A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem

Francisco Ortega and Laurence Wolsey
CORE and INMA
Université Catholique de Louvain

September 28, 2000

Abstract

We present a branch-and-cut algorithm to solve the single commodity uncapacitated fixed charge network flow problem, which includes the Steiner tree problem, uncapacitated lot-sizing problems, and the fixed charge transportation problem as special cases. The cuts used are simple *dicut* inequalities and their variants. A crucial problem when separating these inequalities is to find the right cut set on which to generate the inequalities. The prototype branch-and-cut system, `bc-nd` includes a separation heuristic for the dicut inequalities, and problem specific primal heuristics, branching and pruning rules. Computational results show that `bc-nd` is competitive compared to a variety of special purpose algorithms for problems with explicit flow costs.

We also examine how general purpose MIP systems perform on such problems when provided with formulations that have been tightened a priori with dicut inequalities.

Keywords: Network Design, Fixed Charge, Branch and Cut, Dicut Inequalities, Branching, Heuristics, Minimum cost flow.

1 Introduction

The single commodity uncapacitated fixed charge network flow problem (UFC) is one of a large class of Network Design problems. Specifically, given a digraph/network $D = (V, A)$, demands at the nodes, and fixed and variable costs on the arcs, the problem is to select a set of arcs to be opened, and to find a feasible flow in the resulting network such that the sum of the fixed arc costs plus the variable flow costs is minimized.

Until recently the commercial mixed integer programming solvers just used linear programming based branch-and-bound and did not perform at all well on most instances of UFC. In the last two years these solvers have improved remarkably, and now generate cutting planes such as flow cover inequalities designed for simple fixed charge flow problems. However they still do not take full advantage of the structure of UFC.

This explains in part why much previous work has been on the development of specialized algorithms for specific variants of UFC. Chopra et al. [9] and Koch and Martin [24] have developed branch-and-cut codes for the Steiner tree problem which can be viewed as a special case of UFC in which the flow costs are zero. Several branch-and-bound algorithms have been described for the uncapacitated fixed charge transportation problem in which the underlying network is bipartite, see for example [5], [8], [36], [33]. Some production planning problems such as the single-level and (series) multi-level uncapacitated lot-sizing problem can be formulated as UFC, but here again specialized algorithms have been developed, see for example [6],[37]. Finally, for the single source UFC, Hochbaum and Segev in [19] present a Lagrangian relaxation algorithm and primal heuristics, and very recently Cruz et al. [12] report results obtained with a branch and bound algorithm also based on a Lagrangian relaxation.

The original goal of this study was to test whether the effectiveness of cutting planes in solving uncapacitated lot-sizing problems [4],[40] extended to the more general and more difficult UFC. In particular we were interested in how the prototype branch-and-cut system `bc-opt` [11] (which among others generates path inequalities for fixed charge path networks generalizing the lot-sizing inequalities) behaved on such problems. Another question was whether UFC, lying somewhere between a general mixed integer program, and highly structured problems such as the fixed charge transportation or Steiner problems, was at an appropriate level of generality for the study of cutting planes and the development of algorithms.

One result of our study is that, whereas cuts based on paths (as implemented in `bc-opt`) are fundamental for lot-sizing problems, simple "dicut" inequalities [2] and their variants are crucial for UFC, and form the basis of the branch-and-cut system `bc-nd` implemented here. Our test set contains 31 "hard" instances that are not solved by either Cplex [20], `mp-opt` [14] or `bc-opt` within 30 minutes. `bc-nd` solves 6 of these instances to optimality, and the average final duality gap for the other 25 instances is 4.4 % while it is 30.40, 23.55 and 18.31 % for the three systems just cited. Additionally, all the "medium" instances (those that are solved by exactly one of the general systems) are solved with `bc-nd`.

The outline of the article is as follows. In Section 2 we formulate the problem and give some definitions used throughout the paper. In Section 3 we describe the dicut inequality and its variants, and consider the complexity of the separation problem. Section 4 is devoted to a description of `bc-nd`. The test instances are described in Section 5, and the computational results in Section 6. Some conclusions and extensions are discussed in Section 7.

2 Problem Formulation

The single commodity uncapacitated fixed charge network flow problem (UFC) can be formulated as follows: given a directed graph $D = (V, A)$, a demand vector $b = (b_i)$ for $i \in V$, fixed and variable costs f_{ij} and c_{ij} for $(i, j) \in A$, find a set of arcs and a feasible flow in the resulting network that minimizes the total cost. This problem is NP-hard, as it generalizes the Steiner tree problem [30].

