems, for which several problem-specific variants of the basic algorithm have been developed. Many of the applications in the literature involve integer programming problems, scheduling, location, routing, travelling salesman and related problems. Fewer tabu search procedures have been developed for continuous optimization and in particular for global optimization problems. Some examples of such procedures can be found in [Chelouah and Siarry 2000; Gendreau et al., 1996; Kovacevic-Vujcic and Cangalovic, 1999]; Lan et al., 2007; Rajesh et al., 2003] and references therein.

Tabu search algorithms using extreme points have been successfully applied to zero-one integer and mixed-integer programming. [Aboudi and Jörnsten, 1994] and [Løkketangen et al., 1994] explore the use of the tabu search principle within the Pivot and Complement heuristic by [Balas and Martin, 1980] for solving general zero-one integer programs. [Løkketangen and Glover, 1995] and [Løkketangen and Glover, 1998] designed procedures to solve general zero-one mixed integer problems which also combine tabu search mechanisms with adjacent extreme point search. The principle of using a priority list combined with tabu restrictions on pivot operations has inspired the pivoting heuristic for the LCP to be presented herein.

The tabu pivoting heuristic to be introduced in this paper operates with basic solutions satisfying (1) and performs pivot operations for reaching both complementarity (3) and feasibility (2). It uses two functions to guide the search, measuring the numbers of violated complementary conditions and of violated feasibility constraints. The relative values of these functions define the direction of search at each iteration. A tabu data structure is used for the selection of

an entering nonbasic variable in order to prevent cycling and avoid revisiting solutions. The heuristic is executed until a solution to the LCP is found or a maximum number of iterations is attained. If the heuristic cannot reach a solution of the LCP in the predefined number of iterations, an enumerative algorithm is applied, which incorporates the tabu pivoting heuristic within a branching process.

This Branch and Tabu Pivoting (BTP) algorithm has been tested on randomly generated LCPs of different types. The computational experiments have also included some LCP instances resulting from subset sum problems that were previously tested in other studies. Therefore, a comparison with the EMRG and the EAset algorithms [Júdice et al., 2002; Ribeiro, 2004] and with the EMPEC (enumerative algorithm for MPEC problems) by [Júdice et al., 2006] is presented, leading to the conclusion that the new BTP algorithm is a competitive technique for solving these NP-hard LCPs.

The rest of the paper is organized as follows. Some fundamental concepts of the LCP and the notation are introduced in section 2. In section 3 the tabu pivoting heuristic is described and different procedures for obtaining the initial solution are discussed in section 4. The BTP algorithm is proposed in section 5. Results from the computational experience are reported in section 6 and some concluding remarks are presented in section 7.

2 Basic Concepts for the LCP

It follows from the definition (1)-(3) that the LCP contains the linear constraints, which constitute the so-called feasible set S=$\{(z,w)$: $w = Mz + q$, $z \geq 0$, $w \geq 0\}$, and the nonlinear complementarity constraint $z^T w = 0$. This latter constraint requires the *complementarity conditions* $z_i w_i$ =0 to be satisfied for each pair (z_i, w_i), i=1,...,n, i.e., at least one variable of each pair must be equal to zero.

A solution (z, w) is said to be *feasible* if it belongs to the feasible set S. On the other hand, it is called *complementary* if it satisfies

$$w = Mz + q$$

$$z_i w_i = 0 \quad i = 1, ..., n$$

Due to this definition, for each solution of (1), z_i, and w_i are called *complementary* variables and z_i (w_i) is said to be the *complementary* variable of w_i (z_i), independently of the complementarity conditions to hold or not. Furthermore (z, w) is a solution to the LCP if and only if it is feasible and complementary.

Any basic feasible solution of the LCP has n basic variables and n nonbasic variables, as a direct consequence of the problem structure. If a basic feasible solution is not complementary, it contains at least one pair of variables (z_i, w_i)

violating the complementarity condition, which is called a *violating* pair, and at least another pair (z_j, w_j) such that z_j=0, w_j=0 and both nonbasic, which constitute a *free* pair.

3 The tabu pivoting heuristic algorithm

The tabu pivoting heuristic algorithm aims at finding a solution of the LCP, i.e. a solution that is simultaneously *feasible* and *complementary*. It works with basic solutions satisfying $w = Mz + q$ and performs single pivot operations, alternating some oriented to the satisfaction of the complementarity conditions with others designed to force the feasibility of constraints. The choice for the operation is done according to the relative values of two functions, which are indicators to what extent these requirements are not fulfilled by the current basic solution.

The algorithm requires an initial basic feasible solution to start with. This topic is discussed in the next section. If the initial basic feasible solution is not complementary, the tabu pivoting algorithm is applied. If a basic solution has at least a negative basic variable, then it is infeasible and its indicator of infeasibility, designated by *NoFeasib*, is defined as the number of variables with negative value. One objective of the tabu pivoting heuristic is to minimize this function to zero in order to obtain a feasible solution. On the other hand, for each basic solution we can define *NoCompl* as the number of pairs of variables that violate complementarity (*NoCompl*=0 if a complementarity solution is at hand). The other objective of the heuristic is to minimize *NoCompl* to zero, in order to obtain a complementary solution.

In the initial solution, *NoFeasib*=0 and *NoCompl* is strictly positive, unless it is a solution of the LCP. The heuristic alternates two types of pivot operations, according to the relative values of *NoFeasib* and *NoCompl*, and stops when *NoFeasib*=*NoCompl*=0 (that is, a solution of the LCP has been found) or a predefined number of iterations (*MaxIter*) has already been performed.

If *NoFeasib* $\geq$ *NoCompl*, the pivot operation should be performed towards reaching feasibility. A basic variable with negative value is chosen to leave the basis. In the selection of the nonbasic variable that enters into the basis the first priority is given to the complementary variable of the leaving basic variable. If this is not possible, a *tabu classification* system (consisting in assigning penalties to recent moves) is used to select the entering variable among the candidates.

If *NoCompl* $>$ *NoFeasib*, the pivot operation should be performed towards reaching complementarity. In this situation, a move from the current basic solution to an adjacent one involves exchanging a *violating* basic variable with a *free* nonbasic variable. In the selection of the *violating* and the *free* variables, an exchange that benefits (or does not destroy) feasibility is privileged and the

tabu classification system is used to select the entering *free* variable when there are several candidates.

The *tabu classification* system penalizes recent moves and avoids exchanging variables that have been involved in recent operations. This operates like a classical tabu memory, although there are no absolute forbidden moves, but rather highly penalized variables that are selected only if there are no alternative variables with zero or lower penalties. The tabu data structure is a vector of order $2n$, which registers a tabu classification (penalty) for the variables z_i, i=1,...,n, and w_i, i=1,...,n. The tabu vector is initialized with zeros, indicating that no moves are penalized. Whenever a variable leaves the basis, its tabu classification becomes equal to a *tabu tenure* (TT), the maximum penalty. In addition, all the other positive elements of the tabu vector are decreased by one. These penalties are used for the selection of the entering nonbasic variable, as the variable with lowest tabu classification is chosen among the candidate list.

The type of tabu strategy employed by this heuristic has been so-called '*aspiration by default*' in the Tabu Search literature, which means that if a tabu move has to be performed then it should be selected the "oldest" one in the list (see, e.g., [Glover and Laguna, 1993; Michalewicz and Fogel, 2004]).

3.1 Tabu Pivoting Heuristic Algorithm

Step 0 – Find an initial basic feasible solution. If the problem is infeasible, stop: the LCP has no solution. Otherwise *NoFeasib*=0 and compute *NoCompl* for this solution. Initialize the iteration counter, $Iter = 0$, and the tabu vector with zeros.

While ($NoCompl > 0$ or $NoFeasib > 0$) and ($Iter < Maxiter$) **Do**
Step 1 − Choice of the leaving and entering variables.

(I) If $NoCompl > NoFeasib$, then:

(i) Choose a *violating* leaving basic variable and a *free* entering nonbasic variable, according to the following priorities:

- the current solution is feasible and an exchange is possible between a *violating* and a *free* variables preserving feasibility;
- the solution is infeasible and there exists a negative *violating* variable and a *free* nonbasic variable with a negative entry in the row of the simplex tableau associated with this violating variable.

