Noname manuscript No.
(will be inserted by the editor)

A triangulation and fill-reducing initialization procedure for the simplex algorithm

Nikolaos Ploskas · Nikolaos V. Sahinidis · Nikolaos Samaras

Received: date / Accepted: date

Abstract The computation of an initial basis is of great importance for simplex algorithms since it determines to a large extent the number of iterations and the computational effort needed to solve linear programs. We propose three algorithms that aim to construct an initial basis that is sparse and will reduce the fill-in and computational effort during LU factorization and updates that are utilized in modern simplex implementations. The algorithms rely on triangulation and fill-reducing ordering techniques that are invoked *prior* to LU factorization. We compare the performance of the CPLEX 12.6.1 primal and dual simplex algorithms using the proposed starting bases against CPLEX using its default crash procedure over a set of 95 large benchmarks (NETLIB, Kennington, Mészáros, Mittelmann). The best proposed algorithm utilizes METIS [30], produces remarkably sparse starting bases, and results in 5% reduction of the geometric mean of the execution time of CPLEX's primal simplex algorithm. Although the proposed algorithm improves CPLEX's primal simplex algorithm across all problem types studied in this paper, it performs better on hard problems, i.e., the instances for which the CPLEX default requires over $1,000$ seconds. For these problems, the proposed algorithm results in 37% reduction of the geometric mean of the execution time of CPLEX's primal simplex algorithm. The proposed algorithm also reduces the

N. Ploskas
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
E-mail: nploskas@andrew.cmu.edu

N. V. Sahinidis
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
Tel.: +1 (412) 268-3338, Fax: +1 (412) 268-7139
E-mail: sahinidis@cmu.edu. Address all correspondence to this author.

N. Samaras
Department of Applied Informatics, University of Macedonia, Thessaloniki 54636, Greece
E-mail: samaras@uom.gr

execution time of CPLEX's dual simplex on hard instances by 10%. For the instances that are most difficult for CPLEX, and for which CPLEX experiences numerical difficulties as it approaches the optimal solution, the best proposed algorithm speeds up CPLEX by more than 10 times. Finally, the proposed algorithms lead to a natural way to parallelize CPLEX's dual simplex code with speedups of 1.2 and 1.3 on two and four cores, respectively.

Keywords Linear programming · Revised simplex algorithm · Initial basis · Crash procedure

1 Introduction

Since the introduction of the simplex algorithm in 1947 [9,10], Linear Programming (LP) has been widely used in many application areas in science and engineering and led to the genesis of the mathematical programming community [31]. Since that time, a variety of algorithmic and computational techniques have been developed to improve the computational performance of the simplex algorithm:

- presolve methods that reduce the problem size [47,36,24] (for a review, see [3]).
- scaling techniques that improve the numerical behavior of the simplex algorithm and reduce the number of iterations required to solve LPs [8,45,17] (for a review, see [43]).
- pivoting rules that reduce the number of simplex iterations required to solve LPs [26,19,44] (for a review, see [42]).
- basis factorization and update methods that improve the numerical behavior of the simplex algorithm and reduce its execution times [33,20,22] (for reviews, see [15,16]).

The simplex algorithm starts with a feasible basis and uses pivot operations in order to preserve feasibility of the basis and guarantee monotonicity of the objective value. In some very simple cases, a basic feasible solution may be available.

The quality of the initial basis greatly affects the execution time, the number of iterations, and the required storage of the algorithm's data structures [5,23,34,35,40]. The aim of the crash procedures is to find an initial basis that: (i) is close to optimal, (ii) is sparse, (iii) will reduce the subsequent fill-ins of the LU factorization, (iv) will reduce the execution time per iteration, and (v) will reduce the number of iterations. Crash procedures may sometimes increase the number of iterations but they may also achieve a decrease in the time per iteration and the overall execution time. Most crash procedures use triangulation and sparsification concepts. Considering that the initial basis will be factorized using LU decomposition, most crash procedures form a nearly-triangular and sparse basis that is likely to limit the number of subsequent fill-ins.

Considerable attention has been given to the initialization of the simplex algorithm since its conception. Most linear programming textbooks [7,4,34] present only simple initialization procedures, such as the all-artificial and the slack-artificial basis. Twelve different initialization techniques have been developed for general LPs; six additional techniques have been developed for LPs with special structure. Most notably, advanced crash procedures for initializing the simplex algorithm have been proposed in [6,37,23,5,35]. Initialization procedures that can be applied in special cases or in modified simplex-type algorithms have been presented in [25,32,28,38,1,39]. All these crash procedures will be reviewed in detail in Section 2.

This paper proposes new methods for initializing the simplex algorithm. The overall goal of these methods is to exploit the concepts of triangulation and sparsification in order to create a nearly-triangular and sparse basis that will limit the number of fill-ins of the LU factors of the bases generated by the simplex algorithm. The triangulation step is achieved via permutation of column singletons of the LP problem matrix to identify a maximal submatrix that includes columns of the identity matrix. The sparsification step relies on fill-reducing strategies that have been devised to minimize the maximum potential fill-in in LU factorization procedures. These fill-reducing strategies have been designed for factorizing symmetric matrices in the context of LU factorization. However, for crash procedures based on these strategies, the impact on the performance of modern simplex codes is unknown. Given the obvious relative advantages and disadvantages of starting points that are sparse but far from optimal versus starting points that are less sparse but nearly-optimal, we propose to investigate the impact of these strategies computationally. We thus apply them to the nonsingleton columns of the constraint matrix for the purpose of supplementing column singletons with additional columns that are likely to lead to minimal fill-in in the subsequent LU factorization and update procedures during simplex iterations. In general, finding a permutation matrix that minimizes fill-in is NP-complete [46]. For this reason, heuristics are used to find good orderings. In this paper, we experiment with three different fill-reducing ordering methods: (i) COLAMD [13], (ii) AMD [2], and (iii) METIS [30]. Even though these techniques have not been considered in the numerical linear algebra of the simplex algorithm, we will demonstrate that they can provide starting bases that, in comparison to existing implementations, are sparser and reduce the fill-in and computational effort during LU factorization and updates for many LPs.

The remainder of this paper is organized as follows. In Section 2, we review procedures for finding an initial basis. Section 3 presents the proposed methods. Section 4 presents results from an extensive computational study that compares the performance of the proposed methods against the default CPLEX crash procedure. Conclusions are provided in Section 5.

2 Review of crash procedures

The aim of a crash procedure is to find an initial basic solution. The starting basis may be feasible or infeasible. In case the basis is feasible ($l_B \leq x_B \leq u_B$, where B is the set of the basic variables, l and u are the lower and upper bounds of the variables) simplex algorithms can use it as a starting solution and proceed to find a solution of the problem. On the other hand, if the initial basis is not feasible, different methods can be used to find a basic feasible solution. Three methods are primarily used: (i) the two-phase method, (ii) the big-M method, and (iii) the single artificial variable method. Modern implementations of the simplex algorithm use the two-phase method.

Let's assume that the LP is in the so called computational form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & l \leq x \leq u \end{aligned}$$

where $c, l, x, u \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$, and T denotes transposition. Assume that A has full row rank and contains (implicitly) an identity matrix.

The two-phase method adds an artificial variable to each constraint and solves an auxiliary LP in Phase I:

$$\begin{aligned} \min \quad & e^T y \\ \text{s.t.} \quad & Ax + I_m y = b \\ & x, y \geq 0 \end{aligned}$$

where $e \in \mathbb{R}^m$ is a vector of ones and I_m is an identity matrix of size $m \times m$. The auxiliary LP is solved using the simplex algorithm. If $y \neq 0$ at optimality, then the original LP is infeasible. If $y = 0$, then there are two possibilities:

- $y = 0$ and no auxiliary variable is in the basis: in this case, we have identified a basic feasible solution $x = [x_B, x_N]^T$, where B is the set of the basic variables and N is the set of the nonbasic variables. The nonzero elements in x form x_B; the remaining form x_N. We can solve the original LP starting with this basic feasible solution after eliminating the artificial variables and the corresponding columns from the problem.
- $y = 0$ and at least one auxiliary variable is still in the basis: in this case, we have identified a degenerate solution to the auxiliary problem. We remove the artificial variables from the basis. If the lth variable is an artificial variable, examine the lth element of the columns $A_B^{-1} A_{.j}$, $j = 1, \cdots, n$. If the lth element of the jth column is nonzero, then apply a change of basis with the lth entry serving as the pivot element. The lth basic variable exits the basis and variable x_j enters the basis.

If the initial basis is not feasible, LP solvers search for a feasible point during Phase I. Hence, a crash procedure that produces feasible starting bases avoids Phase I and may lead to fewer simplex iterations. Nonetheless, the problem of finding a feasible point has the same complexity bound as the linear programming problem [41].

The simplest initial basis is the all-artificial basis or all-logical basis, presented in most linear programming textbooks [7, 4, 34]. Artificial variables are added to all constraints and the initial basis consists of the artificial variables. The all-artificial basis is extremely simple and has three distinct advantages [34]: (i) its creation is instantaneous, (ii) the LU decomposition of the starting basis (I) is available, and (iii) the first iterations are very fast as the operations utilize a very sparse LU factorization. Another simple initial basis is the slack-artificial basis [5]. Initially, we add slack and surplus variables to all inequality constraints. Then, we add artificial variables to equality constraints and inequality constraints of the type $\geq$. The initial basis is formed by the slack variables added in inequality constraints of type $\leq$ and the artificial variables. The slack-artificial basis is better than the all-artificial basis since it adds fewer artificial variables and solves a smaller LP in Phase I. The techniques discussed in this paragraph are known to lead to substantially larger numbers of iterations than other initialization techniques.

A variant of the slack-artificial initial basis is the feasible slack basis [5]. In this method, we add slack and surplus variables to all inequality constraints. Then, we add artificial variables to all constraints and form an initial basis consisting of only the artificial variables. Next, available slacks that are initially nonnegative replace the artificial variables in the basis. Bixby [5] also proposed an approach to create a sparse and well-behaved basis with as few artificial variables as possible. The generated basis includes all slack variables. The remainder of the structural variables are assigned a preference order of inclusion in the basis; this preference order aims to place the variables with the most freedom at the start of the list using the objective function to break ties. Then, a heuristic procedure selects the variables that will be included in the basis aiming to form a nearly triangular basis. Bixby's computational results suggested that his basis can greatly reduce the number of iterations, especially for easy problems, but it is generally less effective for harder problems.

Carstens [6] classifies crash procedures into two classes: *GAIN switch on* and *GAIN switch off*. In the GAIN switch off case, the objective function is ignored and the starting basis is chosen based on sparsity grounds alone. Carstens assumes that a starting set of basic variables is given as input to the crash procedure. It may consist entirely of artificial variables in case there is no information about selecting basic variables. At each iteration of these crash procedures, a pivot element a_{ij} is selected to replace column i of B with column j of A. If column j has c_j nonzeros and row i has r_i nonzeros, Carstens discusses three different ways to select a pivot element:

- order the nonbasic columns in order of increasing c_j and choose the pivot element a_{ij} to be a nonzero that minimizes r_i.
- order the rows in order of increasing r_i and choose the pivot element a_{ij} to be a nonzero that minimizes c_j for j nonbasic.
- consider the nonzeros in increasing order of the count $(r_i - 1)(c_j - 1)$ (Markowitz criterion for reinversion [33]).

In the GAIN switch on case, a basis change is made only if it leads to an improvement in the objective function. Carstens recommends the use of the GAIN switch off when the starting basis is totally or mostly artificial and the

GAIN switch on when the starting basis includes few artificial variables. Reid developed an algorithm, presented by Gould and Reid [23], that forms an upper triangular basis. In comparison to Carsten's GAIN switch off algorithm, a column that is chosen late in Reid's algorithm is required to have a nonzero in at least one row that has not yet been pivotal. Gould and Reid [23] proposed a tearing crash procedure that aims to find an initial basis that is as feasible as possible and can be calculated with a reasonable computational effort. The approach relies on the P5 algorithm of Erisman et al. [18] and solves a series of small LPs, the solution of which forms a basis for the initial LP. Maros and Mitra [35] proposed four crash procedures: (i) CRASH(LTSF): a lower triangular symbolic crash procedure designed for feasibility, (ii) CRASH(ADG): an anti-degeneracy crash procedure that deals with LPs where a starting basis may lead to a primal degenerate solution, (iii) CRASH(ART): an artificial removal technique used after CRASH(LTSF), and (iv) CRASH(SOR): an iterative crash procedure based on Kaczmarz's SOR algorithm [29]. MINOS [37] contains a crash procedure where a pivot a_{ij} is selected if its row contains zeros in all the columns that have so far been chosen as basic or if its column contains zeros in all the rows that have been pivotal.

Al-Najjar and Malakooti [1] use a Phase I method that moves through the interior of the feasible region to obtain an initial basic feasible solution. Gülpinar et al. [25] proposed a method to construct an initial basis for LPs with embedded pure network structures. Junior and Lins [28] estimate an optimal (or near-optimal) basis by finding constraints which intersect the gradient plane at minimal angles. Luh and Tsaih [32] developed a search direction that combines the gradient direction and an internal pointing direction with respect to the polyhedron forming the feasible region. Nabli [38] proposed a method for constructing an initial feasible solution from an infeasible one. This method operates without artificial variables and without any perturbation in the objective function. Feasibility is obtained via a modification of the structure of the simplex algorithm in the choice of the entering and leaving variables. Nabli and Chahdoura [39] presented a crash procedure that does not involve any artificial variables and can also detect redundant constraints and infeasibility.

The majority of the state-of-the-art crash procedures focus on finding an initial basis that is as close to optimality as possible without aiming to create a sparse initial basis that will limit the number of fill-ins of the LU factors of the bases generated by the simplex algorithm. In this paper, we investigate whether it may be better–at least in certain cases–to rely on a crash procedure that aims to choose an initial basis in a way that will be very sparse and nearly triangular. Even though it is counter-intuitive that it would be advantageous to use a crash procedure that ignores the objective function, a sparse and near triangular initial basis is more likely to minimize the subsequent fill-ins during the LU factorization of the simplex bases. All state-of-the-art LP solvers apply such techniques to factorize bases in the course of the algorithm. Our proposal is to utilize these techniques *also* for the construction of the initial basis and investigate the computational impact of this approach on primal and dual simplex algorithms.

