Les Cahiers du GERAD ISSN: 0711–2440

Improved Primal Simplex Version 3: Cold Start, Generalization for Bounded Variable Problems and a New Implementation

V. Raymond, F. Soumis,
A. Metrane

G–2009–15

February 2009

Improved Primal Simplex Version 3: Cold Start, Generalization for Bounded Variable Problems and a New Implementation

Vincent Raymond

François Soumis

Abdelmoutalib Metrane

GERAD and École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada, H3C 3A7

vincent.raymond@polymtl.ca
francois.soumis@gerad.ca
abdelmoutalib.metrane@gerad.ca

February 2009

Les Cahiers du GERAD

G–2009–15

Abstract

The *improved primal simplex* (IPS) method has been proposed by Elhallaoui et al. [9]. We rewrite the theory of IPS for a cold start with an initial feasible solution instead of an initial basic feasible solution. This allows us to use an *heuristic first - optimization second* strategy. We generalize this algorithm so that it can handle bounds on variables. We show that variables at upper bounds augment degeneracy, and consequently, increase performance of IPS compared to CPLEX. We simplify the implementation by replacing the UMFPACK [5] procedure by certain modules of the CPLEX library. This allows the user to work with only one commercial software package. We obtain a reduction factor of solution time of 20 on fleet assignment instances with bounded variables.

Key Words: Linear programming, primal simplex, degeneracy.

Résumé

La méthode *improved primal simplex* (IPS) a été proposée par Elhallaoui et al. [9]. Nous réécrivons la théorie de manière à ce que l'algorithme débute avec une solution réalisable initiale plutôt qu'avec une base réalisable initiale. Ceci permet d'utiliser la stratégie *heuristic first - optimization second*. Nous généralisons ensuite la méthode pour qu'elle puisse résoudre des problèmes où les variables sont bornées. Nous montrons que les variables qui sont à leur borne supérieure augmente la dégénérescence, et conséquemment, la performance de IPS. Enfin, nous simplifions l'implémentation en remplaçant les procédures de UMFPACK par certains modules de la librairie de CPLEX. Nous obtenons un facteur de réduction de plus de 20 comparativement à CPLEX sur des problèmes de répartition de flotte d'avions.

1 Introduction

We consider the solution of linear programs in standard form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ c^T x \quad \text{subject to}\ Ax = b,\ x \geq 0, \tag{LP}$$

where $c \in \mathbb{R}^n$ is the cost-vector, A is the $m \times n$ constraint matrix and $b \in \mathbb{R}^m$ is the right-hand side. We are particularly interested in so-called degenerate problems, on which the simplex algorithm [4] is likely to encounter degenerate pivots.

Our paper is organized as follows. In §2 we present the *Improved Primal Simplex* (IPS) proposed by Elhallaoui et al. [9] and developed in [16] (IPS-2) and we present the theory to handle bounds in the simplex method. In §3, we describe the goals of this article, that is, we rewrite the theory so that the algorithm can start with an initial feasible solution instead of an initial basic feasible solution. We generalize IPS to problems with bounded variables to take advantage of the degeneracy of variables that are at their upper bounds. We simplify the implementation of IPS by removing a procedure from UMFPACK [5]. This new version of IPS is called IPS-3. Numerical results are given in §4 and in §5 we present the conclusions.

2 Background

In this section, we present the IPS algorithm and some theoretical properties developed in [9]. A brief subsection on how to handle bounds in the simplex method is also given. We start with the presentation of IPS summarized in Algorithm 2.1.

Algorithm 2.1 The Improved Primal Simplex algorithm [9].

Step 0. Choose an initial basis B for (LP).

Step 1. Form and solve the reduced problem (RP).

Step 2. Form and solve the complementary problem (SD). Let y^* be its optimal value.

Step 3. If $y^* \geq 0$, the current solution of (RP) is optimal for (LP).

Step 4. Otherwise, construct a new basis B' using the solution of (SD) and return to Step 1.

At Step 0 of Algorithm 2.1, the method starts with a degenerate basic feasible solution. To obtain this basis, the authors of [9] solve a phase I of the simplex algorithm on (LP). At Step 1 the reduced problem RP is formed according to the theory of Pan [15]. Note that IPS carries out the reduction by means of the commercial software package UMFPACK. Then at Step 2, the complementary problem (CP) is formed and its dual (SD) is solved. We recall from [9] in §3.2 the construction of this problem. Step 3 is the optimality condition of IPS. Elhallaoui *et al.* [9] prove that if the value of the objective function of (CP) is nonnegative, then the optimal solution of RP is optimal for (LP). Finally, Step 4 constructs a new basis for the reduced problem. We refer the reader to [9] for the technicalities of the last two steps. We note that IPS solves unbounded variable problems only.

In [9], the authors prove the following theorem.

Theorem 2.1 *Let x be a (non-optimal) basic feasible solution of (LP) with corresponding basis B and with $p \leq m$ positive components. Let S be the index set of the positive components in a solution v of (SD) ($|S| \leq m - p + 1$). Let w be the convex combination of columns of A whose coefficients are the v_j, that is, $w = \sum_{j\in S} v_j A_j$.*
Then:

1. *the variables in S can enter the basis B with positive values, decreasing the objective function value of (LP);*

2. w is linearly dependent on the columns corresponding to the p positive components of x.