(ii) If no pair of leaving and entering variables is found in (i), let L be the set of pairs of *violating* and *free* variables that can be exchanged by a pivot operation. If $L \neq \emptyset$, construct a priority list SL of pairs of L such that, for each *violating* variable with a

positive (negative) value, there is a positive (negative) entry of the simplex tableau defined by the row of this variable and a column of a *free* variable. The entering variable is a *free* variable of a pair of SL, or of L if $SL=\emptyset$, associated to the smallest penalty of the tabu classification. The leaving variable is the *violating* variable of the corresponding pair.

(*iii*) If $L = \emptyset$, the entering is a nonbasic variable that has the smallest penalty of the tabu classification and can be exchanged with a *violating* variable by a pivot operation. The leaving is this latter variable.

(II) Else ($NoFeasib \geq NoCompl$):

(*i*) Choose the leaving variable x_i (z_i *or* w_i) as the basic variable with the most negative value.

(*ii*) Choose the complementary of x_i as the entering variable, provided it is nonbasic and there is a negative entry in the row of the current simplex tableau associated with x_i.

(*iii*) Otherwise find the entering variable as the one with smallest penalty in the current tabu classification among the nonbasic variables with negative entries in the row associated with x_i. For tie breaking, give priority to *free* variables if they exist.

Step 2 – Compute the new basic solution by performing a pivot operation with the pivot defined by the leaving and entering variables. Update the tabu vector, calculate *NoCompl* and *NoFeasib*, and set $Iter \leftarrow Iter + 1$.

End While.

A first example illustrating the application of this algorithm is presented below.

Example 1

Consider the LCP with $n = 5$ and defined by the following matrix M and vector q:

$$M = \begin{bmatrix} 9 & 4 & 9 & 3 & 3 \\ 7 & 2 & 10 & 3 & 4 \\ 5 & 8 & 6 & 3 & 4 \\ 2 & 7 & 3 & 8 & 8 \\ 7 & 8 & 1 & 5 & 8 \end{bmatrix} \quad q = \begin{bmatrix} 3 \\ -9 \\ 2 \\ 10 \\ -5 \end{bmatrix}$$

Since M is a positive matrix, then the LCP has a solution for each vector q (Murty 1988) and, in particular, for this given q. Furthermore, Lemke's algorithm (Lemke 1968) is able to process this LCP (Murty 1988). Despite this, we have chosen this problem to illustrate the algorithm.

A basic feasible solution is first computed and is given below.

$z^T =$	(0.650794	0	0.444444	0	0)
$w^T =$	(12.85714	0	7.920635	12.63492	0)
	violating	*free*	*violating*		*free*

The associated simplex tableau is

	$\mathbf{w}_5$	$\mathbf{z}_5$	$\mathbf{w}_2$	$\mathbf{z}_2$	z_4	Xb
$\mathbf{z}_1$	-0.15873	*1.206349*	*0.015873*	*1.238095*	0.746032	0.650794
$\mathbf{w}_1$	-0.42857	3.857143	-0.85714	1.142857	1.714286	12.85714
$\mathbf{z}_3$	***0.11111***	-0.44444	-0.11111	-0.66667	-0.22222	0.444444
$\mathbf{w}_3$	-0.12698	-0.63492	-0.5873	-5.80952	-0.60317	7.920635
w_4	0.015873	-6.92063	-0.30159	-6.52381	-7.1746	12.63492

There exist two *violating* pairs and two *free* pairs (*NoCompl* =2) and *NoFeasib*=0, as the solution is feasible. The algorithm should move towards complementarity and there are four possible moves that preserve feasibility (pivots in italics in the tableau). The first nonbasic variable is chosen since all candidates have no tabu status. The basic variable z_3 is replaced by the nonbasic w_5 and a predefined *tabu tenure* is assigned to the tabu classification of the leaving basic variable z_3. After performing the pivot operation, the following simplex tableau is obtained.

	z_3	z_5	$\mathbf{w}_2$	$\mathbf{z}_2$	z_4	Xb
$\mathbf{z}_1$	1.428571	0.571429	-0.14286	***0.285714***	0.428571	1.285714
$\mathbf{w}_1$	3.857143	2.142857	-1.28571	-1.42857	0.857143	14.57143
w_5	9	-4	-1	-6	-2	4
w_3	1.142857	-1.14286	-0.71429	-6.57143	-0.85714	8.428571
w_4	-0.14286	-6.85714	-0.28571	-6.42857	-7.14286	12.57143

The new solution is feasible (*NoFeasib*=0), but it is not complementary (*NoCompl*=1). Now there is only one pivot operation that preserves feasibility. Hence, a pivot operation exchanging z_1 with z_2 is performed. The following solution of the LCP is found after this operation.

$z^T =$	(0	4.5	0	0	0)
$w^T =$	(21	0	38	41.5	31)

4 Finding an initial basic feasible solution

The tabu pivoting heuristic requires a basic feasible solution to start with. This initial solution can be obtained by solving a dual feasible linear program with

the feasible set S. For instance, the following program can be used for this purpose:

$$min \quad e^T w$$

$$s.t. \quad (z, w) \in S$$

where $e \in \mathrm{R}^n$ is a vector of ones. We denote this program by (P_1). Note that there is no particular reason to consider the objective function of (P_1) and another alternative is to minimize $e^T z$ over S. We denote this latter program by (P_2). More advanced procedures may be designed to find an initial basic feasible solution, which are discussed below.

4.1 Modified reduced-gradient algorithm

A modified reduced-gradient (MRG) algorithm [Al-Khayyal, 1987] searches for a so-called *local start minimum* of the function $g(z,w) = z^T w$ on the set S, which is an extreme point $(\bar{z}, \bar{w})$ of S satisfying $g(\bar{z}, \bar{w}) \leq g(z,w)$ for all extreme points $(z,w) \in S$ adjacent to $(\bar{z}, \bar{w})$.

In order to describe the MRG algorithm, let $(\bar{z}, \bar{w})$ be an extreme point of S corresponding to a basic feasible solution with basis B. If (z,w) is an adjacent extreme point, then $(z,w) = (\bar{z}, \bar{w}) + \mu\ (d_z,\ d_w)$, where μ is the so-called maximum stepsize used in the simplex method and $d=(d_z,\ d_w)$ is a feasible direction. The vector d can be defined in terms of the basis matrix B and of the columns of the matrix M or of the identity matrix I. Let F and T be the index sets of the basic and nonbasic variables respectively and let s be the index of the entering nonbasic variable that is increased from zero to generate the adjacent basic feasible solution. Then the feasible direction d is given by

$$d_s = 1$$
$$d_j = 0 \text{ for all } j \in T - \{s\}$$

$$d_F = \begin{cases} -B^{-1} M_{\bullet s} & \text{if } s \text{ is a column of a } z_i \text{ variable} \\ B^{-1} I_{\bullet s} & \text{if } s \text{ is a column of a } w_i \text{ variable} \end{cases}$$

where $M_{\bullet s}$ and $I_{\bullet s}$ are the s^{th} column of the matrices M and I respectively.

There is a decrease in g produced by a movement to a new adjacent extreme point ($\mu > 0$) if and only if $\bar{z}^T d_w + \bar{w}^T d_z + \mu d_z^T d_w < 0$. In each iteration, the MRG algorithm searches for a feasible descent direction d and a positive stepsize μ satisfying this condition. If such d and μ exist, the algorithm finds a new adjacent extreme point with a decrease of the objective function g. Otherwise the algorithm terminates with a local star minimum, provided that all the stepsizes are positive.

The solution returned by the MRG algorithm is a basic feasible solution. Therefore, it is either a solution to the LCP or it is noncomplementary and can be used as the initial solution for the tabu pivoting heuristic.