3 The proposed algorithms

In this section, we present three algorithms to construct an initial basis for the simplex algorithm. All proposed algorithms ignore the objective function and the bounds of the variables and choose the initial basis in a way that it will be very sparse and nearly triangular. The motivation of these algorithms is to quickly find a starting basis that is likely to minimize subsequent fill-ins during the LU factorization of the simplex bases. The first step in all algorithms is to identify a maximal submatrix of A that is a diagonal. In particular, if a column singleton a_{ij} exists, its column j is permuted to the left and its row i is permuted to the top. Column j and row i are removed from A. Such singleton columns must be present in the original constraint matrix A, not just in the matrix remaining once pivoted rows and columns have been removed. The process repeats until no more singletons exist, leading to

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

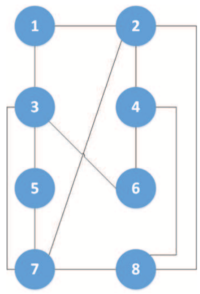
where A_{11} is a diagonal matrix whose diagonal entries are greater than the smallest acceptable pivot value $\tau > 0$. The computational effort of this procedure depends on the kinds of data structures used. In one implementation, the time to find all singletons and permute them to the top left corner of the constraint matrix is reported to be $O(n + |A_{11}| + |A_{12}|)$ [13], where $|A|$ denotes the number of nonzeros of matrix A. If the sum of the number of $\leq$ type of constraints and the singleton columns in the original LP problem is m, initialization stops here with a basis consisting of all slack variables and/or variables with singleton columns.

Once singletons are removed, the remaining matrix A_{22} is ordered with a fill-reducing ordering method. The goal of this procedure is to find a column permutation of A_{22} so that subsequent factorization results in the least possible fill-in in A_{22}. The output of this procedure is a column permutation vector. We use this column permutation vector to select the initial basis for the simplex algorithm. The initial basis will be formed by the s singleton columns ($0 \leq s \leq n$, if $s > m$ we select the first m singletons as the initial basis) and the first $m - s$ columns from the column permutation vector.

The column preordering is based solely on the nonzero pattern of A_{22}. Some methods order matrix A without forming $A^T A$, while others form the explicit pattern of $A^T A$. The nonzero pattern of the symmetric $n_2 \times n_2$ matrix $A_{22}^T A_{22}$ (where n_2 is the number of columns of matrix A_{22}, $n_2 \leq n$) can be represented by a graph $G^0 = \left(V^0, E^0\right)$, where $V^0 = \{1, \cdots, n_2\}$ are the nodes and E^0 are the edges of the graph. An edge $(i, j) \in E^0$ if and only if $a_{ij} \neq 0$ and $i \neq j$. Since the matrix is symmetric, G^0 is undirected. Figure 1 illustrates an example matrix and its elimination graph G^0.

1	x	x					
x	2		x			x	x
x		3		x	x	x	
	x		4		x		x
		x		5		x	
		x	x		6		
	x	x		x		7	x
	x		x			x	8

(a) Example matrix



(b) Elimination graph

Fig. 1: Example matrix and its elimination graph

If A_{22} contains a dense (or nearly dense) row or column, the Markowitz criterion will not chose this row or column until the final stages of the elimination, thus limiting fill-in, which is consistent with our intent to produce a sparse starting basis.

As already mentioned, because the problem of obtaining an ordering with minimum fill-in is NP-complete, heuristics are applied for choosing the pivot columns in LU factorization. In each factorization step, COLMMD [21] selects as pivot the column that minimizes a loose upper bound on the external row degree. AMD [2] is based on a bound on the external row degree that is tighter than the COLMMD bound. The Markowitz rule [33] selects as pivot the element a_{ij} that minimizes the product of the degrees of row i and column j. COLAMD [13] uses an initial COLMMD metric and an AMD metric during the elimination phase. METIS [30] finds a fill-reducing ordering for a symmetric sparse matrix via recursive nested dissection. Amestoy et al. [2] performed a computational study in the context of minimum degree orderings for sparse Cholesky factorization and found that AMD is superior to the COLMMD approximation. In addition, Davis et al. [13] compared the performance of COLAMD, COLMMD, and AMD. Computational results showed that, for square nonsymmetric matrices, COLAMD is much faster and provides better orderings than COLMMD. For rectangular matrices, COLAMD is faster than COLMMD and AMD and finds orderings of comparable quality. Hence, we selected COLAMD, AMD, and METIS to create variants of our method. COLAMD orders matrix A without forming A^TA, while AMD and METIS need to form the explicit pattern of A^TA. The asymptotic run times of these ordering methods have no tight known bounds in terms of quantities that can be readily calculated beforehand [11]. However, experimental results presented in [12] showed that, in most cases, COLAMD and AMD take time roughly proportional to the number of nonzeros in A and A^TA, respectively.

All algorithms select the same singleton columns to include in the initial basis. Their only difference is the ordering method. Therefore, the three variants of the proposed method are:

– Algorithm 1 applies COLAMD.

– Algorithm 2 applies AMD.
– Algorithm 3 applies METIS.

We also experimented with using the Markowitz [33] criterion to select the basis but this approach leads to more simplex iterations. These results are consistent with the results of Davis et al. [12], who also considered the Markowitz criterion prior to the LU factorization in order to permute a matrix and reduce the worst-case fill-in. They report that the Markowitz criterion gave much worse orderings than COLAMD. In our case, these worse orderings resulted in more simplex iterations.

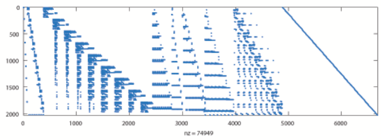
The input to all three algorithms is the constraint matrix A and the output is the basic list B. The basic steps of the aforementioned algorithms can be described as follows:

Step 1. Set $C = \emptyset$, $R = \emptyset$ and $Q = \emptyset$.
Step 2. Find the singletons in the constraint matrix A. A singleton is a column j with a single nonzero a_{ij} whose magnitude is larger than a given threshold τ. We set $\tau = 20\,(m+n)\,\epsilon \max_j \|A_{*j}\|_2$, where ϵ is the machine roundoff and $\max_j \|A_{*j}\|_2$ is the largest 2-norm of any column of A. If a singleton a_{ij} exists and $i \notin R$, add column j to the set C and row i to the set R. If $|C| = m$, go to Step 4; else, repeat this step until there are no more singletons.
Step 3. Apply COLAMD (for Algorithm 1), AMD (for Algorithm 2), or METIS (for Algorithm 3) to submatrix A_{22} (A_{22} is a submatrix of A by deleting rows in A that exist in set C and columns that exist in set R). The resulting column permutation vector is stored in set Q.
Step 4. The initial basic list is B formulated from the variables in set C and the first $m - |C|$ variables in set Q.

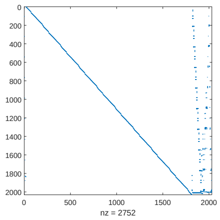
Note that we can create additional variants for each of the proposed methods if we permute the rows in R and columns in C of the constraint matrix A to the top left corner. A preliminary computational study revealed that these permutations result in more iterations and slower execution times of the simplex algorithm. Hence, these variants will not be discussed further.

The proposed algorithms do not guarantee that the initial matrix will be nonsingular since the ordering methods that are used (AMD, COLAMD, METIS) do not choose the orderings in a way that the generated matrices will be nonsingular. In fact, we were able to generate some trivial instances for which the ordering methods generate an ordering that does make our algorithm produce a singular initial matrix. However, the proposed method did not generate a singular initial matrix for any of the benchmark problems we experimented with (from NETLIB, Kennington, Mészáros, Mittelmann benchmark libraries).

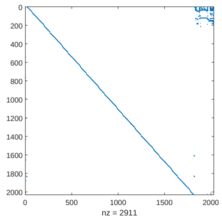
Figures 2 and 3 present the sparsity pattern of the constraint matrix A and the initial basis using Algorithms 1 to 3 for problems pilot87 and qap15 from NETLIB, respectively. All algorithms form nearly-triangular initial bases.



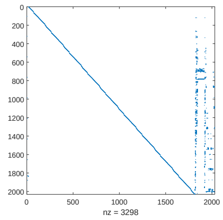
(a) Constraint matrix A



(b) Initial basis from Algorithm 1

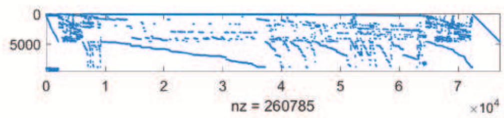


(c) Initial basis from Algorithm 2

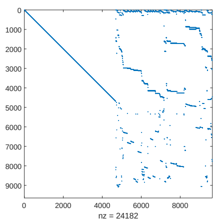


(d) Initial basis for Algorithm 3

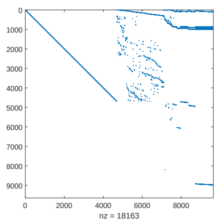
Fig. 2: Sparsity pattern of the constraint matrix A and the initial basis using Algorithms 1 to 3 for problem pilot87 of the NETLIB set



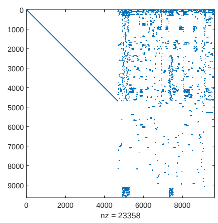
(a) Constraint matrix A



(b) Initial basis from Algorithm 1



(c) Initial basis from Algorithm 2



(d) Initial basis for Algorithm 3

Fig. 3: Sparsity pattern of the constraint matrix A and the initial basis using Algorithms 1 to 3 for problem qap15 of the NETLIB set

4 Computational study

The aim of this computational study is to investigate the performance of the simplex algorithm in conjunction with the proposed crash procedures. We give the initial bases generated by all three algorithms as input to the CPLEX solver and compare their performance against the CPLEX default crash procedure. We do this using both the primal and the dual simplex algorithm.

All computations were performed on an Intel Xeon CPU E5-2660 v3 with 128 GB of main memory, a clock of 2.6 GHz, an L1 code cache of 32 KB per core, an L1 data cache of 32 KB per core, an L2 cache of 256 KB per core, and an L3 cache of 24 MB, running under Centos 7 64-bit. We considered a set of 150 medium-sized and large benchmark problems (NETLIB, Kennington, Mészáros, Mittelmann) in preliminary runs. Then, we eliminated the trivial problems, i.e., instances solved in less than one second with all the algorithms considered in this paper when CPLEX presolve is disabled ("preprocessing.presolve" option is set to 0). The final set of instances that we used in this computational study includes 95 benchmark problems. On average, 6% of the variables in the constraint matrix are singletons while 10% of the variables in the initial basis are singletons. Table S1 in the Online Supplement presents the number of constraints, variables, and nonzeros for each of the benchmark problems. We used CPLEX to presolve all instances and exported the MPS files. We then generated the initial bases for each presolved problem using the three algorithms and stored them in BAS files (MPS basis files, known as BAS files, that contain the information needed to define an initial basis). We gave the generated BAS files as input to CPLEX and compared the performance of the solver against that of the CPLEX default crash procedures. We did this comparison for both the primal and the dual simplex algorithm. We used default values for all algorithmic options of CPLEX. An execution time limit of $15,000$ seconds was imposed on all runs.

In the tables and figures below, the following abbreviations are used: (i) Time: CPU time to solve a problem with CPLEX, and (ii) Tit: total iterations. The time to construct an initial basis with the proposed algorithms is negligible in comparison to the total time needed to solve the instances. Algorithm 1 (based on COLAMD) is faster than Algorithms 2 (based on AMD) and 3 (based on METIS).

Table 1 presents the average value (shifted geometric mean over the entire collection of test problems) of Time and Tit with four different initialization algorithms followed by the application of the primal CPLEX routine to the presolved problems. For the nonnegative numbers $a_1, \cdots, a_k \in \mathbb{R}_+$ and a shift $s \in \mathbb{R}_+$, the average is defined by

$$\gamma_s\left(a_1, \cdots, a_k\right) = \left(\prod_{i=1}^{k} \left(a_i + s\right)\right)^{\frac{1}{k}} - s$$

We use a shift of 10 for the execution time and $1,000$ for the number of iterations in order to decrease the influence of the easy instances in the mean values.

Tables S2–S5 in the Online Supplement present the detailed results for each problem and algorithm combination. As seen in Tables 1 and S2–S5, Algorithm 3, based on METIS, performs better than all the other proposed methods on average. All the proposed methods require less CPU time and fewer iterations than the default CPLEX crash procedure. Primal CPLEX using Algorithm 3 results in 5% reduction of the geometric mean of the execution time of CPLEX's primal simplex algorithm. Moreover, the proposed methods are significantly faster on instances for which the CPLEX default requires over $1,000$ seconds (13 problems). For these problems, primal CPLEX using Algorithm 3 is 37% faster than primal CPLEX using its default crash procedure.

Figures 4 and 5 present performance profiles [14] based on the execution time and the number of iterations, respectively, of the primal simplex algorithm using the three proposed algorithms and the default crash procedure. Performance profiles are displayed in logarithmic scale with base 2. Algorithm 3, based on METIS, performs better than the other proposed methods and the default crash procedure. In particular, Algorithm 3 is better than the other methods in the interval $[1.1, 7]$. Moreover, Algorithm 3 is faster than the CPLEX default crash procedure on 64 out of 95 problems and appears dominant in the performance profile. Algorithm 3 performs 4% fewer Phase I iterations, 2% fewer Phase II iterations, and 6% fewer total iterations than the CPLEX crash procedure. The proposed algorithm performs fewer Phase I iterations on 51 instances, fewer Phase II iterations on 48 instances, and fewer total iterations on 47 instances. Algorithm 3 finds a better starting solution (closer either to feasibility or optimality) than the CPLEX crash procedure on 61 problems. Additionally, Algorithm 3 finds the optimal solution on one problem, a basic feasible solution on six problems and a nearly basic feasible solution (the percentage of Phase I iterations to total iterations is less than 10%) on 18 problems, while the CPLEX crash procedure finds a basic feasible solution on one problem and a nearly basic feasible solution on 27 problems. In addition, Algorithm 3 constructs an initial basis that is, on average, four times sparser than that of the CPLEX crash procedure.