The numerical results of IPS show the efficiency of this method. More precisely, it obtains an objective function reduction factor of less than 2 for vehicle crew scheduling problems and more than 3 for fleet assignment problems. More recently, the improved version of IPS, called IPS-2 [16], obtains a reduction factor of more than 3 and more than 12 respectively for the previous two problem types.

2.1 Bounded Variables in (LP)

We consider a linear program with bounded variables in standard form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ c^T x \quad \text{subject to } Ax = b,\ l \le x \le u, \tag{BLP}$$

where l and u are the vectors of the lower and the upper bounds respectively.

When bounds on variables are present in (LP), the simplex algorithm handles them by using substitution. At a given iteration, the simplex algorithm identifies the variable x_i to enter the basis by taking into consideration the lower bound and the upper bound of this variable. If the value of the variable decreases to its lower bound, the algorithm considers $y_i = x_i - l_i$. On the other hand, if the new value of x_i reaches its upper bound, the algorithm considers $y_i = u_i - x_i$. The standard treatment of the $y_i \ge 0$ constraint allows us to deal with lower and upper bounds on x_i. Therefore, when $x_i = l_i$ or $x_i = u_i$, x_i can be basic or non basic.

2.2 Notation

If $x \in \mathbb{R}^n$ and $I \subseteq \{1,\dots,n\}$ is an index set, we denote by x_I the subvector of x indexed by I. Similarly, if A is an $m \times n$ matrix, we denote by A_I the $m \times |I|$ matrix whose columns are indexed by I. Let $J = \{1,\dots,n\} \setminus I$, we write $x = (x_I, x_J)$ even though the indices in I and J may not appear in order. Moreover, we denote with upper indices the subset of rows associated with the indexed variables set.

The jth column of A is denote by A_j and we denote by A^{-T} the inverse of the transpose of A.

3 Contributions

This article presents three improvements of IPS. First, instead of using an initial basic feasible solution to reduce the problem as in [9], we present in §3.1 a new reduction method using only a feasible degenerate solution. This new reduction is well adapted to the *heuristic first - optimization second* approach [2] that starts with an heuristic algorithm and finishes with a mathematical programming optimization.

Secondly, instead of using UMFPACK [5], we present in §3.3 the procedure using only CPLEX to obtain the reduced and the complementary problems. Doing all the computation with the same programs avoids possible numerical tolerance incompatibilities and allows the user to work with only one commercial software package.

Finally, §3.4 presents the generalization to bounded variables. The algorithm also removes the basic variables at their upper bounds from the reduced problem.

Numerical results of each of the improvements are given in §4.

3.1 Reduced Problem

Let $\bar{x}$ be a feasible solution of (LP) and P be the index set of the p nonzero variables of $\bar{x}$, i.e.,

$$P = \{i \in \{1,\dots,n\} \mid \bar{x}_i > 0\},$$

where $\bar{x}_i$ may be integer or not. We construct a basis A_{B} of (LP) such that the first p variables are the variables of P and the last $m-p$ variables are artificial (denoted by the index set N). Without loss of generality, we assume that the p rows associated with the variables of P are the first p rows of (LP). In the same way, we assume that the $m-p$ rows associated with the $m-p$ artificial variables of N are the last $m-p$ rows of (LP). Thus, we can write

$$A_{\text{B}} = \begin{bmatrix} A_{\text{P}}^{\text{P}} & 0 \\ A_{\text{P}}^{\text{N}} & A_{\text{N}}^{\text{N}} \end{bmatrix}.$$

Suppose that the A_{P}^{P} matrix is non singular. It is the case when the p variables of P are linearly independent. If not, it is possible to find a new solution from $\bar{x}$ such that the nonzero variables are linearly independent by solving

$$\underset{x\in\mathbb{R}^m}{\text{minimize}}\ c_{\text{P}}^T x \quad \text{subject to}\ A_{\text{B}}x = b, x \geq 0.$$

Note that this problem is solved anyway in the CPLEX reduction procedure (see 3.3). Thus, the reduced problem is always created from a linearly independent solution. Denote by Q the inverse matrix of A_{B} and partition

$$Q = \begin{bmatrix} Q^{\text{P}} \\ Q^{\text{Z}} \end{bmatrix} = \begin{bmatrix} (A_{\text{P}}^{\text{P}})^{-1} & 0 \\ -A_{\text{P}}^{\text{N}}(A_{\text{P}}^{\text{P}})^{-1} & A_{\text{N}}^{\text{N}} \end{bmatrix} = A_{\text{B}}^{-1}, \tag{3.1}$$

where Q^{Z} is the *compatibility matrix* formed by the last $m-p$ rows of Q. We have

$$\bar{x} = \begin{bmatrix} Q^{\text{P}}b \\ Q^{\text{Z}}b \end{bmatrix}, \text{ and therefore, } Q^{\text{Z}}b = 0. \tag{3.2}$$

We begin with the following definition.