4.2 Procedure based on Lemke's algorithm

Lemke's algorithm [Lemke, 1968] is probably the most famous procedure for solving the LCP. In this algorithm, a nonnegative vector p and an artificial variable z_0 are added to the LCP in order to get the following Generalized Linear Complementarity Problem (GLCP):

$$w = q + Mz + z_0 p$$

$$z \geq 0, \quad w \geq 0, \quad z_0 \geq 0$$

$$z^T w = 0$$

We assume that the vector q has at least a negative component, as $z = 0$ is a solution to the LCP otherwise. Then there exists a value $\bar{z}_0 > 0$ such that $w = q + \bar{z}_0 p \geq 0$. This value $\bar{z}_0$ can be obtained by performing a pivot operation that replaces a basic variable w_r by z_0, where r satisfies

$$-\frac{q_r}{p_r} = \max\left\{-\frac{q_i}{p_i} : q_i < 0\right\}$$

After performing this operation, a basic feasible solution of the GLCP is at hand.

In each iteration of Lemke's algorithm a pivot operation is performed. The entering variable is the nonbasic variable whose complementary variable has left the basis in the previous iteration. This rule keeps the complementarity constraint satisfied throughout the algorithm. The leaving basic variable is selected by the simple minimum quotient rule similar to the one that is used by the simplex method for linear programming.

The algorithm terminates in a solution of the LCP, when z_0=0, or in an unbounded ray. This last form of termination may have no meaning at all. However, there are some cases where such termination cannot occur or only occurs when the feasible set of the LCP is empty (Murty 1988; Cottle et al. 2009).

If Lemke's algorithm terminates in a ray and the LCP is feasible, the last solution cannot be directly used as the initial solution for the tabu pivoting heuristic, because it is infeasible to the LCP. Therefore, a post-processing phase should be applied in order to get a basic feasible solution of the LCP. This phase consists of minimizing z_0 in the feasible set of the GLCP problem starting with the final solution given by Lemke's algorithm. We have introduced the following modification in the rule of the simplex method for selecting the nonbasic variable to enter into the basis: unless there is an entering variable that leads z_0 to leave the basis (which is the first choice), the procedure attempts to choose a nonbasic variable that decreases z_0 and whose complementary variable is also nonbasic. If there is no such candidate, the procedure selects the entering variable that leads to the largest decrease of z_0.

Example 2

Consider again the LCP of example 1. In that example, the initial basic feasible solution has been obtained by solving (P_1). Now the modified reduced-gradient (*MRG*) algorithm is applied and finds a basic feasible solution given by the tableau below.

	w_5	z_1	$\mathbf{w}_2$	$\mathbf{z}_2$	z_4	*Xb*
z_5	-0.13158	0.82895	0.01316	1.02632	0.61842	0.53947
w_1	0.07895	-3.19737	-0.90789	-2.81579	-0.67105	10.77632
$\mathbf{z}_3$	0.05263	0.36842	***-0.10526***	*-0.21053*	0.05263	0.68421
$\mathbf{w}_3$	-0.21053	0.52632	*-0.57895*	*-5.15789*	-0.21053	8.26316
w_4	-0.89474	5.73684	-0.21053	0.57895	-2.89474	16.36842

Suppose that this solution is the initial basic solution for the tabu pivoting algorithm. Hence *NoCompl* =1 and *NoFeasib*=0 and a variable of the *violating* pair (z_3,w_3) leaves the basis by replacing it with a variable of the *free* pair (z_2, w_2). There is no pivot operation that preserves feasibility and a negative pivot must be chosen. So, z_3 is replaced by w_2 and the resulting solution is complementary but infeasible. The tabu vector, say t, is updated by assigning a predefined *tabu tenure* (TT) to the component of z_3 (the leaving basic variable). By choosing TT equal to 10, we get the following tabu vector.

$$t = \begin{array}{c} \begin{array}{cccccccccc} z_1 & z_2 & z_3 & z_4 & z_5 & w_1 & w_2 & w_3 & w_4 & w_5 \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

The simplex tableau of the new solution is given below.

	w_5	z_1	z_3	z_2	z_4	*Xb*
z_5	-0.125	0.875	0.125	1	0.625	0.625
w_1	-0.375	-6.375	-8.625	-1	-1.125	4.875
w_2	***-0.5***	-3.5	-9.5	2	-0.5	**-6.5**
w_3	-0.5	-1.5	-5.5	-4	-0.5	4.5
w_4	-1	5	-2	1	-3	15

Then *NoFeasib*=1 > *NoCompl* =0 and the leaving variable w_2 is the unique basic variable with a negative value. The candidates to enter into the basis are the nonbasic variables with negative coefficient in the row of w_2, i.e. all except z_2. The highest priority is given to the complementary variable of w_2 if it is a candidate, but unfortunately it is not. Hence, another entering variable has to be chosen causing the loss of complementarity. The tabu classification is first used to filter the set of candidate variables {w_5,z_1, z_3, z_4} leading to the exclusion of z_3 (note the worthwhile role of the tabu status in order to prevent

cycling). As three contenders remain, the first one, w_5, is chosen. The pivot operation exchanging w_2 with w_5 is performed yielding a noncomplementary feasible solution. The tabu vector is updated by assigning the value TT to the component of w_2 and the other nonzero components are decreased by 1, leading to the following vector.

	z_1	z_2	z_3	z_4	z_5	w_1	w_2	w_3	w_4	w_5
$t =$	[0	0	9	0	0	0	10	0	0	0]

The simplex tableau of the new solution is given below.

	$\mathbf{w}_2$	z_1	z_3	$\mathbf{z}_2$	z_4	Xb
$\mathbf{z}_5$	-0.25	1.75	2.5	***0.5***	0.75	2.25
w_1	-0.75	-3.75	-1.5	-2.5	-0.75	9.75
$\mathbf{w}_5$	-2	7	19	-4	1	13
w_3	-1	2	4	-6	0	11
w_4	-2	12	17	-3	-2	28

Then *NoCompl* =1 > *NoFeasib*=0 and the algorithm exchanges a basic variable of the *violating* pair (z_5,w_5) with a variable of the *free* pair (z_2, w_2). The pivot operation defined by z_5 and z_2 is performed, as it is the only one that preserves feasibility. The simplex tableau corresponding to the new solution is given below.

	w_2	z_1	z_3	z_5	z_4	Xb
z_2	-0.5	3.5	5	2	1.5	4.5
w_1	-2	5	11	5	3	21
w_5	-4	21	39	8	7	31
w_3	-4	23	34	12	9	38
w_4	-3.5	22.5	32	6	2.5	41.5

Since *NoCompl* =*NoFeasib* = 0, this is a solution of the LCP.

Examples 1 and 2 illustrate the solution of a LCP by the tabu pivoting algorithm using two different initial basic feasible solutions. In the first case, the initial solution has two violating pairs and the heuristic required 2 iterations to find a solution to the LCP. On the other hand, the initial solution of the second case has only one violating pair, but the heuristic required 3 iterations to find a solution to the LCP. Therefore, these examples illustrate that a better starting solution in terms of the indicator *NoCompl* may lead to a longer process to find a solution to the LCP.

As already mentioned, the tabu pivoting heuristic is allowed to execute a maximum number of iterations. If it attains this number without reaching a

solution to the LCP, then an enumerative algorithm is applied starting with the basic solution given by the heuristic. This algorithm is discussed in the next section.

5 The Branch and Tabu Pivoting Algorithm

The Branch and Tabu Pivoting (BTP) algorithm to be described in this section employs a search tree based on the dichotomy z_i =0 or w_i =0 that holds for each pair of complementary variables. A depth-first search scheme is designed according to the figure 1, where x_i denotes a variable z_i or w_i and $\bar{x}_i$ its complementary.