Although the performance of Algorithm 3 is consistent on both easy and hard instances, it results in significant reductions when solving hard instances. The performance of CPLEX with its default crash procedure deteriorates for large and hard problems. More specifically, there are some problems, e.g., neos2, ns1687037, nug08-3rd, and nug20, where CPLEX experiences numerical difficulties as it approaches the optimal solution. These difficulties caused CPLEX to change the value of the Markowitz tolerance and resort to new Phase I iterations in order to restore feasibility. CPLEX may start again from an infeasible solution more than once during the solution of a problem, e.g., four and six times for the ns1687037 and nug20 instances, respectively. CPLEX also experienced numerical issues when starting from a solution generated by one of the proposed algorithms. In all such cases, however, CPLEX needed only a few iterations to restore a feasible solution. Therefore, the proposed methods seem to have the ability to avoid numerical issues encountered by the starting points obtained through the current default initialization algorithms in CPLEX.

Table 2 presents a summary of the results for the dual simplex algorithm. Detailed results with all problem and algorithm combinations are provided in

Table 1: Shifted geometric times and iterations for the primal simplex algorithm using shifted geometric mean

Algorithm	Test set	Time	Tit
CPLEX using Algorithm 1	All problems	56	41,921
	> 1,000 sec	3,334	308,677
CPLEX using Algorithm 2	All problems	58	41,639
	> 1,000 sec	3,626	348,100
CPLEX using Algorithm 3	All problems	55	40,485
	> 1,000 sec	2,885	300,005
CPLEX using default crash procedure	All problems	58	43,156
	> 1,000 sec	4,606	348,349

95 problems in total and 13 hard problems (problems for which CPLEX using the default crash procedure needs more than $1,000$ seconds to solve)

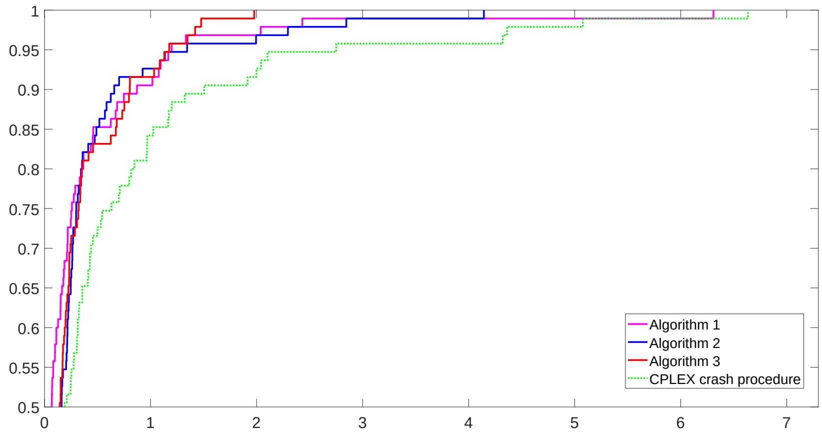


Fig. 4: Performance profiles comparing the three algorithms and default crash procedure based on the execution time for the primal simplex

Tables S6–S9 in the Online Supplement. In this case, CPLEX's dual simplex algorithm using the default crash procedure is 5% faster than CPLEX's dual simplex algorithm using Algorithm 3. However, Algorithm 3 is significantly better on instances for which the CPLEX default requires over $1,000$ seconds (8 problems). For these instances, dual CPLEX using Algorithm 3 is 10% faster than dual CPLEX using its default crash procedure. In addition, Algorithm 3 is performing better than Algorithms 1 and 2.

Figures 6 and 7 present performance profiles based on the execution time and the number of iterations, respectively, of the dual simplex algorithm using the three proposed algorithms and the default crash procedure. CPLEX default

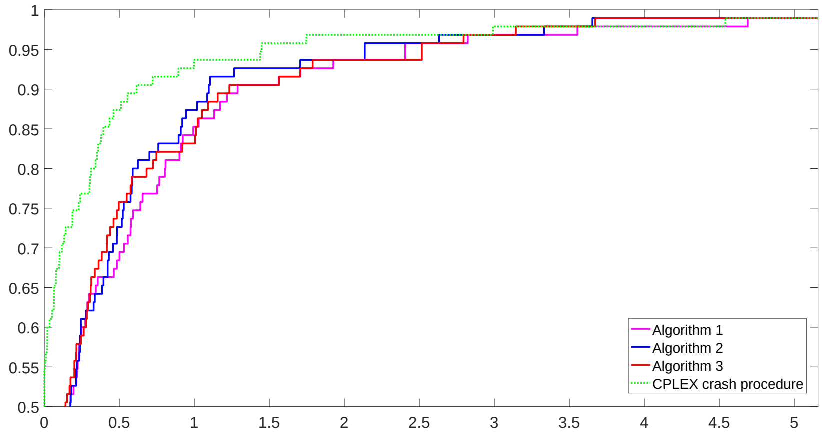


Fig. 5: Performance profiles comparing the three algorithms and default crash procedure based on the number of iterations for the primal simplex algorithm

crash procedure has the highest probability of being the fastest solver for values of τ in the interval $[0.5, 7]$. CPLEX using its default crash procedure is 5% faster than CPLEX using Algorithm 3. The reduction to the execution time that the proposed algorithms offer is more pronounced on hard instances. CPLEX using Algorithm 3 performs 10% faster than CPLEX using its default crash procedure.

Using dual simplex, the CPLEX crash procedure is faster than Algorithm 3 on 52 out of 95 instances. CPLEX crash procedure performs 52% more Phase I iterations, 1% more Phase II iterations, and 11% more total iterations in comparison to Algorithm 3. The CPLEX crash procedure also finds a feasible solution on the majority of the instances. Algorithm 3 performs fewer Phase I iterations on 28 instances, fewer Phase II iterations on 39 instances, and fewer total iterations on 33 instances. Taking into account only the hard instances, Algorithm 3 results in great reductions compared to the CPLEX dual simplex algorithm. Similar to the primal simplex algorithm, the performance of the CPLEX dual simplex algorithm with the CPLEX default crash procedure deteriorates for large and hard problems. It is worth mentioning here that CPLEX's barrier solver can solve the instance nug20 in a few minutes.

Table 3 presents the average performance of the primal simplex algorithm using Algorithm 3 compared to the performance of the dual simplex algorithm using CPLEX's default crash procedure in terms of execution time, number of iterations, and density of the generated basis. CPLEX's primal simplex algorithm initialized with Algorithm 3 is 7% faster than the default CPLEX algorithm. This is correlated with the observed 73% reduction in the density of the generated initial bases. For the instances for which the dual simplex algo-

Table 2: Shifted geometric means of times and iterations for the dual simplex algorithm

Algorithm	Test set	Time	Tit
CPLEX using Algorithm 1	All problems	51	27,854
	> 1,000 sec	8,674	333,956
CPLEX using Algorithm 2	All problems	50	26,901
	> 1,000 sec	8,782	327,671
CPLEX using Algorithm 3	All problems	50	27,281
	> 1,000 sec	7,074	306,132
CPLEX using default crash procedure	All problems	48	24,345
	> 1,000 sec	7,844	291,366

95 problems in total and 8 hard problems (problems for which CPLEX using the default crash procedure needs more than $1,000$ seconds to solve)

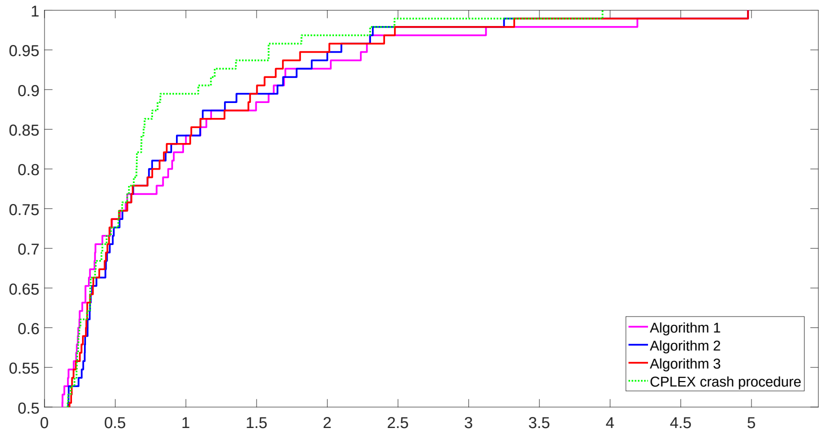


Fig. 6: Performance profiles comparing the three algorithms and default crash procedure based on the execution time for the dual simplex algorithm

rithm with CPLEX's default crash procedure needs more than $1,000$ seconds to solve, CPLEX's primal simplex algorithm using Algorithm 3 is 25% faster than the default CPLEX algorithm. Even though CPLEX with Algorithm 3 performs more iterations than the default CPLEX algorithm, Algorithm 3 spends significantly less time per iteration than CPLEX with the default crash procedure, for both primal and dual simplex.

The above computational results suggest there are many problems for which the proposed algorithms outperform the CPLEX default initialization scheme, while the latter is still useful, especially for easier problems. This observation suggests an opportunity to combine all these algorithms in a speculative parallelization approach on computing equipment with a small number of cores. CPLEX has no parallel simplex facility. Hence, we will compute

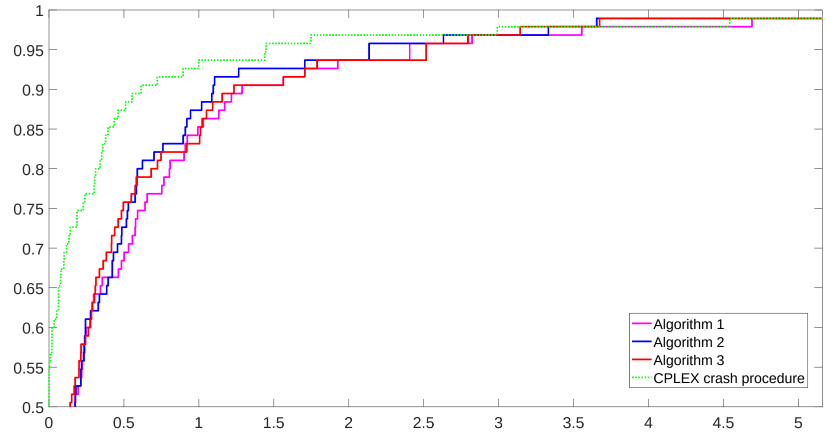


Fig. 7: Performance profiles comparing the three algorithms and default crash procedure based on the number of iterations for the dual simplex algorithm

Table 3: Average performance of the best proposed method and the best CPLEX crash procedure

Algorithm	Test set	Time	Tit	Density
Primal CPLEX using Algorithm 3	All problems	838	148,835	0.11%
	$>$ 1,000 sec	7,378	597,926	0.01%
Dual CPLEX using default crash procedure	All problems	903	90.884	0.50%
	$>$ 1,000 sec	9,825	442,275	0.01%
Speedup of the best proposed method over the best CPLEX crash procedure	All problems	7%	-39%	73%
	$>$ 1,000 sec	25%	-26%	0%

95 problems in total and 8 hard problems (problems for which CPLEX with its default crash procedure needs more than $1,000$ seconds to solve)

speedups due to the utilization of multiple cores to run different variants of the proposed methods versus running the dual CPLEX algorithm on a single core. Table 4 presents the shifted geometric means of the execution times when using multiple cores, each core running CPLEX with a different variant of the crash procedure in a task-dependent fashion. The default dual CPLEX using the default CPLEX crash procedure (running on one core) needs 48 seconds on average to solve the problems in our testset. Running the primal and dual CPLEX using Algorithm 3 on two cores and taking the best performance of each variant results in a mean speedup of 1.2 over CPLEX's dual simplex algorithm. The execution of the primal and dual CPLEX using the default crash procedure and Algorithm 3 (running on four cores) results in a mean speedup of 1.3 over CPLEX's dual simplex algorithm. These speedups are comparable to those of state-of-the-art parallel simplex solvers for a similar number of cores [27].

Table 4: Shifted geometric means of wall-clock times from runs on multiple cores

Algorithm	Time
Dual CPLEX using default crash procedure	48
Best of primal and dual CPLEX using Algorithm 3	40
Best of primal and dual CPLEX using default crash procedure or Algorithm 3	37

5 Conclusions

We presented three algorithms that construct a nearly-triangular and sparse initial basis for the simplex algorithm. The initial basis is artificial-free and includes as many structural variables as possible. The aim of the proposed methods is to reduce the subsequent fill-ins of the LU factorization, the number of iterations, and the computational effort at each iteration. We experimented with various ordering methods in order to create a sparse nearly-triangular initial basis for the simplex algorithm. Using a collection of 95 benchmark LPs, we found that the best way to speed up the primal and dual simplex algorithms for CPLEX is to utilize Algorithm 3, which forms a starting basis using all available column singletons plus the columns obtained from the application of METIS to the remainder of the LP matrix.

Algorithm 3 results in 5% average reduction of the execution time of CPLEX's primal simplex algorithm. Although the proposed algorithm reduces CPLEX's execution time on the majority of instances, it is significantly faster than the CPLEX default crash procedure on hard instances. Taking into account only the hard instances (instances that CPLEX needs more than $1,000$ seconds to solve), Algorithm 3 results in 37% average reduction of the execution time of CPLEX's primal simplex algorithm. CPLEX's dual simplex algorithm using its default crash procedure is 5% faster than CPLEX's dual simplex algorithm using Algorithm 3. CPLEX using Algorithm 3 is 10% faster than CPLEX using its default crash procedure on instances for which CPLEX needs more than $1,000$ seconds to solve.

Finally, the proposed algorithms lend themselves to speculative parallelization of the simplex algorithm. With respect to the dual CPLEX with default initialization, the proposed algorithms lead to speedups of 1.2 and 1.3 on two and four cores, respectively.