Definition 3.1 *The jth variable of (LP), x_j, is said to be* compatible *if and only if $Q^{\text{Z}}A_j = 0$.*

From Definition 3.1, we let $C \subseteq \{1,\ldots,n\}$ denote the indices of variables that are compatible and $I = \{1,\ldots,n\} \setminus C$. Thus, x_{C} and x_{I} are the subvectors of x of compatible and incompatible variables, respectively. We partition the cost vector c accordingly into c_{C} and c_{I}, and the columns of A into A_{C} and A_{I}. The partitioning that we applied to A_{B} can be generalized to A_{C} and A_{I} and yields

$$A_{\text{I}} = \begin{bmatrix} A_{\text{I}}^{\text{P}} \\ A_{\text{I}}^{\text{N}} \end{bmatrix} \quad \text{and} \quad A_{\text{C}} = \begin{bmatrix} A_{\text{C}}^{\text{P}} \\ A_{\text{C}}^{\text{N}} \end{bmatrix}. \tag{3.3}$$

Upon premultiplying the equality constraints of (LP) by Q, we obtain $QAx = Qb$, which may be rewritten

$$\begin{bmatrix} Q^{\text{P}}Ax \\ Q^{\text{Z}}Ax \end{bmatrix} = \begin{bmatrix} ((A_{\text{P}}^{\text{P}})^{-1}A_{\text{C}}^{\text{P}})x_{\text{C}} & ((A_{\text{P}}^{\text{P}})^{-1}A_{\text{I}}^{\text{P}})x_{\text{I}} \\ (-A_{\text{P}}^{\text{N}}(A_{\text{P}}^{\text{P}})^{-1}A_{\text{C}}^{\text{P}} + A_{\text{C}}^{\text{N}})x_{\text{C}} & (-A_{\text{P}}^{\text{N}}(A_{\text{P}}^{\text{P}})^{-1}A_{\text{I}}^{\text{P}} + A_{\text{I}}^{\text{N}})x_{\text{I}} \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}b \\ 0 \end{bmatrix},$$

where we used (3.1) and (3.2). Since by definition x_{C} is compatible, we have $(-A_{\text{P}}^{\text{N}}(A_{\text{P}}^{\text{P}})^{-1}A_{\text{C}}^{\text{P}} + A_{\text{C}}^{\text{N}})x_{\text{C}} = 0$. Upon imposing $x_{\text{I}} = 0$, we obtain the *reduced problem*

$$\underset{x_{\text{C}}}{\text{minimize}}\ c_{\text{C}}^T x_{\text{C}} \quad \text{subject to}\ ((A_{\text{P}}^{\text{P}})^{-1}A_{\text{C}}^{\text{P}})x_{\text{C}} = Q^{\text{P}}b,\ x_{\text{C}} \geq 0. \tag{RP}$$

Problem (RP) is potentially much smaller than (LP) and is obtained from a degenerate primal basis. It only depends on the compatible variables—those that can enter a basis for (RP) without violating the constraints of (LP) that have been omitted. Note that by construction, if x_{C} is feasible for (RP), then $(x_{\text{C}}, 0)$ is feasible for (LP).

With this new theoretical presentation of the reduced problem, we show that we can use an initial feasible solution of (LP) instead of using an initial basic feasible solution to reduce the problem. This particularity allows the use of an heuristic method to obtain the values of the nonzero variables of a feasible solution. Moreover, an heuristic feasible solution has more of a chance to be closer to the optimal solution than the classical phase I solution. Furthermore, the computational time of finding an heuristic initial solution can be much less than a phase I procedure. For example, the phase I on the VCS instances (see 4.1 for definition) takes 55 seconds on average.

3.2 Complementary Problem

In this section, we present the construction of the complementary problem as explained in [9]. Note that a complementary problem is created from a basic feasible solution of the reduced problem and contains only incompatible variables. Let $\bar{x}_{\text{C}}$ be the current feasible solution for (RP). Here, compatibility is understood with respect to Q^{Z} that has been used to calculate the previous reduced problem.

Recall that P indexes the nonzero variables of the current solution of the reduced problem. Assume that the reduced problem is not degenerate (if it is, then reduce it). P can be considered as the indices of the compatible basic variables. Let V index the compatible nonbasic variables and I index the incompatible variables, i.e.,

$$P = \{i \in C \mid \bar{x}_i \textit{ basic}\}, \quad V = \{i \in C \mid \bar{x}_i \textit{ nonbasic}\}, \quad \text{and} \quad I = \{1, \ldots, n\} \setminus C.$$

Then $\bar{x}_{\text{C}}$ is also optimal for (LP) if and only if all reduced costs (i.e., corresponding to all variables, compatible or not) are nonnegative. In other words, there must exist dual variables π such that

$$c_j - A_j^T \pi = 0 \quad \text{for all } j \in P, \tag{3.4a}$$
$$c_j - A_j^T \pi \geq 0 \quad \text{for all } j \in V, \tag{3.4b}$$
$$c_j - A_j^T \pi \geq 0 \quad \text{for all } j \in I. \tag{3.4c}$$

It is easy to see that the constraints 3.4b are satisfied when $\bar{x}_{\text{C}}$ is optimal for (RP). In this case, constraints 3.4b are redundant and may be removed. In the other case, when $\bar{x}_{\text{C}}$ is not optimal, we can handle them subsequently in the next reduced problem.

To find a negative reduced cost set of incompatible variables, the authors of [9] propose to

$$\underset{\pi, y}{\text{maximize}} \; y \quad \text{subject to } \; c_{\text{P}} - A_{\text{P}}^T \pi = 0, \; c_{\text{I}} - A_{\text{I}}^T \pi \geq y. \tag{3.5}$$

By using the same partitioning as A_{P} (see equation (3.3)), we partition the vector of dual variables $\pi = \begin{bmatrix} \pi^{\text{P}} \\ \pi^{\text{N}} \end{bmatrix}$.