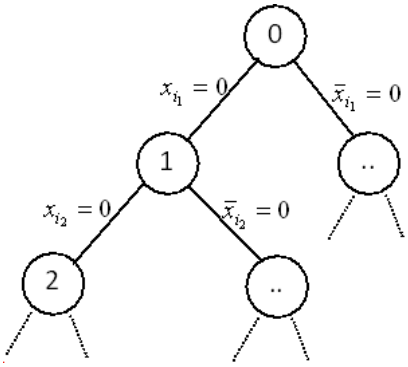

Figure 1: Branching schema

A sub-problem is associated to each node of the tree and is processed by the tabu pivoting heuristic for a maximum number of *MaxIter_child* iterations. This number should be smaller than the maximum number of iterations that is allowed in the root of the tree, *MaxIter_root*, as the algorithm tries to avoid the use of the enumerative method by allowing a larger number of iterations of the heuristic before branching. If the maximum number of iterations is attained with a complementary and infeasible solution, the heuristic continues until a noncomplementary solution (or a solution to the LCP) is found. The tabu vector used by the heuristic is inherited from parent to children.

If the tabu pivoting heuristic is applied to a node i, one of the following situations should occur:

1. A complementary and feasible solution to the sub-problem i is found and the BTP algorithm stops with a solution of the LCP.

2. The sub-problem i is infeasible (an infeasible solution has been obtained if there is no nonbasic variable with a negative coefficient in the row of a negative basic variable) and the node is fathomed.

3. The heuristic performs the maximum number of iterations and terminates with a noncomplementary solution.

In situation 3, the node is branched. The *left* child, node i+1, is created by selecting a *violating* basic variable x_v and a *free* nonbasic variable. This selection follows the same rules as in the tabu pivoting heuristic. The corresponding pivoting operation is performed, yielding the initial solution for the node (sub-problem) i+1. The variable x_v is fixed at 0 to get the sub-problem i+1. Next, the tabu pivoting heuristic is applied to the sub-problem i+1.

In situation 2, the algorithm backtracks by generating the *right* child (node) of the most recent node for which the second child had not been created. Two examples of this case are illustrated in figure 2.

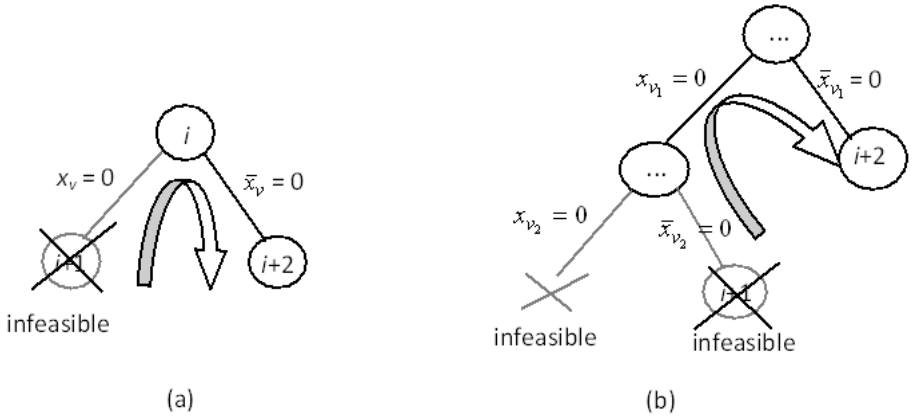

Figure 2: Second branch (*right* child)

The initial basic solution for the *right* child i+2 is also obtained from the final solution of its parent if such operation is possible. If x_v is the *violating* variable fixed at 0 in the *left* child of node i, then its complementary variable $\bar{x}_v$ is the selected variable to be fixed at 0 in the *right* child of node i. This variable leaves the basis by exchanging with a *free* nonbasic variable (chosen according to the rules of the tabu pivoting heuristic), producing the initial solution for the *right* node. The constraint $\bar{x}_v$=0 is imposed upon this problem and the tabu pivoting heuristic is applied to the resulting sub-problem.

Note that a pivoting operation between a *violating* basic variable and a *free* nonbasic variable may not be possible because of the number of variables that are fixed in the node. In this case, the heuristic starts by computing a basic feasible solution (if it exists) to the sub-problem, and the *tabu* vector is reinitialized rather than being inherited from the parent node.

An *additional test of infeasibility* has been further included in the BTP algorithm, which aims at accelerating the identification of infeasible sub-problems.

Basically, if the initial solution to the current sub-problem is infeasible and the tabu pivoting heuristic operates with infeasible solutions all through the iterations, then the linear programming solver is applied to that sub-problem (considering the current basis as initial basis) to check if this sub-problem is infeasible.

6 Computational experiments

In this section we report some computational experiments with the Branch and Tabu Pivoting (BTP) algorithm discussed in the previous section, which was implemented in Delphi for Windows. The experiments were performed on a computer Core 2 CPU 6700, 2.66GHz with 2GB of RAM.

The BTP algorithm was first tested (*experiment 1*) on a set of randomly generated problems belonging to different types. This experiment has considered the optimization problem (P_1) to compute the initial solution and has tested different values for the parameters of the algorithm. Although the problems used in *experiment 1* have been generated according to the rules used in other tests published in the literature, the problems involve the generation of random numbers, thus leading to different instances. Therefore, the results cannot be directly compared with the results of experiments of other authors.

A second experiment has been carried out (*experiment 2*) using a set of LCPs associated with subset sum problems that have already been tested with other algorithms. We present some comparison results for this set of problems.

Finally, a third experiment (*experiment 3*) has been performed using the problems of the previous experiments, 1 and 2, in order to analyse the impact of different initial solutions on the efficiency of the algorithm. The starting procedures presented in section 4 are tested in this experiment.

We start by describing the test problems.

6.1 Test problems

- *LCPs associated with subset sum problems*

Given a positive real number b and a positive vector $a \in \mathrm{R}^{\eta}$, the subset sum problem (a special case of the *knaspack problem*) can be defined as the problem of determining a vector $x \in \mathrm{R}^{\eta}$ such that $a^T x = b$ and $x_i \in \{0,1\}$ for all $i=1,...,\eta$. Three different formulations of this problem as an LCP are presented by [Fernandes et al., 2001], [Júdice et al., 2002] and [Kojima et al., 1991] among other references. We use two of them, which we denote by FORM1 and FORM2, adopting the terminology used by [Júdice et al., 2002] (these are called PROB5 and PROB6 in [Fernandes et al., 2001]). In the two formulations considered

herein, a subset sum problem of dimension η is equivalent to a LCP of dimension $n = \eta+2$.

FORM1 – the LCP problem is defined by

$$q = \begin{bmatrix} e \\ -b \\ b \end{bmatrix} \quad , \quad M = \begin{bmatrix} -I_\eta & 0 & 0 \\ a^T & -\alpha & 0 \\ -a^T & 0 & -\beta \end{bmatrix} \in R^{(\eta+2)\times(\eta+2)}$$

where $e \in \mathrm{R}^\eta$ is a vector of ones, I_η is the identity matrix of order η and α and β are two positive real numbers that can be chosen such that the matrix M is negative semi-definite (NSD) or indefinite (IND).

FORM1-NSD: α and β satisfy

$$\alpha > \theta\frac{a^T a}{4} \quad , \quad \beta > \theta\alpha\frac{a^T a}{4\alpha - a^T a}$$

FORM1-IND: α and β satisfy

$$\alpha > \theta\frac{a^T a}{4} \quad , \quad \beta > \frac{\alpha}{\theta}\frac{a^T a}{4\alpha - a^T a}$$

with $\theta > 1$ a fixed number.

Notice that we can consider $q = \begin{bmatrix} e \\ -b+\varepsilon \\ b+\varepsilon \end{bmatrix}$ with ε a small positive constant, which does not change the result to the subset sum problem and avoids degenerate feasible solutions. This perturbation has been considered in the computational tests.

FORM2 – the LCP problem is defined by

$$q = \begin{bmatrix} a \\ -b \\ b \end{bmatrix} \quad , \quad M = \begin{bmatrix} -I_\eta & e & -e \\ e^T & -2\eta & 0 \\ -e^T & 0 & -2\eta \end{bmatrix} \in R^{(\eta+2)\times(\eta+2)}$$

In this case the matrix M is symmetric NSD.