References

1. Al-Najjar, C., Malakooti, B.: Hybrid-LP: Finding advanced starting points for simplex, and pivoting LP methods. Computers & Operations Research **38**, 427–434 (2011)
2. Amestoy, P.R., Davis, T.A., Duff, I.S.: An approximate minimum degree ordering algorithm. SIAM Journal on Matrix Analysis and Applications **17**, 886–905 (1996)

3. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Mathematical Programming **71**, 221–245 (1995)
4. Bertsimas, D., Tsitsiklis, J.: Introduction to Linear Optimization. Athena Scientific, Boston, MA (1997)
5. Bixby, R.E.: Implementing the simplex method: The initial basis. ORSA Journal on Computing **4**, 267–284 (1992)
6. Carstens, D.M.: Crashing techniques. *In W. Orchard-Hays (ed.)*, Advanced Linear-Programming Computing Techniques, *McGraw-Hill, New York, NY* pp. 131–139 (1968)
7. Chvátal, V.: Linear Programming. W. H. Freeman, New York (1983)
8. Curtis, A.R., Reid, J.K.: On the automatic scaling of matrices for Gaussian elimination. Journal of the Institute of Mathematics and Its Applications **10**, 118–124 (1972)
9. Dantzig, G.B.: Programming in a linear structure. Econometrica **17**, 73–74 (1949)
10. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press, Princeton, NJ (1963)
11. Davis, T.A.: Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. ACM Transactions on Mathematical Software **38**, 8–29 (2011)
12. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: A column approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software **30**, 353–376 (2004)
13. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: Algorithm 836: COLAMD, a column approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software **30**, 377–380 (2004)
14. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)
15. Elble, J.M., Sahinidis, N.V.: A review of LU factorization in the simplex algorithm. International Journal of Mathematics in Operational Research **4**, 319–365 (2012)
16. Elble, J.M., Sahinidis, N.V.: A review of the LU update in the simplex algorithm. International Journal of Mathematics in Operational Research **4**, 366–399 (2012)
17. Elble, J.M., Sahinidis, N.V.: Scaling linear optimization problems prior to application of the simplex method. Computational Optimization and Applications **52**, 345–371 (2012)
18. Erisman, A.M., Grimes, R.G., Lewis, J.G., Poole, Jr., W.G.: A structurally stable modification of Hellerman-Rarick's P^4 algorithm for reordering unsymmetric sparse matrices. SIAM Journal on Numerical Analysis **22**, 369–385 (1985)
19. Forrest, J.J., Goldfarb, D.: Steepest-edge simplex algorithms for linear programming. Mathematical Programming **57**, 341–374 (1992)
20. Forrest, J.J.H., Tomlin, J.A.: Updated triangular factors of the basis to maintain sparsity in the product form simplex method. Mathematical Programming **2**, 263–278 (1972)
21. Gilbert, J.R., Moler, C.B., Schreiber, R.: Sparse matrices in MATLAB: Design and implementation. SIAM Journal on Matrix Analysis and Applications **13**, 333–356 (1992)
22. Goldfarb, D.: On the Bartels-Golub decomposition for linear programming bases. Mathematical Programming **13**, 272–279 (1977)
23. Gould, N.I.M., Reid, J.K.: New crash procedures for large systems of linear constraints. Mathematical Programming **45**, 475–501 (1989)
24. Gould, N.I.M., Toint, P.L.: Preprocessing for quadratic programming. Mathematical Programming **100**, 95–132 (2004)
25. Gülpinar, N., Mitra, G., Maros, I.: Creating advanced bases for large scale linear programs exploiting embedded network structure. Computational Optimization and Applications **21**, 71–93 (2002)
26. Harris, P.M.J.: Pivot selection methods of the Devex LP code. Mathematical Programming **5**, 1–28 (1973)
27. Huangfu, Q., Hall, J.: Parallelizing the dual revised simplex method. Mathematical Programming Computation **10**, 119–142 (2018)
28. Junior, H.V., Lins, M.P.E.: An improved initial basis for the simplex algorithm. Computers & Operations Research **32**, 1983–1993 (2005)
29. Kaczmarz, S.: Angenäherte auflösung von systemen linearer gleichungen. Bulletin International de l'Academie Polonaise des Sciences et des Lettres **35**, 355–357 (1937)
30. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20**, 359–392 (1998)

31. Lenstra, J.K., Rinnoy Kan, A.H.G., Schrijver, A. (eds.): History of Mathematical Programming. CWI North Holland, Amsterdam (1991)
32. Luh, H., Tsaih, R.: An efficient search direction for linear programming problems. Computers & Operations Research **29**, 195–203 (2002)
33. Markowitz, H.M.: The elimination form of the inverse and its application to linear programming. Management Science **3**, 255–269 (1957). (Originally at The RAND Corporation, Research Memorandum RM-1452, 1955)
34. Maros, I.: Computational Techniques of the Simplex Method. Kluwer Academic Publishers, Boston, MA (2003)
35. Maros, I., Mitra, G.: Strategies for creating advanced bases for large-scale linear programming problems. INFORMS Journal on Computing **10**, 248–260 (1998)
36. Mészáros, C., Suhl, U.H.: Advanced preprocessing techniques for linear and quadratic programming. OR Spectrum **25**, 575–595 (2003)
37. Murtagh, B.A., Saunders, M.A.: MINOS 5.1 User's Guide. Tech. rep., Department of Operations Research, Stanford University, Stanford, CA (1987)
38. Nabli, H.: An overview on the simplex algorithm. Applied Mathematics and Computation **210**, 479–489 (2009)
39. Nabli, H., Chahdoura, S.: Algebraic simplex initialization combined with the nonfeasible basis methods. European Journal of Operational Research **245**, 384—-391 (2015)
40. Pan, P.Q.: Linear programming computation. Springer-Verlag, Berlin (2014)
41. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications (1998)
42. Ploskas, N., Samaras, N.: GPU accelerated pivoting rules for the simplex algorithm. Journal of Systems and Software **96**, 1–9 (2014)
43. Ploskas, N., Samaras, N.: A computational comparison of scaling techniques for linear optimization problems on a graphical processing unit. International Journal of Computer Mathematics **92**, 319–336 (2015)
44. Terlaky, T., Zhang, S.: Pivot rules for linear programming: A survey on recent theoretical developments. Annals of Operations Research **46**, 203–233 (1993)
45. Tomlin, J.A.: An accuracy test for updating triangular factors. Mathematical Programming Study **4**, 142–145 (1975)
46. Yannakakis, M.: Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic and Discrete Methods **2**, 77–79 (1981)
47. Ye, Y.: Eliminating columns in the simplex method for linear-programming. Journal Of Optimization Theory and Applications **63**, 69–77 (1989)

ONLINE SUPPLEMENT: Detailed computational results

In the tables below, the following abbreviations are used:
- Time: CPU time to solve a problem with CPLEX,
- PhIit: Phase I iterations,
- PhIIit: Phase II iterations,
- Tit: total iterations,
- Infeas: infeasibility or scaled infeasibility (if the problem has been scaled) of the initial solution (if the starting basis is not feasible),
- Feas: absolute difference between the initial solution objective value and the optimal (if the starting basis is feasible), and
- Den: density of the initial basis.

Table S1: Statistics of the benchmark problems

Problem	**Constraints**	**Variables**	**Nonzeros**
aa01	823	8,904	72,965
aa03	825	8,627	70,806
aa3	825	8,627	70,806
aa5	801	8,308	65,953
aa6	646	7,292	51,728
brazil3	14,646	23,968	133,184
buildingenergy	277,594	154,978	788,969
car4	16,384	33,052	63,724
chromaticindex1024-7	67,583	73,728	270,324
co5	5,774	7,993	53,661
co9	10,789	14,851	101,578
cont1	160,792	40,398	399,990
cont11	160,792	80,396	399,990
cq5	5,048	7,530	47,353
cq9	9,278	13,778	88,897
cre-b	9,648	72,447	256,095
cre-d	8,926	69,980	242,646
d2q06c	2,171	5,167	32,417
dano3mip	3,202	13,873	79,655
dbic1	43,200	183,235	1,038,761
dbir2	18,906	27,355	1,139,637
dfl001	6,071	12,230	35,632
ds-big	1,042	174,997	4,623,442
e18	24,617	14,231	132,095
ex10	69,608	17,680	1,162,000
ex3sta1	17,443	8,156	59,419
fit2p	3,000	13,525	50,284
fome13	48,568	97,840	48,568
fxm3_16	41,340	64,162	370,839
fxm4_6	22,400	30,732	248,989

ge	10,099	11,098	39,554
gen1	1,537	2,560	63,085
gen2	1,121	3,264	81,855
gen4	1,537	4,297	107,102
gen	769	2,569	63,085
gosh	3,792	10,733	97,231
irish-electricity	104,259	61,728	523,257
ken-13	28,632	42,659	97,246
ken-18	105,127	154,699	358,171
L1_sixm250obs	986,069	428,032	4,280,320
Linf_520c	93,326	69,004	566,193
l30	2,701	15,380	51,169
lpl3	10,828	33,538	100,377
maros-r7	3,136	9,408	144,848
model10	4,400	15,447	149,000
nemspmm1	2,372	8,622	55,586
nemswrld	7,138	27,174	190,907
neos1	131,581	1,892	468,009
neos2	132,568	134,128	685,087
neos3	512,209	6,624	1,542,816
neos	479,119	36,786	1,047,675
neos-5052403-cygnet	38,268	32,868	4,898,304
nl	7,039	9,718	41,428
ns1644855	40,698	30,200	2,110,696
ns1687037	50,622	43,749	1,406,739
ns1688926	32,768	16,857	1,712,128
nsct2	23,003	14,981	675,156
nsir2	4,453	5,717	150,599
nug08-3rd	19,728	20,448	139,008
nug12	3,192	8,856	38,304
nug15	6,330	22,275	94,950
nug20	15,240	72,600	304,800
osa-30	4,350	100,024	600,138
osa-60	10,280	232,966	1,397,793
p010	10,090	19,000	117,910
pds-100	156,243	505,360	1,086,785
pds-20	33,874	105,728	230,200
pds-40	66,844	212,859	462,128
physiciansched3-3	266,227	79,555	1,062,479
pilot87	2,030	4,882	73,152
pilot	1,441	3,652	43,167
pltexpa3_16	28,350	74,172	150,801
qap12	3,192	8,856	38,304
qap15	6,330	22,275	94,950
rail02	95,791	270,869	756,228
rail4284	4,284	1,092,610	11,279,748

rat7a	3,136	9,408	268,908
s100	14,733	364,417	1,777,917
s250r10	10,962	273,142	1,318,607
savsched1	295,989	328,575	1,770,507
sc205-2r-1600	35,213	35,214	96,030
scfxm1-2r-256	37,980	57,714	213,159
self	960	7,364	1,148,845
seymourl	4,944	1,372	33,549
sgpf5y6	246,077	308,634	828,070
shs1023	133,944	444,625	1,044,725
square41	40,160	62,234	13,566,426
stat96v1	5,995	197,472	5,995
stat96v4	3,173	62,212	490,472
stocfor3	16,675	15,695	64,875
stormG2_1000	528,185	1,259,121	3,341,696
stp3d	159,488	204,880	662,128
supportcase10	165,684	14,770	555,082
truss	1,000	8,806	27,836
watson_2	352013	671861	1841028

Table S2: Results using the primal CPLEX after initialization with Algorithm 1

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**	**Den**
aa01	1.98	7,851	10,742	18,593	6.31E+02	-	0.8966%
aa03	2.31	11,101	11,622	22,723	5.18E+02	-	0.8165%
aa3	2.01	5,366	17,935	23,301	4.47E+02	-	0.8165%
aa5	1.91	5,835	18,320	24,155	6.93E+02	-	0.8255%
aa6	1.31	3,723	15,289	19,012	4.44E+02	-	1.0577%
brazil3	27.06	45,134	17,092	62,226	5.71E+03	-	0.0347%
buildingen	244.46	53,800	184,658	238,458	4.81E+05	-	0.0005%
car4	0.27	0	4,061	4,061	-	5.48E+03	0.1684%
chrom1024-7	9.38	65,293	83	65,376	2.30E+04	-	0.0015%
co5	1.33	2,247	6,261	8,508	9.74E+03	-	0.0325%
co9	5.36	4,313	15,912	20,225	2.11E+04	-	0.0198%
cont1	734.33	43,262	827	44,089	1.45E+03	-	0.0074%
cont11	4,324.36	100,730	33,163	133,893	6.01E+02	-	0.0037%
cq5	0.96	2,690	5,982	8,672	1.13E+04	-	0.0310%
cq9	3.65	4,793	16,923	21,716	2.26E+04	-	0.0203%
cre-b	1.06	2,020	22,493	24,513	1.54E+04	-	0.0248%
cre-d	0.71	1,872	16,318	18,190	1.64E+04	-	0.0339%
d2q06c	1.10	1,475	5,295	6,770	3.39E+04	-	0.1191%
dano3mip	6.11	10,776	25,042	35,818	2.81E+04	-	0.1187%

dbic1	16.20	35	105,174	105,209	1.12E+02	-	0.0030%
dbir2	0.47	614	2,805	3,419	2.09E+02	-	0.0138%
dfl001	6.10	10,028	18,311	28,339	6.64E+03	-	0.0703%
ds-big	405.15	0	802,134	802,134	-	4.52E+04	0.4628%
e18	4.87	3,029	26,497	29,526	2.46E+02	-	0.0043%
ex10	254.21	76,366	0	76,366	6.25E+02	-	0.0016%
ex3sta1	18.23	12,446	200	12,646	7.96E+03	-	0.0347%
fit2p	1.21	295	11,582	11,877	8.16E+02	-	0.0333%
fome13	168.10	80,963	151,675	232,638	5.28E+04	-	0.0088%
fxm3_16	1.63	5,287	20,707	25,994	1.42E+06	-	0.0169%
fxm4_6	0.81	6,068	14,337	20,405	6.84E+05	-	0.0178%
ge	0.91	3,280	5,268	8,548	6.42E+05	-	0.0294%
gen1	1.38	3,654	465	4,119	8.55E+03	-	0.3857%
gen2	14.59	15,335	0	15,335	1.46E+04	-	0.1795%
gen4	15.07	12,214	2,340	14,554	1.39E+04	-	0.1355%
gen	1.29	3,654	465	4,119	8.55E+03	-	0.3857%
gosh	0.09	2,790	0	2,790	8.18E+02	-	0.1063%
irish-e	190.52	43,804	224,840	268,644	2.53E+03	-	0.0259%
ken-13	1.28	5,408	23,775	29,183	1.89E+05	-	0.0210%
ken-18	9.25	14,673	93,603	108,276	1.38E+06	-	0.0058%
L1_sixm	15,000.00	184,025	93,772	277,797	2.23E+09	-	0.0014%
Linf_520c	4,513.74	293,020	23,612	316,632	2.00E+08	-	0.0054%
l30	14.59	2,237	24,254	26,491	1.00E+00	-	0.0660%
lpl3	0.22	41	7,421	7,462	1.97E+03	-	0.0380%
maros-r7	1.09	266	4,570	4,836	9.31E+06	-	0.0465%
model10	3.36	9,964	15,362	25,326	3.48E+03	-	0.1939%
nemspmm1	1.14	3,658	4,377	8,035	3.55E+04	-	0.1795%
nemswrld	11.48	18,046	27,031	45,077	3.39E+02	-	0.1039%
neos1	152.79	1	75,313	75,314	0.00E+00	-	0.0008%
neos2	677.34	1	174,401	174,402	0.00E+00	-	0.0008%
neos3	15,000.00	873	293,989	294,862	7.90E+08	-	0.0002%
neos	305.50	220,969	11,921	232,890	1.53E+07	-	0.0002%
neos5052403	155.73	14,326	211,506	225,832	1.90E+04	-	0.0026%
nl	2.65	7,509	10,426	17,935	2.34E+04	-	0.0248%
ns1644855	632.58	414,675	48,742	463,417	3.54E+07	-	0.0045%
ns1687037	4,501.54	101,222	74,409	175,631	2.62E+05	-	0.0024%
ns1688926	194.11	1,033	83,459	84,492	4.52E+13	-	0.0068%
nsct2	0.30	808	1,646	2,454	8.11E+02	-	0.0128%
nsir2	0.18	1,541	1,322	2,863	6.16E+02	-	0.0434%
nug08-3rd	11,527.34	18,178	376,758	394,936	1.30E+01	-	0.0180%
nug12	22.35	7,334	20,640	27,974	2.00E+00	-	0.1000%
nug15	233.53	13,453	67,231	80,684	1.00E+00	-	0.0516%
nug20	14,430.12	75,479	828,646	904,125	4.70E+01	-	0.0207%
osa-30	0.18	1,595	2,797	4,392	3.11E+03	-	0.0234%
osa-60	0.71	2,373	6,874	9,247	5.65E+03	-	0.0098%
p010	0.42	2,364	559	2,923	4.24E+04	-	0.0406%