In order to describe UFC as a mixed integer program, define x_{ij} to be the flow on arc (i, j), and $y_{ij} = 1$ if the arc (i, j) is used ($x_{ij} > 0$), and $y_{ij} = 0$ otherwise. A resulting formulation is:

$$
(UFC)\begin{cases}
\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in A} f_{ij}y_{ij} & (1)\\
s.t. & \\
\sum_{j\in V_i^-} x_{ji} - \sum_{j\in V_i^+} x_{ij} = b_i \quad \forall\, i \in V & (2)\\
x_{ij} \le U\ y_{ij} \quad \forall\, (i,j) \in A & (3)\\
x_{ij} \ge 0 \quad \forall\, (i,j) \in A, \qquad y_{ij} \in \{0,1\} \quad \forall\, (i,j) \in A & (4)
\end{cases}
$$

where $V_i^+ = \{j \in V \ : \ (i, j) \in A\}$ and $V_i^- = \{j \in V \ : \ (j, i) \in A\}$, U is a large positive integer and $\sum_{i\in V} b_i = 0$. Constraints (2) are the well-known *conservation constraints*, constraints (3) are the forcing constraints that guarantee that y_{ij} takes value one whenever x_{ij} is positive. Note that it suffices to take $U = \sum_{i\in V: b_i>0} b_i$.

As additional notation we use $V_S = \{i \in V \ : \ b_i < 0\}$ to denote the set of *supply nodes*, $V_D = \{i \in V \ : \ b_i > 0\}$ the set of *demand nodes*, and $V_0 = \{i \in V \ : \ b_i = 0\}$ the set of *transshipment nodes*. Let X be the set of vectors (x, y) satisfying $(2) - (4)$. The next proposition characterizes the extreme points of $conv(X)$ (e.g. [1]).

Proposition 2.1 *Given (x,y) in X, let*

$$\begin{aligned} F(x,y) &= \{a \in A \ / \ 0 < x_a < U \ , \ y_a = 1\} \\ L(x,y) &= \{a \in A \ / \ x_a = 0 \ , \ y_a \in \{0,1\}\} \\ U(x,y) &= \{a \in A \ / \ x_a = U \ , \ y_a = 1\} \end{aligned}$$

Then (x, y) is an extreme point of $conv(X)$ if and only if the graph $D_{x,y} = (V, F(x, y))$ contains no cycles.

This characterization will be used in Section 4 to devise branching rules, a pruning criterion, and also to fix variables in the enumeration tree.

3 Valid Inequalities

In this section we describe the dicut inequality and its variants. To present examples of the different inequalities, we use the instance shown in Figure 1.

$$\begin{aligned} x_{21} - x_{14} - x_{15} &= -1 \\ -x_{21} - x_{23} &= -3 \\ x_{23} + x_{43} - x_{37} &= 0 \\ x_{14} + x_{54} + x_{74} - x_{43} - x_{46} &= 2 \\ x_{15} - x_{54} - x_{56} &= -1 \\ x_{56} + x_{46} - x_{67} &= 0 \\ x_{37} + x_{67} - x_{74} &= 3 \end{aligned}$$

$$\begin{aligned} &x_{ij} \leq 5 \ y_{ij} \\ &x_{ij} \geq 0 \\ &y_{ij} \in \{0, 1\} \end{aligned}$$

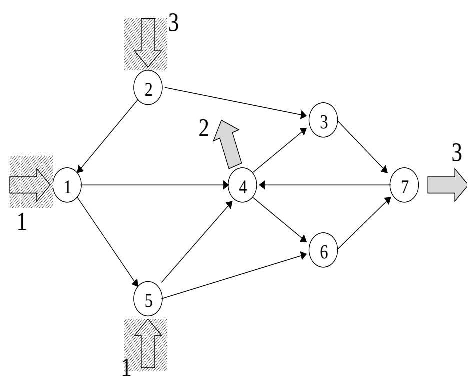

Figure 1: Example

Consider a proper subset S of nodes for which the net demand is positive, i.e. $b(S) = \sum_{i \in S} b_i > 0$. The set obtained by summing up the conservation constraints (2) over all nodes in S is called X_S and known as a single node flow problem. Mathematically it takes the form:

$$X_S = \left\{ (x,y) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} \ : \ \begin{array}{ll} \sum\limits_{(i,j) \in \delta^-(S)} x_{ij} - \sum\limits_{(i,j) \in \delta^+(S)} x_{ij} = \sum\limits_{i \in S} b_i & (5) \\ x_{ij} \leq U \ y_{ij} \quad \text{for } (i,j) \in A & (6) \\ x_{ij} \geq 0, \quad y_{ij} \in \{0,1\} \quad \text{for } (i,j) \in A & (7) \end{array} \right\}$$

where, $\delta^+(S) = \{(i, j) \in A \ : \ i \in S, j \notin S\}$ is the set of arcs leaving S and $\delta^-(S) = \{(i, j) \in A \ : \ i \notin S, j \in S\}$ is the set of arcs entering S.

In the example with $S = \{3, 5, 6, 7\}$, the constraints (5) and (6) take the form:

$$x_{23} \quad +x_{43} \quad +x_{46} \quad +x_{15} \quad -x_{54} \quad -x_{74} \quad = 2$$

$$x_{23} \leq 5y_{23}, \quad x_{43} \leq 5y_{43}, \quad x_{46} \leq 5y_{46}, \quad x_{15} \leq 5y_{15} \quad x_{54} \leq 5y_{54}, \quad x_{74} \leq 5y_{74}.$$

Because S has a positive supply of 2 units, every feasible flow must contain at least 2 units leaving S. Thus, at least one arc of $\delta^-(S) = \{(2,3), (4,3), (4,6), (1,5)\}$, the set of arcs entering S, must be open. This is expressed in the inequality:

$$y_{23} + y_{43} + y_{46} + y_{15} \geq 1.$$

We now make a more general statement.

Proposition 3.1 *For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the basic dicut inequality*

$$\sum_{(i,j) \in \delta^-(S)} y_{ij} \geq 1$$

is valid for X.

These inequalities have been used to formulate the directed Steiner tree problem [26].

Consider again the same subset $S = \{3, 5, 6, 7\}$. A relaxation of the aggregated set X_S is obtained if x_{23} and x_{15} are replaced by their upper bounds, and x_{54} and x_{74} are replaced by their lower bounds of zero. The resulting set is:

$$5y_{23} + x_{43} + x_{46} + 5y_{15} \geq 2, \quad y_{23}, y_{15} \in \{0, 1\}, \; x_{43}, x_{46} \geq 0.$$

Applying coefficient reduction [28] or the mixed integer rounding procedure [32] gives the inequality:

$$2y_{23} + x_{43} + x_{46} + 2y_{15} \geq 2$$

In general we have the following result.

Proposition 3.2 *For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the mixed dicut inequality*

$$\sum_{(i,j) \in \delta^-(S) \setminus C} x_{ij} + \sum_{(i,j) \in C} b(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subseteq \delta^-(S)$.

Further, taking $S = \{4, 5, 6, 7\}$, one such mixed dicut inequality is:

$$x_{14} + 4y_{37} + x_{15} \geq 4.$$

Now, the flow going from $V \setminus S = \{1, 2\}$ to satisfy demand in S passing through the arc $(3, 7)$ is at most 3 units, the supply of node 2. This observation allows us to tighten the coefficient associated to y_{37} giving the valid inequality:

$$x_{14} + 3y_{37} + x_{15} \geq 4.$$

In general, for $e \in C \subseteq \delta^-(S)$ let $E(S) = \{(i,j) \in A \; : \; i, j \in S\}$, $V^+_{e,S} = \{i \in S \; : \; b_i > 0$ and there exists a dipath in $G_S = (S, E(S))$ between the head node of e and the node $i\}$, $V^-_{e,S} = \{i \in V \setminus S \; : \; b_i < 0$ and there exists a dipath in $G_{V \setminus S} = (V \setminus S, E(V \setminus S))$ between the node i and the tail node of $e\}$, $\alpha_e(S) = \min\{\sum_{i \in V^+_{e,S}} b_i, \sum_{i \in V^-_{e,S}} |b_i|\}$.

Proposition 3.3 *For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the simple inflow-outflow inequality*

$$\sum_{(i,j) \in \delta^-(S) \setminus C} x_{ij} + \sum_{(i,j) \in C} \alpha_{ij}(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subset \delta^-(S)$.