Several test problems have been generated following the same rules as in [Júdice et al., 2002]: all the components a_i of the vector $a \in \mathrm{R}^\eta$ have been randomly generated in the interval [1, 50] and $b = \sum_{i\in I} a_i$,where I is a sub-set of {1,...,η} corresponding to the variables x_i that are equal to 1 in a solution of the subset sum problem. Three different cardinalities of the set I have been considered, corresponding to a percentage of variables equal to one of 25%, 50% and 75%, respectively.

The problems FORM1-IND have not been tested in the first experiment, although they have been considered in the second experiment. *Experiment 1* has used the problems FORM1-NSD 50%, FORM1-NSD 25%, FORM1-NSD 75% and FORM2 50%. Five different problems have been generated for each type

and for each value of η =20, 50 and 100. The FORM2 50% problems correspond to the same subset sum problems as FORM1-NSD 50%.

These problem instance parameter values (i.e. cardinality of I and values of η) are the ones used in [Júdice et al., 2002], [Ribeiro, 2004] and [Júdice et al., 2006].

- *Prob 8, Prob 9* [Fernandes et al., 2001]

These LCPs are formulations of non-zero bimatrix games:

$$\text{PROB8}: \quad q = \begin{bmatrix} -e^m \\ -e^r \end{bmatrix}, \quad M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}$$

$$\text{PROB9}: \quad q = \begin{bmatrix} -e^m \\ e^r \end{bmatrix}, \quad M = \begin{bmatrix} 0 & A \\ -B & 0 \end{bmatrix}$$

where e^j is a vector of ones of order j and A and B are positive matrices.

The elements of the matrices A and B have been randomly generated in the interval [1, 50]. We have also considered $m = r$ in all problems.

- *Sherali-ND*, *Sherali-IND* [Sherali et al. 1998]

These LCP problems have been proposed by [Sherali et al., 1998] and are called "Problem Set 1" therein. Two types of matrices M are considered. In the first case M is negative-definite (SHERALI-ND) and M is indefinite (SHERALI-IND) in the second. The procedure for generating the test problems is the following [Sherali et al., 1998]:

(a) A complementary solution $(\tilde{z}, \tilde{w})$ is first generated by randomly setting either $\tilde{z}_i$ or $\tilde{w}_i$ to zero, while the other variable is selected from the set {0,1,2} for each i=1,...,n.

(b) A matrix M composed of random integers in the interval [-15, 15] is generated.

(b.1) For M to be ND, the diagonal entries m_{ii} are redefined as

$$m_{ii} = -\left(1 + \max\left\{ \sum_{\substack{k=1 \\ k \neq i}}^{n} |m_{ik}|, \sum_{\substack{k=1 \\ k \neq i}}^{n} |m_{ki}| \right\}\right), \ \forall i = 1, ..., n$$

(b.2) For M to be IND, a full row of M is made negative.

(c) Finally, q is computed by $q = \tilde{w} - M\tilde{z}$.

6.2 Experiment 1

The Branch and Tabu Pivoting (BTP) algorithm has been tested on the above problems considering different values for the parameters *MaxIter_root* (maximum number of iterations for the heuristic in node 0), *MaxIter_child* (maximum number of iterations for the heuristic in the other nodes of the search tree) and the tabu tenure TT (the maximum tabu status and also the number of iterations that a variable has a positive penalty).

Preliminary tests indicated that a better overall performance is obtained with high values of the tabu tenure TT. Thus, for the Experiment 1 reported in Table 1 we considered TT=50 in all problems. In addition, *MaxIter_root* was set to 50 and two different values for *MaxIter_child* were used: 10 and 30. These search parameter values have been selected after some initial experiences with a wider range of values. However, we did not seek to tailor search parameters for each type of problem separately. A brief note on the performance of the algorithm with other parameter values is given afterwards.

Table 1 indicates the number of nodes required by the algorithm (Nd) and the total number of iterations (Nit), i.e. the number of pivot operations. This number includes extra iterations eventually performed in the *additional test of infeasibility* but excludes the iterations required in Step 0 to obtain a first basic solution for node 0. The dimension of the LCP is denoted by n.

Table 1 shows that the algorithm has been able to find a solution to the LCP formulations of the subset sum problems (FORM1 and FORM2) in a few iterations for most cases. Three atypical cases can be observed, where a large number of nodes and iterations (>1000) has been required. On average, the algorithm has shown a better performance using *MaxIter_child* =30 than using *MaxIter_child* =10 in this set of problems. The running times have been quite small for all the test problems.

The algorithm has shown a poor performance on problems PROB8, PROB9 and SHERALI. It has generally required the exploration of many nodes and a large number of iterations. In these problems, the algorithm has shown an average performance with *MaxIter_child* =10 better than with *MaxIter_child* =30.

We have carried out further tests considering different values of the parameters. We have experimented with *MaxIter_root*/ *MaxIter_child* equal to 100/10 and 100/30, keeping TT=50. The results for 100/10 have shown an overall superiority in relation to 100/30. Different values of the tabu tenure have been further tried, in particular TT=2n. The algorithm has shown an average performance similar to that of considering TT=50.

As a final conclusion, the results do not show the superiority of a set of parameter values in relation to the others in all circumstances. According to this study and regardless of the parameter values, the algorithm seems to be

efficient for the problems FORM1 and FORM2 and has revealed a much poorer performance in problems PROB8, PROB9 and SHERALI. Next, we include some observations that may explain the differences on the performance of the algorithm.

In problems FORM1 and FORM2, the noncomplementary solutions obtained throughout the algorithm have only one *violating* pair. Furthermore, it is easy to restore feasibility when the solutions are infeasible. Therefore, the tabu pivoting heuristic works with low values of *NoFeasib* and *NoCompl* and, in most cases, only one of these measures is strictly positive.

On the contrary, feasible solutions of the PROB8, PROB9 and SHERALI often have several *violating* pairs. The algorithm may require high levels of infeasibility to achieve complementarity and vice-versa. Therefore, the tabu pivoting heuristic works with values of *NoFeasib* and *NoCompl* significantly higher than in the previous problems. Furthermore, problems PROB8 and PROB9 give an additional difficulty to the heuristic. Due to their structure, a negative basic variable can never exchange with its complementary variable when the latter is nonbasic, because the respective entry in the pivot row is always zero.

Table 1: Numerical results of the BTP algorithm in *Experiment 1*

Problems	n	Instance	BTP algorithm (*initial solution given by* P_1)			
			MaxIter_child =10		*MaxIter_child* =30	
			Nd	Nit	Nd	Nit
FORM1-NSD 50%	22	K0_n20_1	7	120	3	118
		K0_n20_2	12	184	5	192
		K0_n20_3	9	142	3	123
		K0_n20_4	0	36	0	36
		K0_n20_5	26	282	2	98
	52	K0_n50_1	2	72	1	72
		K0_n50_2	0	24	0	24
		K0_n50_3	0	45	0	45
		K0_n50_4	6	119	3	117
		K0_n50_5	0	44	0	44
	102	K0_n100_1	0	38	0	38
		K0_n100_2	8	140	5	201
		K0_n100_3	0	14	0	14
		K0_n100_4	0	16	0	16
		K0_n100_5	3	78	3	129
FORM1-NSD 25%	22	K1_n20_1	2	68	3	129
		K1_n20_2	3	82	3	137
		K1_n20_3	121	840	4	160
		K1_n20_4	0	39	0	39
		K1_n20_5	1	58	1	58