pds-100	300.26	30,196	818,528	848,724	6.90E+07	-	0.0025%
pds-20	3.72	0	85,208	85,208	-	5.49E+09	0.0198%
pds-40	31.14	0	389,835	389,835	-	1.05E+10	0.0082%
psched3-3	128.22	94,521	51,496	146,017	3.58E+04	-	0.0021%
pilot87	2.90	4,124	3,155	7,279	1.62E+04	-	0.3400%
pilot	1.28	3,758	1,661	5,419	2.43E+04	-	0.2500%
pltexpa3_16	0.04	0	879	879	-	1.42E+01	0.0097%
qap12	21.00	6,031	20,302	26,333	1.00E+00	-	0.1000%
qap15	247.25	15,623	69,270	84,893	1.00E+00	-	0.0519%
rail02	2,565.18	64,800	308,688	373,488	2.29E+02	-	0.0045%
rail4284	1,877.47	453,733	2,483,833	2,937,566	3.94E+03	-	0.0353%
rat7a	5.36	244	6,417	6,661	9.32E+06	-	0.0465%
s100	28.00	26,891	306,381	333,272	1.42E+02	-	0.0069%
s250r10	30.94	5,921	278,525	284,446	8.00E+00	-	0.0131%
savsched1	99.56	4,181	181,457	185,638	6.67E+04	-	0.0003%
sc205-2r-1600	0.02	130	14	144	5.40E+04	-	0.0184%
scfxm1-2r-256	4.71	11,799	11,650	23,449	4.82E+05	-	0.0079%
self	0.01	0	0	0	-	0.00E+00	0.1042%
seymourl	1.18	3,191	3,219	6,410	4.36E+03	-	0.0215%
sgpf5y6	0.05	1,284	550	1,834	4.76E+08	-	0.0084%
shs1023	125.40	12,799	398,787	411,586	2.84E+06	-	0.0018%
square41	987.59	102,368	40,634	143,002	4.62E+03	-	0.0570%
stat96v1	209.02	6,138	772,530	778,668	2.50E-01	-	0.0402%
stat96v4	124.36	153,521	164,498	318,019	7.41E+03	-	0.1700%
stocfor3	0.48	855	6,517	7,372	9.25E+02	-	0.0197%
storm_1000	352.76	67,425	202,387	269,812	7.78E+03	-	0.0812%
stp3d	2,048.75	336,865	265,550	602,415	2.19E+03	-	0.0016%
support10	711.88	32,623	140,098	172,721	1.69E+03	-	0.0010%
truss	0.67	3,776	14,453	18,229	1.96E+03	-	0.2100%
watson_2	60.97	29,842	145,116	174,958	6.30E+05	-	0.0037%
Average	882.47	37,262	122,051	159,313	-	-	0.1011%

Table S3: Results using the primal CPLEX after initialization with Algorithm 2

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**	**Den**
aa01	2.15	6,754	11,443	18,197	8.09E+02	-	0.9359%
aa03	2.30	10,329	11,328	21,657	7.33E+02	-	0.8804%
aa3	2.08	5,053	17,374	22,427	5.18E+02	-	0.8804%
aa5	1.77	5,372	13,118	18,490	4.85E+02	-	0.7945%
aa6	1.41	3,334	15,881	19,215	4.28E+02	-	0.8938%
brazil3	29.35	47,551	17,551	65,102	1.12E+04	-	0.0347%
buildingen	110.28	59,380	122,738	182,118	4.81E+05		0.0004%

car4	0.28	0	4,061	4,061	-	5.48E+03	0.1684%
chrom1024-7	17.81	83,236	366	83,602	2.30E+04		0.0015%
co5	1.31	2,025	6,598	8,623	9.74E+03	-	0.0299%
co9	5.55	4,116	15,069	19,185	2.10E+04	-	0.0163%
cont1	764.66	43,262	827	44,089	1.45E+03		0.0037%
cont11	4,283.30	100,730	33,163	133,893	6.01E+02		0.0074%
cq5	1.14	2,690	5,982	8,672	1.13E+04	-	0.0310%
cq9	3.77	5,174	15,647	20,821	2.25E+04	-	0.0170%
cre-b	1.05	2,159	20,926	23,085	1.54E+04	-	0.0217%
cre-d	0.82	1,933	17,074	19,007	1.64E+04	-	0.0288%
d2q06c	1.15	1,686	5,275	6,961	3.41E+04	-	0.0734%
dano3mip	8.13	18,434	22,138	40,572	3.60E+04	-	0.0857%
dbic1	18.01	12	106,852	106,864	1.16E+02	-	0.0030%
dbir2	0.56	614	2,805	3,419	2.09E+02	-	0.0138%
dfl001	7.19	8,564	17,099	25,663	1.62E+04	-	0.0693%
ds-big	401.65	0	802,134	802,134	-	4.52E+04	0.4628%
e18	2.28	2,493	14,268	16,761	2.46E+02	-	0.0043%
ex10	243.32	73,052	0	73,052	5.81E+02	-	0.0016%
ex3sta1	0.23	0	99	99	-	6.31E+01	0.0347%
fit2p	1.07	295	11,582	11,877	8.16E+02	-	0.0333%
fome13	165.92	80,458	153,055	233,513	4.84E+04	-	0.0092%
fxm3_16	2.19	7,427	19,874	27,301	1.40E+06	-	0.0065%
fxm4_6	0.73	3,871	14,565	18,436	5.64E+05	-	0.0072%
ge	1.09	3,280	5,268	8,548	6.42E+05	-	0.0278%
gen1	1.33	3,712	527	4,239	8.91E+03	-	0.3857%
gen2	14.53	14,385	0	14,385	1.47E+04	-	0.1795%
gen4	17.68	12,898	2,245	15,143	1.42E+04	-	0.1355%
gen	1.51	3,712	527	4,239	8.91E+03	-	0.3857%
gosh	0.10	3,267	0	3,267	8.34E+02	-	0.0626%
irish-e	187.05	43,804	224,840	268,644	2.53E+03	-	0.0259%
ken-13	1.21	5,282	24,014	29,296	1.88E+05	-	0.0210%
ken-18	9.32	15,287	93,094	108,381	1.59E+06	-	0.0058%
L1_sixm	15,000.00	184,025	94,953	278,978	2.23E+09	-	0.0014%
Linf_520c	2,833.06	252,615	84,048	336,663	8.25E+07	-	0.0050%
l30	61.91	28	59,338	59,366	1.00E+00	-	0.0656%
lpl3	0.24	39	7,892	7,931	1.93E+03	-	0.0380%
maros-r7	1.23	266	4,570	4,836	9.31E+06	-	0.0465%
model10	4.03	7,820	17,901	25,721	4.91E+03	-	0.1732%
nemspmm1	1.21	3,924	4,691	8,615	3.29E+04	-	0.0656%
nemswrld	13.22	19,471	27,406	46,877	1.03E+04	-	0.1044%
neos1	154.13	1	75,313	75,314	0.00E+00	-	0.0008%
neos2	681.65	1	174,401	174,402	0.00E+00	-	0.0008%
neos3	15,000.00	811	360,732	361,543	1.00E+07	-	0.0002%
neos	306.65	209,397	12,539	221,936	1.53E+07	-	0.0002%
neos5052403	158.75	14,326	211,506	225,832	1.90E+04	-	0.0026%
nl	2.92	7,526	10,924	18,450	2.35E+04	-	0.0202%

ns1644855	613.78	397,962	47,517	445,479	3.52E+07	-	0.0034%
ns1687037	5,221.70	101,222	74,409	175,631	2.62E+05	-	0.0024%
ns1688926	316.04	2,784	106,590	109,374	1.19E+07	-	0.0061%
nsct2	0.27	808	1,646	2,454	8.11E+02	-	0.0128%
nsir2	0.19	1,541	1,322	2,863	6.16E+02	-	0.0434%
nug08-3rd	10,468.26	21,740	315,043	336,783	1.50E+01	-	0.0182%
nug12	23.87	7,117	18,924	26,041	2.00E+00	-	0.1000%
nug15	281.57	23,166	67,490	90,656	1.09E+02	-	0.0501%
nug20	14,696.64	94,515	617,895	712,410	1.02E+02	-	0.0206%
osa-30	0.25	1,623	3,010	4,633	3.11E+03	-	0.0234%
osa-60	0.56	2,373	6,874	9,247	5.65E+03	-	0.0098%
p010	0.25	1,745	388	2,133	4.33E+04	-	0.0413%
pds-100	219.10	8,528	806,668	815,196	3.68E+08	-	0.0025%
pds-20	4.12	0	84,513	84,513	-	5.49E+09	0.0198%
pds-40	26.83	0	357,361	357,361	-	1.05E+10	0.0082%
psched3-3	126.59	90,240	55,204	145,444	8.08E+03	-	0.0015%
pilot87	3.05	4,088	3,098	7,186	1.62E+04	-	0.1100%
pilot	1.27	4,024	1,963	5,987	2.49E+04	-	0.1400%
pltexpa3_16	0.05	0	965	965	-	1.42E+01	0.0078%
qap12	26.10	7,546	19,996	27,542	4.00E+00	-	0.1000%
qap15	237.82	14,187	68,664	82,851	1.00E+00	-	0.0515%
rail02	1,402.02	65,460	229,116	294,576	2.52E+02	-	0.0048%
rail4284	1,881.26	453,733	2,483,833	2,937,566	3.94E+03	-	0.0353%
rat7a	5.41	244	6,417	6,661	9.32E+06	-	0.0465%
s100	23.59	33,211	260,855	294,066	4.31E+02	-	0.0069%
s250r10	43.81	21,782	308,599	330,381	2.61E+02	-	0.0131%
savsched1	98.69	4,181	181,457	185,638	6.67E+04	-	0.0003%
sc205-2r-1600	0.02	286	4	290	5.40E+04	-	0.0203%
scfxm1-2r-256	2.93	7,036	12,803	19,839	4.92E+05	-	0.0061%
self	0.01	0	0	0	-	0.00E+00	0.1042%
seymourl	1.28	3,191	3,219	6,410	4.36E+03	-	0.0215%
sgpf5y6	0.05	1,544	597	2,141	7.60E+06	-	0.0062%
shs1023	103.56	16,640	367,199	383,839	2.90E+06	-	0.0018%
square41	14,662.25	0	1,258,880	1,258,880	-	8.32E+02	0.0570%
stat96v1	278.68	4,530	912,753	917,283	5.00E-01	-	0.0312%
stat96v4	122.23	151,808	166,071	317,879	7.97E+03	-	0.1700%
stocfor3	0.59	2,073	6,382	8,455	7.47E+02	-	0.0158%
storm_1000	423.45	73,317	190,133	263,450	1.13E+04	-	0.0817%
stp3d	2,306.65	304,397	314,352	618,749	1.19E+02	-	0.0020%
support10	712.53	32,623	140,098	172,721	1.69E+03	-	0.0010%
truss	0.52	3,732	8,398	12,130	2.07E+03	-	0.2200%
watson_2	67.62	30,116	140,708	170,824	3.97E+05	-	0.0037%
Average	999.37	35,589	133,127	168,716	-	-	0.0940%