Introducing this notation into the first set of constraints of (3.5), we obtain

$$c_{\text{P}} - (A_{\text{P}}^{\text{P}})^T \pi^{\text{P}} - (A_{\text{P}}^{\text{N}})^T \pi^{\text{N}} = 0,$$

and we may thus express

$$\pi^{\text{P}} = (A_{\text{P}}^{\text{P}})^{-T} c_{\text{P}} - (A_{\text{P}}^{\text{P}})^{-T} A_{\text{P}}^{\text{N}} \pi^{\text{N}}.$$

Substituting the latter into the second set of constraints, (3.5) may be rewritten as the *complementary problem*

$$\underset{\pi \in \mathbb{R}^p, y}{\text{maximize}} \; y \quad \text{subject to } \; y - \left(A_{\text{P}}^{\text{N}} (A_{\text{P}}^{\text{P}})^{-1} A_{\text{I}}^{\text{P}} - A_{\text{I}}^{\text{N}}\right) \pi^{\text{N}} \leq \left(c_{\text{I}} - ((A_{\text{P}}^{\text{P}})^{-1} A_{\text{I}}^{\text{P}})^T c_{\text{P}}\right). \tag{CP}$$

The following property [6, 9] justifies the use of (CP).

Proposition 3.1 *Let x_C^* be an optimal solution of (RP) and let y^* be an optimal solution of (CP). Then $(x_C^*, x_I^*) = (x_C^*, 0)$ is an optimal solution of (LP) if and only if $y^* \geq 0$.*

It is informative to consider the dual of (CP), the *simplified dual*

$$\underset{v}{\text{minimize}} \; \left(c_{\text{I}} - ((A_{\text{P}}^{\text{P}})^{-1} A_{\text{I}}^{\text{P}})^T c_{\text{P}}\right)^T v \quad \text{subject to } \; \left(A_{\text{P}}^{\text{N}} (A_{\text{P}}^{\text{P}})^{-1} A_{\text{I}}^{\text{P}} - A_{\text{I}}^{\text{N}}\right) v = 0, \; e^T v = 1, \; v \geq 0. \tag{SD}$$

Note that (SD) possesses $m - p + 1$ equality constraints. From now on, we refer to the *complementary problem* as the pair (CP) and (SD). Theorem 2.1 links (SD) with (LP).

We mention that the matrix $A_{\text{P}}^{\text{N}} (A_{\text{P}}^{\text{P}})^{-1} A_{\text{I}}^{\text{P}} - A_{\text{I}}^{\text{N}}$ of the complementary problem is the lower right-hand matrix of QA obtained by the previous reduction. Thus, we do not need to compute this matrix at each solution of (SD). We just have to update the matrix after each augmentation or each reduction of the reduced problem.

3.3 CPLEX Reduction

To avoid possible numerical tolerance incompatibilities between UMFPACK [5] and CPLEX, we use the latter to execute the reduction.

Algorithm 3.1 CPLEX reduction of a problem that does not contain bounds on variables.

Step 0. Create a temporary CPLEX problem (LPtmp) that contains only the nonzero variables.

Step 1. Add to (LPtmp) an identity matrix that represents artificial variables.

Step 2. Solve (LPtmp).

Step 3. The rows associated with a basic artificial variable must be removed.

Step 4. Use the basis inverse of (LPtmp) to construct (RP) and (SD).

Algorithm 3.1 describes the method of reducing problems that do not contain bounds on variables. Step 0 and Step 1 are clear. At Step 2, we solve the temporary problem. The optimal basic feasible solution of this problem contains the nonzero variables of the current solution of (LP), that is (A_{P}), and the artificial variables associated with the rows that are not associated with a nonzero basic variable, that is (A_{N}). At Step 3, we identify the subsets of rows: A^{N} are the rows associated with artificial variables and A^{P} are the rows associated with nonzero basic variables. At Step 4, we construct the reduced and complementary problems. Since the required Q matrix is given by the current inverse basis of the temporary problem, we can compute QA to create both problems. We might add that the computational time to find Q is relatively insignificant compared with that needed to calculate QA.

3.4 Bounds

We mentioned previously that we generalized our algorithm to handle problems with bounded variables. As we explained in §2.1, variables that are at their upper bounds can be handled as zero variables. Thus, since these problems may have more degeneracy, their reduced problems may be smaller. Our algorithm must be modified in the reduction process and in the composition of (SD).

We define the index sets L and U:

$$L = \{i \in C \cup I \mid \bar{x}_i = l_i\} \quad \text{and} \quad U = \{i \in C \cup I \mid \bar{x}_i = u_i\}.$$

where $\bar{x}_i$ is the current value of x_i.

To take into account the bounds of the variables, we must add some steps in Algorithm 3.1. The procedure to reduce problems that contain bounds with CPLEX is summarized in Algorithm 3.2.

Step 0, Step 3, Step 4 and Step 5 of Algorithm 3.2 are the same as in Algorithm 3.1. Step 1 assures that the optimal solution of (LPtmp) is the same as the feasible solution given. Step 2 allows maximizing the number of artificial variables in the basis, that is, allows completing reduction of the problem.