Table 1: Numerical results of the BTP algorithm in *Experiment 1*

Problems	n	Instance	BTP algorithm (*initial solution given by* P_1)			
			MaxIter_child =10		*MaxIter_child* =30	
			Nd	Nit	Nd	Nit
	52	K1_n50_1	0	47	0	47
		K1_n50_2	0	44	0	44
		K1_n50_3	0	38	0	38
		K1_n50_4	6	109	2	112
		K1_n50_5	23	281	3	124
	102	K1_n100_1	0	27	0	27
		K1_n100_2	0	11	0	11
		K1_n100_3	1	61	1	61
		K1_n100_4	0	38	0	38
		K1_n100_5	0	18	0	18
FORM1-NSD 75%	22	K2_n20_1	0	17	0	17
		K2_n20_2	6	112	2	98
		K2_n20_3	0	21	0	21
		K2_n20_4	336	2246	6	216
		K2_n20_5	320	2143	3	121
	52	K2_n50_1	0	5	0	5
		K2_n50_2	0	33	0	33
		K2_n50_3	0	48	0	48
		K2_n50_4	0	42	0	42
		K2_n50_5	5	108	2	108
	102	K2_n100_1	0	28	0	28
		K2_n100_2	0	12	0	12
		K2_n100_3	0	39	0	39
		K2_n100_4	3	78	2	89
		K2_n100_5	0	22	0	22
FORM2 50%	22	KK_n20_1	2	70	1	70
		KK_n20_2	10	155	4	156
		KK_n20_3	0	37	0	37
		KK_n20_4	391	2606	3	121
		KK_n20_5	9	151	2	85
	52	KK_n50_1	0	14	0	14
		KK_n50_2	5	104	3	117
		KK_n50_3	6	120	5	190
		KK_n50_4	0	0	0	0
		KK_n50_5	0	6	0	6
	102	KK_n100_1	2	70	1	70
		KK_n100_2	1	58	1	58
		KK_n100_3	2	68	4	160
		KK_n100_4	0	41	0	41

Table 1: Numerical results of the BTP algorithm in *Experiment 1*

Problems	n	Instance	BTP algorithm (*initial solution given by* P_1)			
			MaxIter_child =10		*MaxIter_child* =30	
			Nd	Nit	Nd	Nit
		KK_n100_5	14	210	2	84
FORM1,2 ***average values***			22	196	**1**	**74**
PROB8	20	Prob8_n20_1	153	1482	120	2849
		Prob8_n20_2	151	1699	217	5287
		Prob8_n20_3	156	1657	379	7818
		Prob8_n20_4	866	8732	503	11090
		Prob8_n20_5	1	51	1	51
PROB9	20	Prob9_n20_1	521	5414	1758	39779
		Prob9_n20_2	571	5831	235	4722
		Prob9_n20_3	0	12	0	12
		Prob9_n20_4	435	4365	263	5948
		Prob9_n20_5	168	1659	5	211
PROB 8, 9 ***average values***			**302**	**3090**	348	7777
SHERALI-ND	25	SND_n25_1	52	898	17	657
		SND_n25_2	129	1465	10	471
		SND_n25_3	0	1	0	1
		SND_n25_4	0	5	0	5
		SND_n25_5	6	134	7	289
SHERALI-IND	25	SI_n25_1	164	2055	4	156
		SI_n25_2	16	376	514	13665
		SI_n25_3	0	40	0	40
		SI_n25_4	4	116	3959	94461
		SI_n25_5	8155	98921	9943	250849
SHERALI ***average values***			**853**	**10401**	1445	36059

6.3 Experiment 2

The second experiment has consisted of applying the BTP algorithm to a set of 36 LCPs associated with subset sum problems that have already been tested in other studies. These instances have formulations of types FORM1-NSD, FORM1-IND and FORM2 (75%, 50% and 25%, respectively for instances identified with A, B and C in Table 2). The results of the BTP algorithm on these instances are provided for comparison with the results of other enumerative methods. The three enumerative algorithms EASET, EMRG and EMPEC discussed in Section 1 have been used for this comparison.

As before, we have also considered the initial solution obtained by the optimization of the problem (P_1). Table 2 shows the results of the BTP algorithm

using *MaxIter_root*=50, *MaxIter_child*=30 and TT=50 and the results of the other enumerative algorithms mentioned before. The best performance values for each instance are in bold type.

As in the previous experiment, the BTP algorithm appears to perform well for LCPs associated with subset sum problems. In this case the results assume more significance as they can be compared with results of previous studies. As can be seen in Table 2, the average number of nodes and of iterations over this set of problems are lower with the BTP algorithm than with the other enumerative algorithms.

Table 2: *Experiment 2*- Comparison of BTP with other enumerative algorithms

Prob.	n	Instance	BTP		EAsetE		EMRG		EMPEC	
			Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
	22	1NA1	5	191	75	168	31	140	**2**	**66**
	52	1NA2	**2**	106	10	**71**	14	98	280	11894
	102	1NA3	**0**	**40**	2	57	66	272	2	206
	152	1NA4	**0**	**10**	25	171	156	570	2	528
Form1	22	1NB1	**3**	124	78	167	**5**	**21**	18	304
-nsd	52	1NB2	**0**	**35**	31	113	35	150	2	72
	102	1NB3	**1**	60	4	**34**	16	104	2	415
	152	1NB4	**0**	**38**	2	48	32	174	2	205
	22	1NC1	**3**	119	5	**12**	5	17	6	163
	52	1NC2	3	136	17	**57**	55	237	**2**	217
	102	1NC3	**1**	53	24	108	4	**38**	2	208
	152	1NC4	**0**	**37**	12	62	23	113	2	162
	22	1IA1	5	191	75	168	31	140	**2**	**66**
	52	1IA2	**2**	106	10	**71**	14	98	280	11886
	102	1IA3	**0**	**40**	2	57	66	272	2	206
	152	1IA4	**0**	**10**	25	171	156	570	2	528
Form1	22	1IB1	3	124	78	167	**5**	**21**	18	304
-ind	52	1IB2	**0**	**35**	31	113	35	150	2	72
	102	1IB3	**1**	60	4	**34**	16	104	2	415
	152	1IB4	**0**	**38**	2	48	32	174	2	205
	22	1IC1	3	119	**5**	**12**	5	17	6	163
	52	1IC2	3	136	17	**57**	55	237	**2**	217
	102	1IC3	**1**	53	24	108	4	**38**	2	208
	152	1IC4	**0**	**37**	12	62	23	113	2	162
	22	2A1	4	121	12	75	**2**	**23**	2	191
	52	2A2	3	125	6	60	3	**58**	**2**	443
	102	2A3	**0**	**19**	13	142	1	81	2	974
	152	2A4	**0**	**13**	1	113	2	117	2	593
	22	2B1	**0**	**5**	2	11	46	343	46	1357

Table 2: *Experiment 2*- Comparison of BTP with other enumerative algorithms

Prob.	n	Instance	BTP		EAsET		EMRG		EMPEC	
			Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
FORM2	52	2B2	4	147	6	**44**	8	78	**2**	638
	102	2B3	**0**	**5**	1	55	1	51	2	256
	152	2B4	**0**	**42**	10	115	1	83	2	682
	22	2C1	2	107	16	70	5	35	**2**	**29**
	52	2C2	**0**	46	2	**20**	5	39	22	1126
	102	2C3	**0**	46	4	47	2	**37**	2	1269
	152	2C4	2	94	7	71	**2**	**56**	2	733
Average values:			***1***	***74***	*18*	*82*	*27*	*135*	*20*	*1032*

6.4 Experiment 3

Experiment 3 was intended to test different procedures for computing the initial basic feasible solution and to analyse the performance of the BTP algorithm after applying these procedures. This experiment has been carried out using the problems already considered in the previous experiments. Four procedures presented in Section 4 are considered to obtain the initial solution:

a) minimizing the sum of the variables w_i over the feasible region, i.e., solving (P_1) (this has been used in *experiments 1* and 2);

b) minimizing the sum of the variables z_i over the feasible region, i.e., solving (P_2);

c) applying a modified reduced-gradient (MRG) algorithm;

d) using Lemke's algorithm and a post-processing phase to reach feasibility of the LCP when Lemke's algorithm cannot solve the LCP.

Table 3 shows the results of the BTP algorithm using these procedures. The parameter values are TT=50, *MaxIter_root* =50 and *MaxIter_child*=30 for the LCPs associated with subset sum problems and *MaxIter_child*=10 for the other problems. Note that the alternative procedures are used only to compute the initial basic feasible solution for the first time the tabu pivoting heuristic is applied, that is, in the root of the search tree. The notation "—" is used in Table 3 whenever the initial procedure yields a solution to the LCP. These cases are ignored in the computation of the average values shown in this table.