Table S4: Results using the primal CPLEX after initialization with Algorithm 3

Problem	Time	PhIit	PhIIit	Tit	Infeas	Feas	Den
aa01	1.85	7,380	11,516	18,896	7.29E+02	-	0.8966%
aa03	2.34	10,580	10,751	21,331	5.78E+02	-	0.9340%
aa3	1.86	5,714	16,234	21,948	5.50E+02	-	0.9340%
aa5	1.72	5,377	16,659	22,036	5.35E+02	-	0.8938%
aa6	1.41	3,301	15,720	19,021	3.86E+02	-	1.0939%
brazil3	26.95	43,845	18,102	61,947	5.63E+03	-	0.0389%
buildingen	108.85	59,380	122,738	182,118	4.81E+05	-	0.0006%
car4	0.25	0	4,061	4,061	-	5.48E+03	0.1684%
chrom1024-7	19.20	86,780	304	87,084	2.30E+04	-	0.0015%
co5	1.62	1,825	7,346	9,171	9.74E+03	-	0.0334%
co9	6.30	4,173	17,026	21,199	2.11E+04	-	0.0248%
cont1	761.70	43,262	827	44,089	1.45E+03	-	0.0074%
cont11	4,276.39	100,730	33,163	133,893	6.01E+02	-	0.0074%
cq5	0.99	2,690	5,982	8,672	1.13E+04	-	0.0310%
cq9	3.47	4,412	14,927	19,339	2.14E+04	-	0.0204%
cre-b	1.06	2,284	20,501	22,785	1.54E+04	-	0.0277%
cre-d	0.83	1,932	16,652	18,584	1.64E+04	-	0.0399%
d2q06c	0.90	1,382	5,086	6,468	3.29E+04	-	0.0958%
dano3mip	6.97	7,055	28,098	35,153	4.22E+04	-	0.1274%
dbic1	18.80	12	106,852	106,864	1.16E+02	-	0.0030%
dbir2	0.54	614	2,805	3,419	2.09E+02	-	0.0138%
dfl001	5.96	10,999	18,532	29,531	5.31E+03	-	0.0684%
ds-big	405.85	0	802,134	802,134	-	4.52E+04	0.4628%
e18	3.79	2,474	19,378	21,852	2.46E+02	-	0.0044%
ex10	296.50	85,224	0	85,224	6.30E+02	-	0.0016%
ex3sta1	0.27	0	98	98	-	6.31E+01	0.0347%
fit2p	1.08	295	11,582	11,877	8.16E+02	-	0.0333%
fome13	165.87	80,004	156,286	236,290	7.80E+04	-	0.0095%
fxm3_16	1.56	7,585	19,017	26,602	1.50E+06	-	0.0065%
fxm4_6	1.23	9,742	13,217	22,959	4.29E+05	-	0.0153%
ge	1.11	3,280	5,268	8,548	6.42E+05	-	0.0315%
gen1	1.34	3,435	517	3,952	8.72E+03	-	0.3857%
gen2	13.39	12,672	0	12,672	1.50E+04	-	0.1795%
gen4	18.79	13,435	2,389	15,824	1.57E+04	-	0.1355%
gen	1.37	3,435	517	3,952	8.72E+03	-	0.3857%
gosh	0.10	3,056	0	3,056	8.38E+02	-	0.1186%
irish-e	181.39	43,804	224,840	268,644	2.53E+03	-	0.0259%
ken-13	1.31	5,391	23,669	29,060	1.88E+05	-	0.0210%
ken-18	10.44	14,910	93,075	107,985	1.53E+06	-	0.0058%
L1_sixm	15,000.00	184,025	87,587	271,612	2.23E+09	-	0.0014%
Linf_520c	2,229.51	237,970	31,442	269,412	1.12E+07	-	0.0055%
l30	28.49	2,248	42,526	44,774	1.00E+00	-	0.0671%

lpl3	0.35	36	9,217	9,253	1.18E+03	-	0.0380%
maros-r7	1.25	266	4,570	4,836	9.31E+06	-	0.0465%
model10	4.31	7,820	17,901	25,721	4.91E+03	-	0.1660%
nemspmm1	1.11	3,658	4,377	8,035	3.55E+04	-	0.1112%
nemswrld	13.67	20,319	27,004	47,323	1.03E+04	-	0.1031%
neos1	154.49	1	75,313	75,314	0.00E+00	-	0.0008%
neos2	682.48	1	174,401	174,402	0.00E+00	-	0.0008%
neos3	15,000.00	811	352,164	352,975	1.00E+07	-	0.0002%
neos	332.20	224,467	12,495	236,962	1.53E+07	-	0.0002%
neos5052403	154.26	14,326	211,506	225,832	1.90E+04	-	0.0026%
nl	3.01	7,469	11,497	18,966	2.42E+04	-	0.0235%
ns1644855	606.63	397,962	47,517	445,479	3.52E+07	-	0.0034%
ns1687037	5,333.46	101,222	74,409	175,631	2.62E+05	-	0.0024%
ns1688926	540.24	2,115	157,799	159,914	6.90E+08	-	0.0054%
nsct2	0.30	808	1,646	2,454	8.11E+02	-	0.0128%
nsir2	0.17	1,541	1,322	2,863	6.16E+02	-	0.0434%
nug08-3rd	8,519.24	17,471	259,001	276,472	1.30E+01	-	0.0179%
nug12	22.87	6,683	20,536	27,219	2.00E+00	-	0.1100%
nug15	269.65	20,912	69,046	89,958	2.00E+00	-	0.0542%
nug20	15,000.00	72,188	484,904	557,092	1.56E+02	-	0.0215%
osa-30	0.24	1,623	3,010	4,633	3.11E+03	-	0.0234%
osa-60	0.63	2,373	6,874	9,247	5.65E+03	-	0.0098%
p010	0.25	1,800	367	2,167	4.30E+04	-	0.0410%
pds-100	300.82	10,394	845,130	855,524	8.34E+07	-	0.0025%
pds-20	6.46	0	112,355	112,355	-	5.49E+09	0.0201%
pds-40	31.52	0	384,040	384,040	-	1.05E+10	0.0082%
psched3-3	134.14	93,987	55,148	149,135	1.98E+04	-	0.0019%
pilot87	3.18	4,182	3,174	7,356	1.63E+04	-	0.4000%
pilot	1.07	3,334	1,741	5,075	2.45E+04	-	0.3500%
pltexpa3_16	0.05	0	936	936	-	1.42E+01	0.0085%
qap12	24.71	5,764	21,327	27,091	1.00E+00	-	0.1100%
qap15	235.67	13,554	67,206	80,760	1.00E+00	-	0.0578%
rail02	2,446.94	67,430	311,597	379,027	2.80E+02	-	0.0049%
rail4284	1,886.50	453,733	2,483,833	2,937,566	3.94E+03	-	0.0353%
rat7a	5.44	244	6,417	6,661	9.32E+06	-	0.0465%
s100	23.66	27,540	264,453	291,993	4.00E+00	-	0.0069%
s250r10	32.78	6,953	271,725	278,678	9.14E+01	-	0.0131%
savsched1	98.34	4,181	181,457	185,638	6.67E+04	-	0.0003%
sc205-2r-1600	0.02	321	13	334	5.40E+04	-	0.0446%
scfxm1-2r-256	3.70	9,185	14,067	23,252	1.91E+06	-	0.0071%
self	0.01	0	0	0	-	0.00E+00	0.1042%
seymourl	1.04	3,191	3,219	6,410	4.36E+03	-	0.0215%
sgpf5y6	0.05	1,544	597	2,141	7.60E+06	-	0.0062%
shs1023	180.98	16,471	334,488	350,959	2.80E+06	-	0.0018%
square41	829.32	102,911	200,542	303,453	7.24E+03	-	0.0570%
stat96v1	38.75	18	139,331	139,349	2.50E-01	-	0.0396%

stat96v4	113.82	151,808	166,071	317,879	7.97E+03	-	0.1700%
stocfor3	0.53	2,001	6,006	8,007	7.34E+02	-	0.0159%
storm_1000	396.24	70,418	192,933	263,351	6.74E+04	-	0.0827%
stp3d	1,798.47	367,963	221,056	589,019	2.00E+03	-	0.0017%
support10	711.50	32,623	140,098	172,721	1.69E+03	-	0.0010%
truss	0.66	3,699	11,973	15,672	8.43E+02	-	0.3000%
watson_2	75.75	30,010	138,061	168,071	3.13E+05	-	0.0038%
Average	838.28	36,906	111,929	148,835	-	-	0.1069%

Table S5: Results using the primal CPLEX after initialization with CPLEX default crash procedure

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**	**Den**
aa01	2.07	7,768	11,191	18,959	4.02E+02	-	0.6495%
aa03	2.34	7,774	14,109	21,883	2.95E+02	-	0.6716%
aa3	1.93	6,061	15,252	21,313	2.79E+02	-	0.6654%
aa5	1.92	4,652	21,005	25,657	2.99E+02	-	0.6503%
aa6	1.41	3,475	15,873	19,348	2.56E+02	-	0.7456%
brazil3	25.26	42,285	15,805	58,090	1.78E+03	-	0.0308%
buildingen	2,238.68	28,010	382,787	410,797	4.81E+05	-	0.0005%
car4	0.27	342	3,037	3,379	9.49E+02	-	0.5283%
chrom1024-7	37.49	50,387	14,840	65,227	3.69E+04	-	0.0015%
co5	1.62	2,052	6,864	8,916	9.77E+03	-	0.0511%
co9	5.48	3,925	14,504	18,429	2.10E+04	-	0.0274%
cont1	1,647.41	60,714	915	61,629	7.95E+02	-	0.0057%
cont11	4,404.29	105,961	29,090	135,051	8.33E+02	-	0.0074%
cq5	1.06	2,554	7,146	9,700	1.08E+04	-	0.0658%
cq9	3.55	4,412	14,927	19,339	2.14E+04	-	0.0349%
cre-b	1.12	2,033	22,189	24,222	1.54E+04	-	0.0262%
cre-d	0.73	1,941	17,514	19,455	1.64E+04	-	0.0359%
d2q06c	1.21	1,998	4,420	6,418	6.08E+04	-	0.1792%
dano3mip	7.60	19,307	21,486	40,793	6.37E+04	-	0.0829%
dbic1	19.64	3,957	101,232	105,189	2.62E+02	-	0.0052%
dbir2	0.92	864	15,287	16,151	2.04E+02	-	0.1913%
dfl001	6.45	9,968	19,736	29,704	1.15E+04	-	0.0682%
ds-big	254.63	44,389	447,485	491,874	3.36E+02	-	1.0788%
e18	5.14	2,641	19,978	22,619	6.45E+03	-	0.0075%
ex10	110.77	51,910	0	51,910	4.27E+03	-	0.0016%
ex3sta1	22.85	14,879	215	15,094	7.96E+03	-	0.0282%
fit2p	1.45	188	13,619	13,807	1.37E+03	-	0.0333%
fome13	191.86	74,558	144,176	218,734	1.03E+05	-	0.0087%
fxm3_16	3.58	10,006	19,967	29,973	2.49E+06	-	0.0076%
fxm4_6	1.27	9,742	13,217	22,959	4.29E+05	-	0.0125%

ge	1.13	3,280	5,268	8,548	6.42E+05	-	0.0287%
gen1	1.88	3,728	385	4,113	8.72E+03	-	9.3176%
gen2	12.54	11,794	0	11,794	9.19E+03	-	0.4211%
gen4	18.22	12,929	2,183	15,112	1.20E+05	-	0.5854%
gen	1.88	3,728	385	4,113	8.72E+03	-	9.3176%
gosh	0.13	3,110	0	3,110	8.26E+02	-	0.2478%
irish-e	187.06	35,885	212,233	248,118	2.38E+03	-	0.0367%
ken-13	1.23	3,479	22,963	26,442	1.70E+05	-	0.0213%
ken-18	11.42	11,305	91,007	102,312	7.84E+05	-	0.0059%
L1_sixm	9,728.08	96,907	77,128	174,035	2.96E+07	-	0.0014%
Linf_520c	15,000.00	645,647	0	645,647	1.05E+05	-	0.0039%
l30	12.61	857	17,189	18,046	1.00E+00	-	0.1839%
lpl3	0.16	39	4,548	4,587	1.90E+03	-	0.0377%
maros-r7	1.11	870	3,128	3,998	4.00E+06	-	0.2363%
model10	4.30	7,820	17,901	25,721	4.91E+03	-	0.1713%
nemspmm1	1.12	3,658	4,377	8,035	3.55E+04	-	0.3186%
nemswrld	12.59	18,046	27,023	45,069	3.39E+02	-	0.1041%
neos1	60.68	1	24,409	24,410	0.00E+00	-	0.0008%
neos2	321.05	1	93,276	93,277	0.00E+00	-	0.0008%
neos3	15,000.00	812	310,291	311,103	7.90E+08	-	0.0002%
neos	262.71	194,562	11,938	206,500	1.62E+07	-	0.0003%
neos5052403	193.49	14,803	228,960	243,763	1.90E+04	-	0.0035%
nl	2.77	9,067	10,851	19,918	2.42E+04	-	0.0284%
ns1644855	154.00	131,086	24,451	155,537	1.07E+07	-	0.1272%
ns1687037	4,876.91	88,140	78,247	166,387	2.77E+05	-	0.0066%
ns1688926	300.64	2,784	106,590	109,374	1.19E+07	-	0.2200%
nsct2	1.16	408	20,637	21,045	9.16E+02	-	0.1200%
nsir2	0.70	565	14,606	15,171	1.54E+03	-	0.5900%
nug08-3rd	15,000.00	15,709	311,041	326,750	1.30E+01	-	0.0169%
nug12	24.30	6,518	20,295	26,813	2.00E+00	-	0.1700%
nug15	253.69	13,357	65,577	78,934	1.00E+00	-	0.0807%
nug20	15,000.00	45,068	702,031	747,099	1.00E+00	-	0.0367%
osa-30	0.19	1,599	2,782	4,381	3.11E+03	-	0.0246%
osa-60	1.14	2,373	6,883	9,256	5.65E+03	-	0.0100%
p010	0.71	3,808	2,207	6,015	7.82E+04	-	0.0262%
pds-100	260.17	214	838,735	838,949	1.15E+04	-	0.0025%
pds-20	4.95	0	91,024	91,024	-	5.49E+09	0.0197%
pds-40	29.82	1	373,187	373,188	0.00E+00	-	0.0081%
psched3-3	127.80	72,603	72,254	144,857	9.41E+03	-	0.0016%
pilot87	3.23	4,023	2,887	6,910	1.81E+04	-	0.4400%
pilot	1.37	3,490	1,862	5,352	2.46E+04	-	0.4500%
pltexpa3_16	1.35	9,917	4,757	14,674	3.92E+05	-	0.0169%
qap12	24.90	8,718	19,471	28,189	1.30E+01	-	0.1800%
qap15	257.25	14,754	69,248	84,002	1.40E+01	-	0.0885%
rail02	2,734.70	36,297	344,931	381,228	2.76E+02	-	0.0044%
rail4284	1,674.74	381,936	2,549,213	2,931,149	4.16E+03	-	0.0245%

rat7a	4.26	2,481	1,619	4,100	9.38E+06	-	2.7300%
s100	28.12	19,636	272,206	291,842	5.06E+00	-	0.0069%
s250r10	29.52	382	224,237	224,619	9.61E+00	-	0.0131%
savsched1	104.06	3,868	193,572	197,440	4.70E+04	-	0.0003%
sc205-2r-1600	0.05	572	5	577	2.78E+05	-	0.0294%
scfxm1-2r-256	4.79	10,169	15,010	25,179	7.96E+05	-	0.0111%
self	0.20	509	0	509	4.17E-03	-	4.7077%
seymourl	1.16	2,948	3,203	6,151	4.39E+03	-	0.0542%
sgpf5y6	0.09	880	655	1,535	1.14E+09	-	0.0079%
shs1023	202.39	5,727	290,340	296,067	3.71E+04	-	0.0017%
square41	1,100.30	89,125	205,013	294,138	1.39E+03	-	0.0570%
stat96v1	146.12	2,428	577,866	580,294	2.50E-01	-	0.0907%
stat96v4	156.13	151,808	166,071	317,879	7.97E+03	-	0.1600%
stocfor3	0.78	3,317	7,060	10,377	2.47E+03	-	0.0413%
storm_1000	441.44	44,678	165,538	210,216	1.35E+07	-	0.1200%
stp3d	2,050.59	391,397	268,585	659,982	2.17E+02	-	0.0016%
support10	957.34	32,640	139,783	172,423	1.69E+03	-	0.0010%
truss	0.56	2,578	9,050	11,628	1.10E+03	-	0.2100%
watson_2	47.25	34,074	69,823	103,897	3.95E+06	-	0.0053%
Average	1,008.94	34,775	115,009	149,784	-	-	0.3988%