This modification is important when the underlying graph is sparse. For the single item uncapacitated lot-sizing problem all the inequalities required to give a complete description of the convex hull are of this type.

With $S = \{3, 4, 6, 7\}$, we obtain the following mixed dicut inequality

$$x_{23} + x_{14} + 5y_{54} + 5y_{56} \geq 5.$$

Now suppose that the contribution of the flow in arc $(7, 4)$ to satisfy the demand $b(S)$ is measured separately. In this case, the maximum flow that can pass through the arc $(5, 6)$ using the arcs of $E(S)$ except for $(7, 4)$ to satisfy the demand $b(S)$ is 3, the demand of node 7. So the inequality

$$x_{23} + x_{14} + 5y_{54} + 3y_{56} + x_{74} \geq 5$$

is valid.

In general, given $R \subset E(S)$, define $V^R_{e,S} = \{i \in S \ : \ b_i > 0$ and there exists a dipath in $G_S = (S, E(S) \setminus R)$ between the head node of e and the node $i\}$, and $\alpha^R_e(S) = \sum_{i \in V^R_{e,S}} b_i$.

Proposition 3.4 *For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the inflow-outflow inequality*

$$\sum_{(i,j) \in (\delta^-(S) \setminus C) \cup R} x_{ij} + \sum_{(i,j) \in C} \alpha_{ij}(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subset \delta^-(S)$, $R \subseteq E(S)$.

The above inequality is a particular case of the network inequalities of Van Roy and Wolsey [39].

With $S = \{3, 5, 6, 7\}$, another possible relaxation of X_S is given by

$$5y_{23} + 5y_{43} + x_{46} + 5y_{15} \geq 2 + x_{74}, \quad x_{74} \leq 5y_{74}.$$

Letting, $\bar{x}_{74} = 5y_{74} - x_{74} \geq 0$ and $\bar{y}_{74} = 1 - y_{74}$ we get

$$5y_{23} + 5y_{43} + x_{46} + 5y_{15} + \bar{x}_{74} + 5\bar{y}_{74} \geq 7.$$

Now applying the MIR procedure, we obtain

$$2y_{23} + 2y_{43} + x_{46} + 2y_{15} + \bar{x}_{74} + 2\bar{y}_{74} \geq 4,$$

and reintroducing the original variables, we get the following valid inequality:

$$2y_{23} + 2y_{43} + x_{46} + 2y_{15} \geq 2 + (x_{74} - 3y_{74}).$$

The general expression is given in the next proposition.

Proposition 3.5 *For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the mixed dicut with outflow inequality*

$$\sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + \sum_{(i,j) \in C^-} b(S) y_{ij} \geq b(S) + \sum_{(i,j) \in C^+} \{x_{ij} - r(S) y_{ij}\}$$

is valid for X for all and $C^- \subseteq \delta^-(S)$ and $C^+ \subseteq \delta^+(S)$, with $r(S) = U - b(S)$.

Proof: The proof is a direct generalization of the procedure used in the example. Consider the following relaxation of X_S:

$$\sum_{(i,j)\in\delta^-(S)\setminus C^-} x_{ij} + U \sum_{(i,j)\in C^-} y_{ij} \geq b(S) + \sum_{(i,j)\in C^+} x_{ij}$$

plus $x_{ij} \leq U y_{ij}$ for all $(i,j) \in \delta^-(S) \cup C^+$ and all y_{ij} binary. Defining the variables $\bar{x}_{ij} = U y_{ij} - x_{ij}$, and $\bar{y}_{ij} = 1 - y_{ij}$, we can make the substitution $x_{ij} = U - U\bar{y}_{ij} - \bar{x}_{ij}$ for $(i,j) \in C^+$. Now the previous constraint can be rewritten as

$$\sum_{(i,j)\in\delta^-(S)\setminus C^-} x_{ij} + U \sum_{(i,j)\in C^-} y_{ij} + \sum_{(i,j)\in C^+} \bar{x}_{ij} + U \sum_{(i,j)\in C^+} \bar{y}_{ij} \geq b(S) + U|C^+|$$

Applying the MIR procedure, we get:

$$\sum_{(i,j)\in\delta^-(S)\setminus C^-} x_{ij} + \sum_{(i,j)\in C^+} \bar{x}_{ij} \geq b(S)\left(1 + |C^+| - \sum_{(i,j)\in C^-} y_{ij} - \sum_{(i,j)\in C^+} \bar{y}_{ij}\right),$$

which after substitution back gives the required inequality.