As can be seen in Table 3, the MRG method was able to find a solution to the LCP in 72 out of 115 problems. On the other hand, Lemke's algorithm could solve the LCPs of type PROB9 and terminated in an unbounded ray in

the other problems. In all these problems, except in SHERALI-IND, Lemke's algorithm only required one iteration to terminate in a ray.

This experiment has shown that the initial solution has a significant impact on the computation of a solution to a LCP. The MRG and Lemke's algorithms are useful procedures to start with, because they are able to achieve a solution to the LCP in several cases without requiring any further computations. However, in the other cases, there is no guarantee that a solution to the LCP is found more easily if the algorithm BTP starts with a basic solution given by one of these more advanced techniques. We have observed that the initial solutions obtained by MRG for the problems PROB8, PROB9 and SHERALI have less violating pairs than the corresponding solutions obtained by solving (P_1), but this does not ensure a better performance of the algorithm. Nevertheless, those starting procedures are powerful tools because they can provide different starting solutions that can be used to initiate several runs of the algorithm. Using a multi-start approach, the BTP algorithm has a good change of finding a solution to the LCP in a few iterations. This is supported by the experiment reported in Table 3, as all but two problems were solved in less than 130 iterations of the BTP algorithm in at least one run (from *a* to *d*).

We have also tested a different implementation of the enumerative algorithm in which the MRG is applied before the tabu pivoting heuristic in each node of the search tree. The results are not encouraging as a larger total number of iterations has been mostly required.

As in *experiment* 1, further tests with different parameters values have been also carried out. We have considered some combinations of *MaxIter_root* = 50 or 100, *MaxIter_child* = 10 or 30 and $TT = 50$ or $2n$. A comparison of these tests suggests that the combination *MaxIter_root* = 100, *MaxIter_child* = 10 and $TT = 50$ seems to be a good compromise parameter setting within the set of the tested problems. In this set of 115 problems, 114 were solved in less than 220 iterations (113 of them solved in less than 120 iterations), by at least one run of the BTP algorithm (from *a* to *d*). The remaining problem (Prob8_n20_3) required 3214 iterations.

Table 3: *Experiment 3*- BTP algorithm with different initial solutions

		BTP algorithm with initial solution given by:							
		a)(P_1)		*b*)(P_2)		*c*) MRG		*d*) Lemke	
Prob. type	Instance	Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
	1NA1	5	191	10	335	—		3	131
	1NA2	2	106	0	37	—		1	65
	1NA3	0	40	0	15	—		0	17
	1NA4	0	10	0	5	—		1	66
***Group 1*:**	1NB1	3	124	1	59	4	145	147	2612

Table 3: *Experiment 3*- BTP algorithm with different initial solutions

		BTP algorithm with initial solution given by:							
		a)(P_1)		*b*)(P_2)		*c*) MRG		*d*) Lemke	
Prob. type	Instance	Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
LCPs	1NB2	0	35	0	37	—		0	37
associated	1NB3	1	60	0	31	—		0	16
with	1NB4	0	38	0	25	—		0	48
subset sum	1NC1	3	119	12	396	1	70	0	1
problems	1NC2	3	136	5	179	2	104	5	187
	1NC3	1	53	1	70	—		1	70
	1NC4	0	37	1	81	—		0	18
	1IA1	5	191	10	335	—		3	131
	1IA2	2	106	0	37	—		1	65
	1IA3	0	40	0	15	—		0	17
	1IA4	0	10	0	5	—		1	66
	1IB1	3	124	1	59	4	145	147	2612
	1IB2	0	35	0	37	—		0	37
	1IB3	1	60	0	31	—		0	16
	1IB4	0	38	0	25	—		0	48
	1IC1	3	119	12	396	5	197	0	1
	1IC2	3	136	5	179	2	104	5	187
	1IC3	1	53	1	70	—		1	70
	1IC4	0	37	1	81	—		0	18
	2A1	4	121	4	121	5	176	0	11
	2A2	3	125	0	10	0	45	0	50
	2A3	0	19	0	19	—		0	0
	2A4	0	13	3	132	—		5	185
	2B1	0	5	0	5	—		0	40
	2B2	4	147	2	87	—		0	0
	2B3	0	5	0	5	—		2	105
	2B4	0	42	1	53	—		1	74
	2C1	2	107	0	49	—		2	98
	2C2	0	46	0	38	—		8	300
	2C3	0	46	2	108	2	87	2	98
	2C4	2	94	0	16	—		3	141
	K0_n20_1	3	118	2	86	—		5	200
	K0_n20_2	5	192	0	37	0	7	0	37
	K0_n20_3	3	123	0	22	—		0	37
	K0_n20_4	0	36	0	1	3	116	0	1
	K0_n20_5	2	98	1	63	—		0	43
	K0_n50_1	1	72	1	65	—		0	22
	K0_n50_2	0	24	0	18	—		1	68
	K0_n50_3	0	45	4	158	—		1	61
	K0_n50_4	3	117	3	136	1	57	1	63

Table 3: *Experiment 3*- BTP algorithm with different initial solutions

		BTP algorithm with initial solution given by:							
		a)(P_1)		*b*)(P_2)		*c*) MRG		*d*) Lemke	
Prob. type	Instance	Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
	K0_n50_5	0	44	0	16	—		0	39
	K0_n100_1	0	38	1	68	—		0	19
	K0_n100_2	5	201	0	47	—		1	53
	K0_n100_3	0	14	0	29	—		0	26
	K0_n100_4	0	16	0	48	—		0	24
	K0_n100_5	3	129	2	108	—		2	108
	K1_n20_1	3	129	97	1719	4	150	6	228
	K1_n20_2	3	137	2	105	121	2141	2	105
	K1_n20_3	4	160	0	13	116	2020	0	31
	K1_n20_4	0	39	0	11	—		0	25
	K1_n20_5	1	58	0	49	1	58	1	69
	K1_n50_1	0	47	0	22	—		0	39
	K1_n50_2	0	44	5	187	—		0	31
	K1_n50_3	0	38	0	16	0	22	3	130
	K1_n50_4	2	112	0	14	—		6	215
	K1_n50_5	3	124	0	42	3	123	0	1
	K1_n100_1	0	27	0	24	—		0	19
	K1_n100_2	0	11	0	34	0	5	0	26
	K1_n100_3	1	61	0	45	—		0	1
	K1_n100_4	0	38	5	184	—		1	72
	K1_n100_5	0	18	0	36	—		1	77
	K2_n20_1	0	17	0	5	—		0	5
	K2_n20_2	2	98	0	39	0	7	0	1
	K2_n20_3	0	21	2	104	—		2	104
	K2_n20_4	6	216	6	216	3	128	118	2149
	K2_n20_5	3	121	9	329	3	121	1	55
	K2_n50_1	0	5	1	77	—		0	5
	K2_n50_2	0	33	3	121	—		0	31
	K2_n50_3	0	48	1	68	0	11	1	71
	K2_n50_4	0	42	0	28	—		0	33
	K2_n50_5	2	108	0	18	0	14	0	18
	K2_n100_1	0	28	0	19	—		1	71
	K2_n100_2	0	12	2	97	0	11	1	78
	K2_n100_3	0	39	0	36	—		0	20
	K2_n100_4	2	89	0	19	2	90	0	38
	K2_n100_5	0	22	0	16	—		0	5
	KK_n20_1	1	70	1	70	0	30	4	147
	KK_n20_2	4	156	4	156	—		1	75
	KK_n20_3	0	37	715	11956	0	33	7	244
	KK_n20_4	3	121	0	5	0	27	0	0