Table S6: Results using the dual CPLEX after initialization with Algorithm 1

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**
aa01	0.83	925	3,728	4,653	-	-5.53E+04
aa03	0.73	838	3,178	4,016	-	-4.79E+04
aa3	0.66	1,305	2,193	3,498	-	-4.79E+04
aa5	0.86	613	4,175	4,788	-	-5.25E+04
aa6	0.43	597	1,723	2,320	-	-2.69E+04
brazil3	17.27	956	29,873	30,829	2.82E+04	-
buildingen	14.39	48,011	109,802	157,813	1.50E+03	-
car4	0.15	341	1,005	1,346	3.50E+03	-
chrom1024-7	115.36	0	97,237	97,237	-3.00E+00	-
co5	0.77	361	5,467	5,828	4.65E+05	-
co9	3.41	708	13,494	14,202	6.18E+05	-
cont1	1,745.81	38	73,824	73,862	-	-8.78E-03
cont11	15,000.00	6,599	208,640	215,239	-	-6.10E+01
cq5	0.57	158	4,926	5,084	6.88E+03	-
cq9	1.97	277	13,968	14,245	1.82E+04	-
cre-b	0.68	0	9,210	9,210	-	-2.27E+07
cre-d	0.32	0	6,889	6,889	-	-2.41E+07
d2q06c	0.82	1,166	4,315	5,481	2.09E+04	-
dano3mip	16.83	69	48,620	48,689	-	-5.76E+02

dbic1	31.78	2,575	49,582	52,157	9.00E+08	-
dbir2	0.27	700	435	1,135	3.02E+09	-
dfl001	10.17	20,666	19,603	40,269	9.73E+10	-
ds-big	357.63	1,772	72,174	73,946	2.04E+08	-
e18	1.51	14	5,438	5,452	3.10E+01	-
ex10	4,355.58	0	545,762	545,762	-	-7.00E+01
ex3sta1	10.06	887	8,558	9,445	1.00E+00	-
fit2p	1.02	1,765	4,725	6,490	2.39E+05	-
fome13	212.16	150,084	170,188	320,272	7.88E+11	-
fxm3_16	1.45	8,203	37,223	45,426	1.78E+03	-
fxm4_6	0.69	6,030	18,666	24,696	6.82E+02	-
ge	0.43	119	4,835	4,954	2.64E+02	-
gen1	0.33	562	752	1,314	5.96E+03	-
gen2	4.47	0	3,766	3,766	-	0.00E+00
gen4	3.57	1,811	2,151	3,962	1.57E+04	-
gen	0.34	562	752	1,314	5.96E+03	-
gosh	0.11	1,307	439	1,746	2.82E+02	-
irish-e	33.28	609	39,363	39,972	6.42E+05	-
ken-13	0.90	16,902	12,755	29,657	-	8.82E+09
ken-18	6.06	52,793	50,331	103,124	-	4.57E+10
L1_sixm	7,476.34	13,970	184,956	198,926	1.90E+03	-
Linf_520c	15,000.00	2,419	403,451	405,870	1.28E+00	-
l30	4.85	6,517	2,805	9,322	7.58E+02	-
lpl3	0.55	1,123	5,064	6,187	2.56E+11	-
maros-r7	0.51	0	2,708	2,708	-	-1.00E+07
model10	6.88	593	24,190	24,783	4.09E+05	-
nemspmm1	1.26	449	6,242	6,691	4.70E+02	-
nemswrld	14.91	184	29,168	29,352	1.14E+03	-
neos1	130.23	257	28,078	28,335	4.10E+01	-
neos2	263.93	661	39,976	40,637	-	-4.76E+04
neos3	15,000.00	0	177,214	177,214	-	0.00E+00
neos	194.59	26,156	78,980	105,136	-	-2.25E+08
neos5052403	280.13	0	122,852	122,852	-	-1.79E+02
nl	0.80	317	8,278	8,595	4.28E+04	-
ns1644855	145.28	0	57,103	57,103	-	-1.98E+05
ns1687037	10,632.00	37,242	956,499	993,741	-	8.44E+02
ns1688926	8.39	1	4,115	4,116	0.00E+00	-
nsct2	0.13	1,072	937	2,009	3.76E+09	-
nsir2	0.04	389	620	1,009	1.78E+09	-
nug08-3rd	645.07	8,744	104,596	113,340	5.75E+03	-
nug12	31.78	11,031	61,843	72,874	-	-5.23E+02
nug15	370.94	33,587	229,361	262,948	-	-1.04E+03
nug20	15,000.00	385,232	1,187,806	1,573,038	2.47E+06	-
osa-30	2.61	0	4,691	4,691	-	-1.96E+06
osa-60	10.37	0	8,953	8,953	-	-3.65E+06
p010	0.25	37	9,366	9,403	5.40E+03	-

pds-100	27.04	29,914	154,569	184,483	1.11E+09	-
pds-20	2.61	11,006	22,267	33,273	-	5.49E+09
pds-40	8.18	29,389	47,798	77,187	1.93E+08	-
psched3-3	355.77	665	154,652	155,317	5.08E+05	-
pilot87	4.66	2,027	8,527	10,554	1.10E+00	-
pilot	0.70	708	2,510	3,218	2.73E+00	-
pltexpa3_16	0.30	2,230	14,179	16,409	1.00E+00	-
qap12	34.16	10,262	63,865	74,127	-	-5.23E+02
qap15	459.30	38,638	245,547	284,185	-	-1.04E+03
rail02	737.29	1,346	612,196	613,542	6.27E+03	-
rail4284	1,804.28	214	56,435	56,649	6.36E+04	-
rat7a	1.89	0	3,049	3,049	-	-1.06E+07
s100	583.33	15,315	275,391	290,706	1.38E+03	-
s250r10	41.28	4,833	90,772	95,605	6.10E+03	-
savsched1	15,000.00	295,621	1,219,843	1,515,464	2.25E+06	-
sc205-2r-1600	0.04	4	590	594	-	0.00E+00
scfxm1-2r-256	1.56	11,037	18,803	29,840	2.85E+01	-
self	0.03	0	0	0	-	0.00E+00
seymourl	0.52	1	3,234	3,235	-	-2.41E+02
sgpf5y6	0.08	343	5,760	6,103	-	-5.68E+03
shs1023	105.19	35,463	246,899	282,362	1.30E+07	-
square41	497.79	2,391	62,902	65,293	5.97E+05	-
stat96v1	30.93	0	15,422	15,422	-	-3.74E+00
stat96v4	52.52	39,279	33,186	72,465	1.00E+00	-
stocfor3	0.55	4,971	5,362	10,333	9.98E+04	-
storm_1000	83.90	183,681	668,546	852,227	6.39E+05	-
stp3d	195.52	377	134,901	135,278	1.73E+03	-
support10	175.87	0	89,581	89,581	-	-3.38E+00
truss	2.27	383	16,997	17,380	-	-4.59E+05
watson_2	19.83	176,559	4,482	181,041	5.75E+01	-
Average	1,130.56	18,494	102,188	120,683	-	-

Table S7: Results using the dual CPLEX after initialization with Algorithm 2

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**
aa01	1.00	1,160	4,334	5,494	-	-5.53E+04
aa03	0.67	1,156	2,728	3,884	-	-4.79E+04
aa3	0.61	734	2,159	2,893	-	-4.79E+04
aa5	0.81	1,373	3,852	5,225	-	-5.25E+04
aa6	0.44	719	1,484	2,203	-	-2.69E+04
brazil3	14.31	1,114	25,824	26,938	-	5.70E+01
buildingen	13.34	65,561	114,014	179,575	1.78E+03	-
car4	0.13	341	1,005	1,346	3.50E+03	-

chrom1024-7	115.18	0	95,594	95,594	-	-3.00E+00
co5	0.64	324	4,686	5,010	4.53E+05	-
co9	3.15	619	12,048	12,667	5.55E+05	-
cont1	1,726.44	38	73,824	73,862	-	-8.80E-03
cont11	15,000.00	6,599	211,712	218,311	-	-6.10E+01
cq5	0.57	158	4,926	5,084	6.88E+03	-
cq9	1.96	266	13,024	13,290	8.73E+03	-
cre-b	0.74	0	9,929	9,929	-	-2.27E+07
cre-d	0.39	0	6,808	6,808	-	-2.41E+07
d2q06c	0.70	976	3,999	4,975	1.65E+04	-
dano3mip	16.40	32	43,684	43,716	-	-5.76E+02
dbic1	30.16	2,575	49,654	52,229	9.00E+08	-
dbir2	0.28	700	435	1,135	3.02E+09	-
dfl001	12.75	21,708	21,248	42,956	9.74E+10	-
ds-big	399.71	1,772	72,174	73,946	-	-8.68E+01
e18	1.70	1	5,917	5,918	0.00E+00	-
ex10	4,813.17	1	546,468	546,469	-	-1.00E+02
ex3sta1	0.69	93	331	424	3.86E+02	-
fit2p	1.01	1,765	4,725	6,490	2.39E+05	-
fome13	155.33	96,998	146,847	243,845	7.42E+11	-
fxm3_16	2.24	10,879	40,457	51,336	6.13E+01	-
fxm4_6	0.89	6,358	16,550	22,908	6.19E+01	-
ge	0.44	119	4,835	4,954	2.64E+02	-
gen1	0.28	405	685	1,090	5.65E+03	-
gen2	4.50	0	3,878	3,878	-	0.00E+00
gen4	3.90	1,781	1,689	3,470	1.53E+04	-
gen	0.30	405	685	1,090	5.65E+03	-
gosh	0.13	1,456	473	1,929	1.10E+02	-
irish-e	33.81	609	39,363	39,972	6.42E+05	-
ken-13	0.94	16,713	12,379	29,092	-	8.82E+09
ken-18	6.78	51,310	51,022	102,332	-	4.57E+10
L1_sixm	7,543.06	13,970	184,956	198,926	-	-4.63E+03
Linf_520c	15,000.00	2,419	403,451	405,870	1.28E+00	-
l30	4.12	6,162	2,588	8,750	1.75E+03	-
lpl3	0.63	1,335	4,782	6,117	2.66E+11	-
maros-r7	0.56	0	2,708	2,708	-	-1.00E+07
model10	11.48	75	26,144	26,219	2.05E+03	-
nemspmm1	1.18	1,771	6,728	8,499	3.27E+05	-
nemswrld	17.23	441	35,414	35,855	1.33E+04	-
neos1	131.60	257	28,078	28,335	-	-4.67E+04
neos2	267.30	661	39,976	40,637	-	-4.76E+04
neos3	15,000.00	0	182,449	182,449	-	0.00E+00
neos	236.96	26,789	82,434	109,223	-	-2.25E+08
neos5052403	283.86	0	122,852	122,852	-	-1.79E+02
nl	0.82	118	8,572	8,690	1.05E+04	-
ns1644855	143.47	0	54,268	54,268	-	-1.98E+05

ns1687037	10,636.30	37,242	956,499	993,741	-	8.44E+02
ns1688926	12.83	0	5,375	5,375	-	-3.91E+02
nsct2	0.13	1,072	937	2,009	3.76E+09	-
nsir2	0.06	389	620	1,009	1.78E+09	-
nug08-3rd	1,678.77	12,323	155,379	167,702	-	-2.14E+02
nug12	39.25	9,028	69,365	78,393	-	-5.23E+02
nug15	438.30	36,380	247,270	283,650	-	-1.04E+03
nug20	15,000.00	416,051	876,378	1,292,429	2.24E+06	-
osa-30	2.69	0	4,692	4,692	-	-1.96E+06
osa-60	11.53	0	8,953	8,953	-	-3.65E+06
p010	0.31	19	10,026	10,045	4.31E+03	-
pds-100	23.16	24,339	131,502	155,841	-	-1.09E+10
pds-20	2.21	12,376	21,397	33,773	-	5.49E+09
pds-40	7.78	25,394	49,052	74,446	-	1.05E+10
psched3-3	126.13	619	66,177	66,796	4.04E+05	-
pilot87	5.10	2,563	8,541	11,104	1.09E+00	-
pilot	0.72	720	3,074	3,794	2.73E+00	-
pltexpa3_16	0.36	2,203	14,382	16,585	1.00E+00	-
qap12	32.35	10,528	57,367	67,895	-	-5.23E+02
qap15	424.75	35,134	235,138	270,272	-	-1.04E+03
rail02	412.39	912	327,693	328,605	-	2.06E+02
rail4284	1,786.85	214	56,435	56,649	-	-1.03E+03
rat7a	1.92	0	3,049	3,049	-	-1.06E+07
s100	565.23	14,355	252,034	266,389	1.56E+03	-
s250r10	55.30	14,711	93,927	108,638	2.63E+04	-
savsched1	15,000.00	295,621	1,328,291	1,623,912	-	0.00E+00
sc205-2r-1600	0.10	875	1,148	2,023	-	0.00E+00
scfxm1-2r-256	2.18	11,066	20,126	31,192	3.31E+01	-
self	0.02	0	0	0	-	0.00E+00
seymourl	0.65	1	3,234	3,235	-	-2.41E+02
sgpf5y6	0.09	111	6,776	6,887	-	-5.68E+03
shs1023	87.28	24,623	244,651	269,274	5.80E+06	-
square41	179.24	4	24,047	24,051	-	-8.84E+00
stat96v1	67.19	11,966	16,090	28,056	2.80E+04	-
stat96v4	57.69	18,589	44,725	63,314	1.00E+00	-
stocfor3	0.45	4,809	4,683	9,492	8.28E+04	-
storm_1000	72.67	165,653	655,301	820,954	5.13E+05	-
stp3d	147.24	43	115,181	115,224	-	-4.52E+02
support10	173.15	0	89,581	89,581	-	-3.38E+00
truss	2.16	434	16,251	16,685	-	-4.59E+05
watson_2	21.72	178,109	4,482	182,591	6.96E+01	-
Average	1,137.74	18,098	95,481	113,579	-	-