Step 6 is executed as follows. If a variable x_i with $i \in L$ is incompatible, instead of deleting it from (RP), we "remove" it by changing its bounds in (RP) such that $l_i \leq x_i \leq l_i$. Thus, the values of these incompatible variables cannot be changed in the reduced problem as null incompatible variables. Moreover, by changing bounds, CPLEX will take into account the values of x_i in the right-hand side of the reduced problem. The complementary problem is constructed as usual for this type of variable.

In the same way, if a variable x_i with $i \in U$ is incompatible, instead of deleting it from (RP), we "remove" it by changing its bounds in (RP) such that $u_i \leq x_i \leq u_i$. However, these variables are modified in the complementary problem. Since the theoretical substitution of this type of variable is $y_i = u_i - x_i$, we take into account the negative relation between y_i and x_i by multiplying the coefficient of this variable in

Algorithm 3.2 CPLEX reduction of problems that contain bounds on variables.

Step 0. Create a temporary CPLEX problem (LPtmp) that contains only the nonzero variables.

Step 1. Change the bounds of variables in (LPtmp):
$l_i \leq x_i \leq l_i$ for all $i \in L$.
$u_i \leq x_i \leq u_i$ for all $i \in U$.

Step 2. Make nonbasic the variables x_i for all $i \in L \cup U$ in (LPtmp).

Step 3. Add to (LPtmp) an identity matrice that represents artificial variables.

Step 4. Solve (LPtmp).

Step 5. The rows associated with a basic artificial variable must be removed.

Step 6. Use the basis inverse of (LPtmp) to construct (RP) and (SD).

(SD) by -1. Note that u_i or l_i in the substitution are present indirectly in (SD) by the modification of the bounds in (RP).

We then obtain the bounded reduced problem (BRP)

$$\underset{x}{\text{minimize}} \sum_{i\in C} c_i x_i + \sum_{j\in I\cap L} c_j x_j + \sum_{k\in I\cap U} c_k x_k \tag{BRP}$$

$$\sum_{i\in C} (A_{\text{P}}^{\text{P}})^{-1} A_i^{\text{P}} x_i + \sum_{j\in I\cap L} (A_{\text{P}}^{\text{P}})^{-1} A_j^{\text{P}} x_j + \sum_{k\in I\cap U} (A_{\text{P}}^{\text{P}})^{-1} A_k^{\text{P}} x_k = Q^{\text{P}} b$$

$$\begin{aligned} l_i \leq x_i \leq u_i \quad &\text{for all} \quad i \in C \\ l_j \leq x_j \leq l_j \quad &\text{for all} \quad j \in I \cap L \\ u_k \leq x_k \leq u_k \quad &\text{for all} \quad k \in I \cap U \end{aligned}$$

and the associated simplified problem (ASD)

$$\underset{v}{\text{minimize}} \sum_{i\in I\setminus U} \left(c_i - ((A_{\text{P}}^{\text{P}})^{-1} A_i^{\text{P}})^T c_{\text{P}}\right) v_i - \sum_{j\in I\cap U} \left(c_j - ((A_{\text{P}}^{\text{P}})^{-1} A_j^{\text{P}})^T c_{\text{P}}\right) v_j \tag{ASD}$$

$$\sum_{i\in I\setminus U} \left(A_{\text{P}}^{\text{N}} (A_{\text{P}}^{\text{P}})^{-1} A_i^{\text{P}} - A_i^{\text{N}}\right) v_i - \sum_{j\in I\cap U} \left(A_{\text{P}}^{\text{N}} (A_{\text{P}}^{\text{P}})^{-1} A_j^{\text{P}} - A_j^{\text{N}}\right) v_j = 0$$

$$\sum_{i\in I} v_i = 1$$

$$v_{i\in I} \geq 0.$$

When a variable x_i with $i \in I \cap (L \cup U)$ is chosen to enter (RP), we enter the rows as usual and we enter the variable by re-initializing the bounds of x_i since the latter variable is already in (RP).

4 Numerical Results

This section presents numerical results that were obtained with IPS-3. More precisely, we present a comparison of reduction times obtained by CPLEX and UMFPACK. We compare solution time of CPLEX when starting with an initial basic feasible solution and an initial feasible solution. Then, we present a sensitivity analysis

as a function of the quality of the initial solution. Finally, we present numerical results on problems with bounded variables. Characteristics of instances used are presented in §4.1 and §4.5.

Before the presentation of the results, we define the different versions of IPS to avoid confusion. IPS has been proposed and developed by Elhallaoui et al. [9] and is the basic algorithm. IPS-2 has been developed by Raymond et al. [16] and contains different improvements. Here, we present IPS-3, a version based on IPS-2 which has the improvements that we mentioned in §3. All the results in this paper have been computed with IPS-3.

4.1 Vehicle and Crew Scheduling Data Set

We selected a number of instances of *combined vehicle and crew scheduling problems in public transit* (VCS) which exhibit important degeneracy. These instances were generated by Elhallaoui et al. [9] using a random generator of Haase [13] and was also used in [16].

Table 4.1 reports the number of constraints and variables of each instance as well as the average percentage of degenerate basic variables in (RP) encountered in the course of the iterations of Algorithm 2.1.