Table 3: *Experiment 3*- BTP algorithm with different initial solutions

		BTP algorithm with initial solution given by:							
		a)(P_1)		*b*)(P_2)		*c*) MRG		*d*) Lemke	
Prob. type	Instance	Nd	Nit	Nd	Nit	Nd	Nit	Nd	Nit
	KK_n20_5	2	85	0	42	0	40	0	4
	KK_n50_1	0	14	0	14	—		1	73
	KK_n50_2	3	117	3	117	—		0	11
	KK_n50_3	5	190	0	29	—		0	45
	KK_n50_4	0	0	0	5	—		0	20
	KK_n50_5	0	6	0	20	—		1	56
	KK_n100_1	1	70	0	3	—		5	201
	KK_n100_2	1	58	4	161	—		3	132
	KK_n100_3	4	160	0	0	—		4	161
	KK_n100_4	0	41	0	43	—		0	42
	KK_n100_5	2	84	2	84	—		3	144
Group 1 average values		*1*	*74*	*10*	*213*	*9*	*209*	*5*	*141*
	Prob8_n20_1	153	1482	76	858	483	4794	4	100
	Prob8_n20_2	151	1699	51	596	253	2675	1	61
Group 2:	Prob8_n20_3	156	1657	724	7159	1282	12516	1048	10378
PROB8	Prob8_n20_4	866	8732	1033	10494	236	2404	0	11
and	Prob8_n20_5	1	51	513	5269	2	78	0	41
PROB9	Prob9_n20_1	521	5414	0	4	68	809	—	
	Prob9_n20_2	571	5831	395	3972	1069	10846	—	
	Prob9_n20_3	0	12	481	5222	9	175	—	
	Prob9_n20_4	435	4365	0	24	—		—	
	Prob9_n20_5	168	1659	382	4211	203	2158	—	
Group 2 average values		*302*	*3090*	*366*	*3781*	*401*	*4051*	*211*	*2118*
	SND_n25_1	52	898	278	3995	332	3993	0	41
	SND_n25_2	129	1465	0	39	34	643	1161	13894
	SND_n25_3	0	1	0	5	—		0	3
Group 3:	SND_n25_4	0	5	2	100	0	3	2	100
SHERALI	SND_n25_5	6	134	0	5	—		454	4326
	SI_n25_1	164	2055	3383	44793	—		105	1542
	SI_n25_2	16	376	437	6060	—		1221	15924
	SI_n25_3	0	40	3	108	4	112	517	6301
	SI_n25_4	4	116	13064	165435	—		1593	19297
	SI_n25_5	8155	98921	176	2322	12451	149328	9198	115838
Group 3 average values		*853*	*10401*	*1734*	*22286*	*2564*	*30816*	*1425*	*17727*

7 Concluding Remarks

In this paper we have proposed an algorithm for solving the linear complementarity problem, which results from the integration of a tabu pivoting heuristic into an enumerative framework. The heuristic uses the *tabu search* principle by imposing tabu restrictions on pivot operations in order to avoid inversions or repetitions of recent moves.

Computational experiments with this Branch and Tabu Pivoting (BTP) algorithm have shown that for some types of NP-hard LCPs the algorithm is able to find a solution in a few iterations. However, for other LCPs the algorithm may require to visit many nodes of the search tree and to perform a large number of iterations.

We believe that the good performance revealed in many cases makes the BTP algorithm a promising technique for using together with some other techniques usually employed in enumerative algorithms. In addition, simple iterative procedures as the MRG and Lemke's algorithms may be useful tools to start with because either they find a solution to the LCP or provide different starting solutions that can be used to initialize several runs of the BTP algorithm. In fact, considering only one parameter setting of the algorithm, 113 out of 115 problems could be solved in less than 130 iterations by the BTP algorithm with one of the four starting procedures.

References

[1] Aboudi, R., Jörnsten, K., (1994) Tabu search for general zero-one integer programs using the pivot and complement heuristic. ORSA Journal on Computing, vol. 6, nr. 1, pp. 82-93.

[2] Al-Khayyal, F.A., (1987) An implicit enumeration procedure for the general linear complementarity problem. Mathematical Programming Study, vol. 31, pp. 1-20.

[3] Balas, E., Martin, C.H., (1980) Pivot and complement – a heuristic for 0-1 programming. Management Science, vol.26, nr.1, pp. 86-96.

[4] Chelouah, R., Siarry, P., (2000) Tabu Search Applied to Global Optimization. European Journal of Operations Research, vol. 123, pp. 256-270.

[5] Cottle, R., Pang, J., Stone, R., (2009) The Linear Complementarity Problem. SIAM.

[6] Fernandes, L., Friedlander, A., Guedes, M.C., Júdice, J., (2001) Solution of a general linear complementarity problem using smooth optimization and

its application to bilinear programming and LCP. Applied Mathematics and Optimization, vol. 43, pp. 1-19.

[7] Gendreau, M., Marcotte, P., Savard, G., (1996) A hybrid Tabu-ascent algorithm for the linear bilevel programming problem. Journal of Global Optimization, vol. 8, nr. 3, pp. 217-233.

[8] Glover, F. (1986) Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research, vol. 13, pp. 533-549.

[9] Glover, F., Laguna, M. (1993) Tabu Search. In: Reeves, C. (Ed), Modern Heuristics for Combinatorial Problems, Blackwell Scientific Publications, Oxford, pp. 70-150.

[10] Júdice, J.J., (1994) Algorithms for Linear Complementarity Problems. In: Spedicato, E. (Ed), Algorithms for Continuous Optimization, Kluwer Academic Publishers, pp. 435-474.

[11] Júdice, J.J., Faustino, A.M., Ribeiro, I.M. (2002) On the solution of NP-hard linear complementary problems. TOP-Sociedad Espanola de Estadística e Investigación Operativa, vol. 10, nr.1, pp. 125-145.

[12] Júdice, J.J., Sherali, H.D., Ribeiro, I.M., Faustino, A.M., (2006) A complementarity-based partitioning and disjunctive cut algorithm for mathematical programming problems with equilibrium constraints. Journal of Global Optimization, vol. 36, pp. 89-114.

[13] Kojima, M., Megiddo, N., Noma, T., Yoshise, A. (1991) A unified approach to interior-point algorithms for linear complementarity problems. Lecture Notes in Computer Science, vol. 538, Springer-Verlag.

[14] Kovacevic-Vujcic, V.V., Cangalovic, M.M., (1999) Tabu search methodology in global optimization. Computers and Mathematics with Applications, vol. 37, nr. 4-5, pp. 125-133.

[15] Lan, K.M., Wen, U.P., Shih, H.S., Lee, E.S., (2007) A hybrid neural network approach to bilevel programming problems. Applied Mathematics Letters, vol. 20, nr. 8, pp. 880-884.

[16] Lemke, C., (1968) On complementary pivot theory. In: Dantzig, G., Veinott, A. (Eds), Mathematics of Decision Sciences, American Mathematical Society, Providence, pp. 95-114.

[17] Løkketangen, A., Glover, F., (1995) Tabu search for zero/one mixed integer programming with advanced level strategies and learning. International Journal of Operations and Quantitative Management, vol. 1, nr. 2, pp. 89-109.

[18] Løkketangen, A., Glover, F., (1998) Solving zero-one mixed integer programming problems using tabu search. European Journal of Operational Research, vol. 106, pp. 624-658.

[19] Løkketangen, A., Jörnsten, K., Storøy, S., (1994) Tabu search within a pivot and complement framework. International Transactions in Operational Research, vol. 1, nr. 3, pp. 305-316.

[20] Michalewicz, Z., Fogel, D.B., (2004) How to Solve It: Modern Heuristics. Springer.

[21] Murty, K., (1988) Linear Complementarity, Linear and Nonlinear Programming. Heldermann Verlag, Berlin.

[22] Rajesh, J., Gupta, K., Kusumakar, H.S., Jayaraman, V.K., Kulkarni, B.D., (2003) A tabu search based approach for solving a class of bilevel programming problems in chemical engineering. Journal of Heuristics, vol. 9, nr. 4, pp. 307-319.

[23] Ribeiro, I., (2004) Optimização global e aplicações em engenharia estrutural. PhD Dissertation (in Portuguese), Faculdade de Engenharia, Universidade do Porto, Portugal.

[24] Sherali, H.D., Krishnamurthy, R.S., Al-Khayyal, F.A. (1998) Enumeration approach for linear complementarity problems based on a reformulation-linearization technique. Journal of Optimization theory and applications, vol. 99, nr. 2, pp. 481-507.