Table S8: Results using the dual CPLEX after initialization with Algorithm 3

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**
aa01	0.94	311	4,233	4,544	-	-5.53E+04
aa03	0.50	659	2,406	3,065	-	-4.79E+04
aa3	0.50	545	2,170	2,715	-	-4.79E+04
aa5	0.69	725	3,803	4,528	-	-5.25E+04
aa6	0.39	877	1,667	2,544	-	-2.69E+04
brazil3	13.82	661	25,618	26,279	-	5.70E+01
buildingen	13.06	65,561	114,014	179,575	1.78E+03	-
car4	0.17	341	1,005	1,346	3.50E+03	-
chrom1024-7	118.70	0	96,266	96,266	-	-3.00E+00
co5	0.68	376	4,787	5,163	2.28E+06	-
co9	3.27	773	12,135	12,908	9.94E+05	-
cont1	1,725.01	38	73,824	73,862	-	-8.80E-03
cont11	15,000.00	6,599	210,352	216,951	-	-6.10E+01
cq5	0.58	158	4,926	5,084	6.88E+03	-
cq9	1.88	147	14,428	14,575	7.13E+02	-
cre-b	0.74	37	10,193	10,230	5.79E+05	-
cre-d	0.44	49	6,983	7,032	9.06E+05	-
d2q06c	0.80	1,081	4,089	5,170	1.79E+04	-
dano3mip	15.86	19	45,992	46,011	-	-5.76E+02
dbic1	29.97	2,575	49,654	52,229	9.00E+08	-
dbir2	0.27	700	435	1,135	3.02E+09	-
dfl001	11.99	23,032	18,988	42,020	8.93E+10	-
ds-big	400.15	1,772	72,174	73,946	-	-8.68E+01
e18	1.81	3	5,872	5,875	1.40E+01	-
ex10	4,753.58	0	582,121	582,121	-	-7.40E+01
ex3sta1	0.55	101	265	366	1.00E+00	-
fit2p	1.09	1,765	4,725	6,490	2.39E+05	-
fome13	192.50	128,216	163,886	292,102	7.61E+11	-
fxm3_16	2.22	11,001	41,680	52,681	5.98E+02	-
fxm4_6	1.89	6,643	22,789	29,432	8.58E+00	-
ge	0.53	119	4,835	4,954	2.64E+02	-
gen1	0.37	500	919	1,419	5.79E+03	-
gen2	4.80	0	3,705	3,705	-	0.00E+00
gen4	4.10	1,870	2,016	3,886	1.50E+04	-
gen	0.39	500	919	1,419	5.79E+03	-
gosh	0.09	1,318	385	1,703	4.30E+02	-
irish-e	34.23	609	39,363	39,972	6.42E+05	-
ken-13	0.85	15,703	12,323	28,026	-	8.82E+09
ken-18	6.12	52,022	49,372	101,394	-	4.57E+10
L1_sixm	6,809.64	13,970	184,956	198,926	-	-4.63E+03
Linf_520c	3,040.32	2	266,742	266,744	1.00E+00	-
l30	4.75	6,431	2,737	9,168	5.60E+03	-
lpl3	0.50	1,246	4,700	5,946	3.32E+11	-

maros-r7	0.55	0	2,708	2,708	-	-1.00E+07
model10	11.40	75	26,144	26,219	2.05E+03	-
nemspmm1	1.20	449	6,242	6,691	4.70E+02	-
nemswrld	13.65	354	26,923	27,277	1.09E+04	-
neos1	132.32	257	28,078	28,335	-	-4.67E+04
neos2	258.27	661	39,976	40,637	-	-4.76E+04
neos3	15,000.00	0	176,729	176,729	-	0.00E+00
neos	280.34	28,156	82,669	110,825	-	-2.25E+08
neos5052403	280.77	0	122,852	122,852	-	-1.79E+02
nl	0.98	135	8,839	8,974	1.71E+04	-
ns1644855	138.36	0	54,268	54,268	-	-1.98E+05
ns1687037	10,466.08	37,242	956,499	993,741	-	8.44E+02
ns1688926	11.51	5,003	7,169	12,172	2.00E-06	-
nsct2	0.13	1,072	937	2,009	3.76E+09	-
nsir2	0.06	389	620	1,009	1.78E+09	-
nug08-3rd	467.42	5,149	81,334	86,483	-	-2.14E+02
nug12	36.37	8,452	63,899	72,351	-	-5.23E+02
nug15	383.78	29,909	227,430	257,339	-	-1.04E+03
nug20	15,000.00	166,947	944,570	1,111,517	1.42E+06	-
osa-30	2.60	0	4,692	4,692	-	-1.96E+06
osa-60	11.47	0	8,953	8,953	-	-3.65E+06
p010	0.26	42	9,265	9,307	5.88E+03	-
pds-100	27.00	31,886	150,115	182,001	-	-1.09E+10
pds-20	2.82	10,581	24,793	35,374	-	5.49E+09
pds-40	5.65	26,692	40,344	67,036	-	1.05E+10
psched3-3	304.81	646	141,716	142,362	4.36E+05	-
pilot87	5.72	1,958	9,546	11,504	1.10E+00	-
pilot	0.59	665	2,678	3,343	2.73E+00	-
pltexpa3_16	0.33	2,228	14,218	16,446	1.00E+00	-
qap12	35.34	9,708	56,082	65,790	-	-5.23E+02
qap15	465.74	27,132	236,135	263,267	-	-1.04E+03
rail02	724.88	892	666,540	667,432	-	2.06E+02
rail4284	1,779.70	214	56,435	56,649	-	-1.03E+03
rat7a	1.88	0	3,049	3,049	-	-1.06E+07
s100	534.42	32,125	240,055	272,180	1.47E+03	-
s250r10	110.61	60,081	66,812	126,893	6.78E+03	-
savsched1	15,000.00	295,621	1,349,777	1,645,398	-	0.00E+00
sc205-2r-1600	0.07	1,605	171	1,776	-	0.00E+00
scfxm1-2r-256	2.84	11,987	23,327	35,314	5.80E+02	-
self	0.02	0	0	0	-	0.00E+00
seymourl	0.65	1	3,234	3,235	-	-2.41E+02
sgpf5y6	0.09	111	6,776	6,887	-	-5.68E+03
shs1023	70.66	25,835	196,927	222,762	5.92E+06	-
square41	494.27	362	58,991	59,353	-	-8.84E+00
stat96v1	31.18	0	15,180	15,180	-	-3.74E+00
stat96v4	57.54	18,589	44,725	63,314	1.00E+00	-

stocfor3	0.49	4,959	3,864	8,823	8.91E+04	-
storm_1000	73.69	195,886	648,149	844,035	4.49E+05	-
stp3d	181.54	24	132,354	132,378	-	-4.52E+02
support10	172.55	0	89,581	89,581	-	-3.38E+00
truss	2.94	231	20,184	20,415	-	-4.59E+05
watson_2	13.82	175,662	4,332	179,994	5.04E+01	-
Average	997.90	16,516	98,540	115,056	-	-

Table S9: Results using the dual CPLEX after initialization with CPLEX default crash procedure

Problem	**Time**	**PhIit**	**PhIIit**	**Tit**	**Infeas**	**Feas**
aa01	0.74	0	3,668	3,668	-	-5.51E+04
aa03	0.37	0	2,294	2,294	-	-4.77E+04
aa3	0.36	0	2,004	2,004	-	-4.77E+04
aa5	0.67	0	3,473	3,473	-	-5.23E+04
aa6	0.23	0	1,540	1,540	-	-2.69E+04
brazil3	14.94	0	26,917	26,917	-	-2.00E+00
buildingen	13.20	28,934	113,277	142,211	1.78E+03	-
car4	0.18	0	1,137	1,137	-	-3.55E+01
chrom1024-7	137.36	0	89,575	89,575	-	-3.00E+00
co5	0.82	159	5,080	5,239	1.00E+04	-
co9	2.87	283	11,685	11,968	1.20E+04	-
cont1	536.14	2	22,640	22,642	0.00E+00	-
cont11	6,985.17	48,180	204,097	252,277	4.00E+04	-
cq5	0.64	83	5,038	5,121	4.67E+02	-
cq9	2.22	153	14,511	14,664	7.13E+02	-
cre-b	0.70	0	10,486	10,486	-	-2.27E+07
cre-d	0.40	0	6,902	6,902	-	-2.41E+07
d2q06c	0.83	419	4,782	5,201	7.64E+02	-
dano3mip	13.14	16	32,606	32,622	-	-5.76E+02
dbic1	37.49	18	50,122	50,140	7.14E+08	-
dbir2	0.47	0	9,022	9,022	-	-2.03E+06
dfl001	5.87	0	17,872	17,872	-	-1.12E+07
ds-big	459.73	133	70,734	70,867	-	-8.56E+01
e18	1.62	0	5,466	5,466	-	-3.70E+02
ex10	4,637.19	0	534,091	534,091	-	-9.90E+01
ex3sta1	8.48	968	7,549	8,517	1.00E+00	-
fit2p	1.17	0	5,353	5,353	-	-6.85E+04
fome13	93.75	0	131,119	131,119	-	-9.00E+07
fxm3_16	5.11	10,969	40,744	51,713	8.59E+00	-
fxm4_6	1.56	6,832	22,199	29,031	8.58E+00	-
ge	0.46	113	4,714	4,827	2.61E+02	-

gen1	0.07	0	248	248	-	0.00E+00
gen2	4.49	0	3,088	3,088	-	-3.29E+00
gen4	0.41	0	560	560	-	0.00E+00
gen	0.07	0	248	248	-	0.00E+00
gosh	0.23	1,520	1,088	2,608	1.16E+02	-
irish-e	185.06	12	108,395	108,407	9.48E+02	-
ken-13	0.30	5	13,522	13,527	-	8.82E+09
ken-18	1.97	0	50,523	50,523	-	-2.88E+13
L1_sixm	5,371.34	1,354	170,338	171,692	-	-4.63E+03
Linf_520c	15,000.00	196	215,865	216,061	1.00E+00	-
l30	6.62	6,990	4,230	11,220	2.55E+02	-
lpl3	0.17	72	3,898	3,970	7.10E+04	-
maros-r7	0.79	0	2,746	2,746	-	-6.94E+05
model10	12.14	1,501	25,372	26,873	2.05E+03	-
nemspmm1	1.29	512	5,831	6,343	4.70E+02	-
nemswrld	31.44	172	54,324	54,496	1.14E+03	-
neos1	128.61	254	18,768	19,022	-	-4.67E+04
neos2	258.39	741	28,323	29,064	-	-4.76E+04
neos3	15,000.00	0	173,985	173,985	-	0.00E+00
neos	248.26	27,284	79,193	106,477	-	-2.25E+08
neos5052403	328.77	0	117,360	117,360	-	-1.79E+02
nl	0.91	33	9,189	9,222	7.21E+02	-
ns1644855	77.09	10	28,682	28,692	3.67E+00	-
ns1687037	15,000.00	65,113	902,000	967,113	-	3.34E+00
ns1688926	10.52	0	5,348	5,348	-	-3.91E+02
nsct2	0.39	0	6,743	6,743	-	-4.83E+06
nsir2	0.12	0	2,754	2,754	-	-3.43E+06
nug08-3rd	547.60	0	78,444	78,444	-	-2.13E+02
nug12	45.80	0	92,294	92,294	-	-5.21E+02
nug15	582.97	0	366,689	366,689	-	-1.04E+03
nug20	15,000.00	258,760	913,794	1,172,554	-	2.18E+03
osa-30	2.62	0	4,690	4,690	-	-1.96E+06
osa-60	12.23	0	9,589	9,589	-	-3.65E+06
p010	0.23	0	13,671	13,671	-	-1.02E+06
pds-100	24.97	6,430	102,010	108,440	-	-1.09E+10
pds-20	2.10	460	17,091	17,551	-	5.49E+09
pds-40	6.61	1,281	44,535	45,816	-	1.05E+10
psched3-3	204.99	174	87,587	87,761	1.75E+04	-
pilot87	5.22	1,752	9,397	11,149	6.76E-01	-
pilot	1.00	771	3,203	3,974	2.71E+00	-
pltexpa3_16	0.49	1,062	19,294	20,356	-	-5.49E+01
qap12	42.77	0	81,061	81,061	-	-5.19E+02
qap15	621.47	0	362,522	362,522	-	-1.04E+03
rail02	648.53	205	542,029	542,234	-	-5.59E+03
rail4284	1,603.98	0	50,428	50,428	-	-1.03E+03
rat7a	2.95	0	2,870	2,870	-	-1.27E+06

s100	545.92	299	197,393	197,692	-	-1.26E+03
s250r10	40.71	45	79,080	79,125	-	-4.19E+02
savsched1	476.86	0	129,056	129,056	-	-1.69E+02
sc205-2r-1600	0.02	4	197	201	-	0.00E+00
scfxm1-2r-256	2.36	1,822	24,627	26,449	4.60E+00	-
self	0.03	0	0	0	-	0.00E+00
seymourl	0.65	0	3,354	3,354	-	-2.40E+02
sgpf5y6	0.17	6,466	4,877	11,343	-	-5.68E+03
shs1023	72.28	0	179,323	179,323	-	-5.88E+03
square41	122.34	0	17,177	17,177	-	-7.84E+00
stat96v1	26.22	0	13,217	13,217	-	-3.74E+00
stat96v4	51.37	41,561	27,851	69,412	1.00E+00	-
stocfor3	0.61	4,456	3,618	8,074	7.52E+03	-
storm_1000	65.66	0	693,373	693,373	-	-1.53E+07
stp3d	236.80	4	155,754	155,758	-	-4.52E+02
support10	171.01	0	93,614	93,614	-	-3.38E+00
truss	2.66	0	19,693	19,693	-	-4.59E+05
watson_2	8.97	163,972	4,698	168,670	2.36E+00	-
Average	903.31	7,290	83,594	90,884	-	-