Table 4.1: Characteristics of VCS instances.

instance	constraints	variables	degeneracy	instance	constraints	variables	degeneracy
VCS1	2084	10343	44%	VCS6	2084	8308	48%
VCS2	2084	6341	45%	VCS7	2084	8795	47%
VCS3	2084	6766	45%	VCS8	2084	9241	47%
VCS4	2084	7337	48%	VCS9	2084	10150	50%
VCS5	2084	7837	48%	VCS10	2084	6327	45%

4.2 CPLEX Reduction Instead of UMFPACK Reduction

When we compare a CPLEX reduction to the same UMFPACK reduction, the time reduction is significant. Moreover, the implementation of IPS-3 is simplified since it uses only one commercial software package. However, the use of CPLEX instead of UMFPACK to create the reduced problem and the complementary problem results in a relatively small gain in the total computing time.

We present in Table 4.2 the CPLEX and the UMFPACK reduction time of the first reduced problem for each VCS instance. All the times are in seconds. We see that the CPLEX reduction is 1.48 times faster than the UMFPACK reduction.

Table 4.2: Reduction times of UMFPACK and CPLEX for VCS instances.

instance	UMFPACK	CPLEX	instance	UMFPACK	CPLEX
VCS1	6.58	3.87	VCS6	4.49	3.45
VCS2	3.53	2.34	VCS7	4.72	3.45
VCS3	3.69	2.54	VCS8	4.27	3.16
VCS4	3.83	2.64	VCS9	5.55	3.53
VCS5	4.60	2.84	VCS10	3.47	2.41
			AVERAGE	4.47	3.02

4.3 Initial Feasible Solution Instead of Initial Basic Feasible Solution

Beginning with an initial feasible solution instead of an initial basic feasible solution increases the generality of IPS-3. Indeed, it can begin with an initial feasible solution obtained by an heuristic or begin with an initial basic feasible solution.

We present in Table 4.3 the solution time of CPLEX and IPS-3 when they start with initial solutions obtained from the phase I basis. The times are again in seconds.

The average solution times of CPLEX and IPS-3 are 177 and 42 seconds respectively. The reduction factor is 4.17 on average. Note that the times of IPS-3 are similar to those of IPS-2 (see [16]) since the latter does not include the phase I time used to find the initial basis.

In short, starting with an initial feasible solution increases the performance time of CPLEX while augmenting the generality of IPS. The performance time of the latter is decreased when the initial feasible solution is given since no computational time is needed to obtain an initial basic feasible solution (phase I) or to complete it.

Table 4.3: CPLEX and IPS-3 solution times when starting with an initial solution.

instance	CPLEX	IPS-3	reduction factor	instance	CPLEX	IPS-3	reduction factor
VCS1	252	58	4.34	VCS6	176	45	3.91
VCS2	105	33	3.18	VCS7	185	44	4.20
VCS3	135	35	3.85	VCS8	214	44	4.86
VCS4	163	35	4.65	VCS9	246	58	4.24
VCS5	171	39	4.38	VCS10	124	30	4.13
				AVERAGE	177	42	4.17

4.4 Solution Time as a Function of the Quality of the Initial Solution

As we stated in §3.1, the theory allows us to begin with an initial feasible solution. This subsection presents the results of IPS-3 on VCS instances when our algorithm starts with different initial solutions. These initial solutions have been generated by CPLEX and their costs are from 0.5 percent to 7 percent more than those of the optimal solution. To generate these initial basic feasible solutions, we first find the optimal solution. We then execute CPLEX from phase I and write the basis in a file when we reach a solution whose objective value has the predetermined gap value compared to the optimal solution.

As we show in Figures 4.1 and 4.2, our method is more stable when compared to CPLEX. Indeed, we see that the results of IPS-3 are almost always better when the initial solution is better. We can surely say that IPS-3 takes advantage of a good initial solution. By contrast, the results with CPLEX on VCS instances show that the initial solution can handicap the method.

4.5 Fleet Assignment With Bounded Variables Data Set

Fleet assignment (FA) problems consist in maximizing the profits of assigning a type of aircraft to each flight segment over a horizon of one week. The paths of the aircrafts must respect maintenance conditions and availability. Our problem instances are generated from real data with 2505 flight segments and four types of aircraft. The variables are flight sequences between maintenance. Those problems have one set partitioning constraint per flight segment, one availability constraint per aircraft type, and one flow conservation constraint between flight sequences at the maintenance base. Some variables are also bounded above.

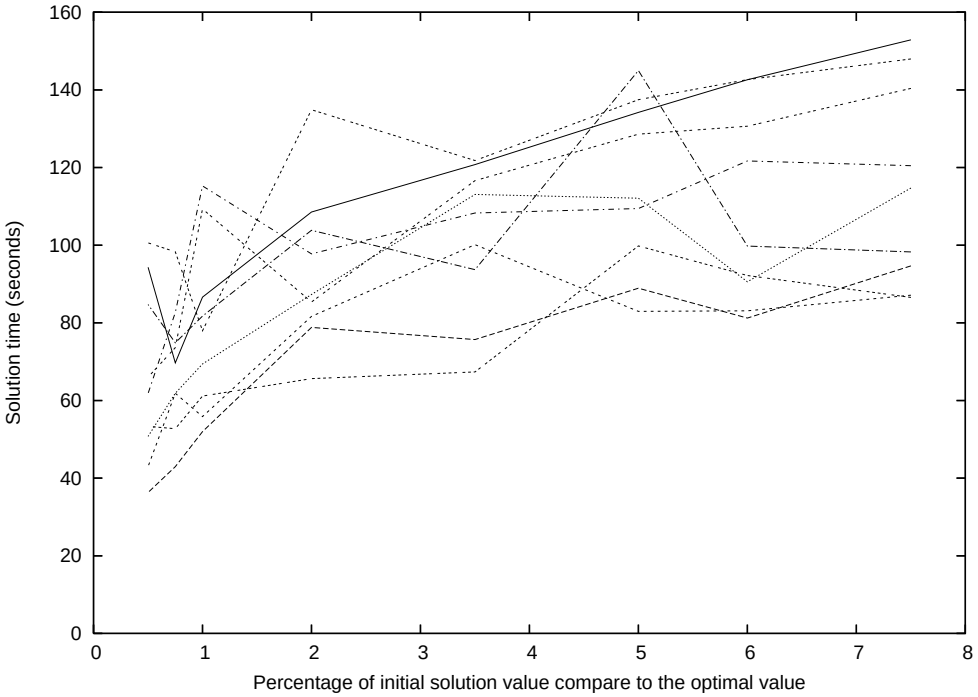

Figure 4.1: Results of CPLEX on VCS instances

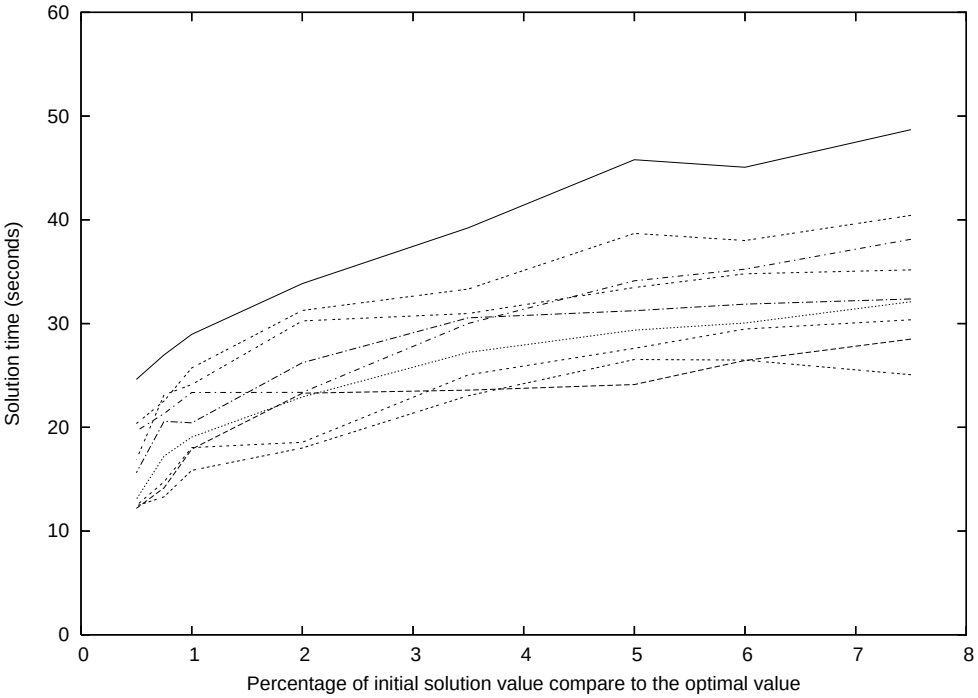

Figure 4.2: Results of IPS-3 on VCS instances

Those problems are not so large but there are some typical real-life master problems that need to be solved at each iteration of a column generation algorithm imbedded in a *branch & bound* procedure. These instances were used in [16] and [10].

To test IPS-3 on problems that contain bounds on variables, we modify our FA instances. We add upper bounds of 1 on each variable, i.e., $x_i \leq 1, \quad i = \{1, \ldots, n\}$. The resulting instances are called UBFA. We choose these instances instead of the VCS because there is a significant number of variables with value of 1 in optimal solutions of FA instances. Consequently, adding upper bounds of 1 on variables in FA instances augments degeneracy.

Table 4.4 gives the number of constraints and variables of each instance along with the average percentage of degenerate variables encountered with IPS-3.

Table 4.4: Characteristics of UBFA instances .

			degeneracy					degeneracy	
instance	constraints	variables	$x_i = 0$	$x_i = 1$	instance	constraints	variables	$x_i = 0$	$x_i = 1$
UBFA6	5067	17594	68%	12%	UBFA13	5159	25746	65%	15%
UBFA7	5159	20434	59%	11%	UBFA14	5159	22641	71%	15%
UBFA8	5159	21437	65%	14%	UBFA15	5182	23650	63%	13%
UBFA9	5159	23258	66%	14%	UBFA16	5182	23990	64%	13%
UBFA10	5159	24492	66%	14%	UBFA17	5182	24282	65%	14%
UBFA11	5159	24812	66%	14%	UBFA18	5182	24517	65%	14%
UBFA12	5159	24645	66%	14%	UBFA19	5182	24875	65%	14%

4.6 Results for UBFA Data Set

We used UBFA instances to test IPS-3. To start the algorithms, we find initial feasible solutions through the phase I of CPLEX. These initial solutions are at 1.5 percent of the optimal solutions on average.

Table 4.5 presents the computational time in seconds for CPLEX and IPS-3 for solving the instances. The reduction factor is the CPLEX time divided by the IPS-3 time. These factors show that on average IPS-3 is 20 times faster than CPLEX for solving problems that contain bounds on variables. This reduction factor is very significant when such problems need to be solved thousands of times in a column generation scheme embedded in a *branch & bound* procedure.

These impressive performances of our algorithm can be explained as follow. First, we saw in §3.4 that the variables whose values are equal to their upper bounds are handled like zero variables. In other words, the addition of upper bounds in FA instances increases degeneracy. As a result, there are more degenerate

Table 4.5: Results on UBFA instances.

			reduction				reduction
instance	CPLEX	IPS-3	factor	instance	CPLEX	IPS-3	factor
UBFA6	581	43	13.51	UBFA13	1362	62	21.97
UBFA7	643	40	16.08	UBFA14	1590	76	20.92
UBFA8	690	38	18.16	UBFA15	1242	45	27.60
UBFA9	1083	58	18.67	UBFA16	1340	57	23.51
UBFA10	948	69	13.74	UBFA17	924	60	15.40
UBFA11	1384	50	27.68	UBFA18	1049	56	18.73
UBFA12	1731	61	28.38	UBFA19	1169	62	18.85
				AVERAGE	1124	55.5	20.23

pivots in the CPLEX solution and greater computational time. Secondly, our algorithm takes advantage of degeneracy. As result, we treat smaller reduced problems and faster solution time relative to CPLEX. Starting with an initial feasible solution instead of an initial basic feasible solution also can be a part of the explanation as well as the reduction now executed by CPLEX.

5 Conclusion

We proposed algorithm IPS-3: a theoretical and implementational generalization of IPS. First, we rewrote the theory so that the algorithm can start with a feasible solution instead of a basic feasible solution. We can claim that IPS-3 takes advantage of initial solutions whereas CPLEX is less predictable. Moreover, we simplified the implementation by using CPLEX instead of UMFPACK to create the reduced and the complementary problems. Then, we added procedures to handle problems with bounded variables. The numerical results on fleet assignment instances with bounded variables show the efficiency of IPS-3. Indeed, we obtain on average a reduction factor of 20 on the total computing time compared to CPLEX.

References

[1] Bland, R.G. (1977). New Finite Pivoting Rules for the Simplex Method. *Mathematical of Operations Research*, **2**, 103–107.

[2] Boubaker, K., G. Desaulniers and I. Elhallaoui (2008). Bidline Scheduling with Equity by Heuristic Dynamic Constraint Aggregation. *Les Cahiers du GERAD*, Montreal, Canada. G–2008–43.

[3] Charnes, A. (1952). Optimality and Degeneracy in Linear Programming. *Econometrica*, **20**, 160–170.

[4] Dantzig, G.B. (1963). *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ.

[5] Davis, T.A. and I.S. Duff (1997). An Unsymmetric-Pattern Multifontal Method for Sparse LU Factorization. *Siam Journal on Matrix Analysis and Applications*, **18(01)**, 140–158.

[6] Elhallaoui, I., D. Villeneuve, F. Soumis and G. Desaulniers (2005). Dynamic Aggregation of Set Partitioning Constraints in Column Generation. *Operations Research* **53**, 632–645.

[7] Elhallaoui, I., A. Metrane, F. Soumis and G. Desaulniers (2008). Multi-phase Dynamic Constraint Aggregation for Set Partitioning Type Problems. *Mathematical Programming*.

[8] Elhallaoui, I., G. Desaulniers, A. Metrane and F. Soumis (2006). Bi-dynamic Constraint Aggregation and Subproblem Reduction. *Computers and Operations Research*, doi:10.1016/j.cor.2006.10.007.

[9] Elhallaoui, I., A. Metrane, G. Desaulniers and F. Soumis (2008). An Improved Primal Simplex Algorithm for Degenerate Linear Programs. *Submitted to SIAM Journal of Optimization*.

[10] Lacasse-Guay, E. *Ph.D. Thesis*, École Polytechnique Montreal, Canada.

[11] Filiz, B., Csizmadia, Z. and T. Illés (2001). Anstreicher-Terlaky Type Monotonic Simplex Algorithms for Linear Feasibility Problems. *Operations Research Report***35**, 286–303.

[12] Fukuda, K. (1982). Oriented Matroid Programming. *Ph.D. Thesis*, University of Waterloo, Canada.

[13] Haase, K., G. Desaulniers, and J. Desrosiers (2001). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science* **35**, 286–303.

[14] Omer, J. (2006). Méthode de Réduction Dynamique de Contraintes pour un Programme Linéaire. *Masters Thesis*, École Polytechnique de Montréal.

[15] Pan, P.-Q. (1998). A Basis Deficiency-Allowing Variation of the Simplex Method for Linear Programming. *Computers and Mathematics with Applications*, **36(3)**, 33–53.

[16] Raymond, V., F. Soumis and D. Orban (2008). A New Version of the Improved Primal Simplex for Degenerate Linear Programs. *Submetted to Computer and Operational Research*.

[17] Ryan, D. M. and Osborne, M. (1988). On the Solution of Highly Degenerate Linear Programmes. *Mathematical Programming*, **41**, 385–392.

[18] Terlaky, T. and Sushong, Z. (1993). Pivot Rules for Linear Programming : A Survey on Recent Theoretical Developments. *Annals of Operations Research*, **46**, 203–233.

[19] Wolfe, P. (1963). A Technique for Resolving Degeneracy in LP. *SIAM Journal*, **2**, 205–211.