∎

At least two other classes of inequalities can potentially be used in solving UFC. First it is possible to mix dicut inequalities for different sets S using the mixing procedure of Günlük and Pochet [17]. A second class are the *Multi-dicut* inequalities. They were initially presented by Rardin and Wolsey in [35] for the single source case. A recent version for the multiple source case can be found in [27].

Before finally leaving the Example in Figure 1, it is also worth pointing out that all but one of the inequalities presented are facet defining. In fact, the complete description of $conv(X)$, obtained with the code Porta [10], contains 7143 facet defining inequalities of which the majority are not dicut inequalities.

3.1 Difficulty of the Separation Problem

First we formalize the separation problem for simple dicut inequalities. To find a violated simple dicut inequality, we look for a subset of nodes S such that $b(S) > 0$ and

$$\sum_{(i,j)\in\delta^-(S)} \bar{y}_{ij} < 1.$$

This can be seen as a minimum cut problem with an additional constraint to ensure that $b(S) > 0$. For $i \in V$, define variable $z_i = 1$ if i belongs to S, and $z_i = 0$ otherwise. The separation problem reduces to solving the problem:

$$\xi = \min \left\{ \sum_{(i,j)\in A} \bar{y}_{ij} z_j (1 - z_i) \;:\; \sum_{i\in V} b_i z_i > 0, z_i \in \{0,1\} \text{ for all } i \in V. \right\}$$

If $\xi < 1$, the set S defined by $\{i \in V \;:\; z_i = 1\}$ leads to a violated inequality.

By reduction from the exact partitioning problem the separation problem associated to the simple dicut inequalities can be shown to be NP-complete [31]. However for the single source problem in which $|V_S| = 1$, the imposed constraint can be dropped. The separation problem can be solved in polynomial time. It can be reduced to $|V_D|$ minimum s-t cut problems where s is the source and t varies over the set V_D.

The problem of finding a violated mixed dicut inequality can be stated as follows: given a fractional point $(\bar{x}, \bar{y})$, we look for $S \subset V$ with $b(S) > 0$ and $C \subseteq \delta^-(S)$ such that:

$$\sum_{(i,j)\in\delta^-(S)\setminus C} \bar{x}_{ij} + \sum_{(i,j)\in C} b(S)\bar{y}_{ij} < b(S).$$

For a given S, finding the most violated inequality is trivial. It suffices to set $C = \{ij \in \delta^-(S) \ : \ \bar{x}_{ij} > b(S)\bar{y}_{ij}\}$. Therefore, the principal difficulty regarding the separation of dicut inequalities is to find the right set S.

For mixed dicut inequalities, the complexity of the separation problem is still an open problem (e.g [2]) as far as we know.

The above observations led us to consider using heuristics to separate the various dicut inequalities. The separation heuristic based on searching for good candidate cut sets is presented in the next section.

4 `bc-nd` : a Branch-and-Cut System for UFC

In this section we describe the branch-and-cut system `bc-nd`. We begin with some basic implementation issues, then we present the separation heuristic, the primal heuristic, branching rules, and finally pruning and variable fixing criteria.

4.1 The Basics

Our implementation is based on the Extended Modeling and Optimisation Subroutine Library (EMOSL) from Xpress [13]. This library implements a branch and bound algorithm with a series of "entry points" that allow users to include their own routines. Using these entry points we can generate cuts and add them to the matrix, apply heuristics and develop branching rules. At the top node we have used those entry points to generate cuts, and to apply a heuristic. In the enumeration tree we have used the entry points to generate cuts, prune a node, to choose branching variables and to implement a primal heuristic. Additionally, the library has routines to access the information contained in the model file. Such information is used to determine the digraph that defines the instance being solved. The data structure used to store the graph has been borrowed from MCF [25].

Because several cuts can be generated from the same set S, we have set up a set pool in order to store the candidate sets. The set pool is a dynamic double linked list. Sets are *active* until an associated *frequency parameter* falls below a certain value, at which point the set becomes inactive. How this parameter is updated and when a set is declared inactive are described later in the Cut generation step.

4.2 Separation

The separation consists of the following steps: cut deletion, shrinking, set generation, cut generation and reoptimization. One realization of all these steps is called a `pass`. The default number of passes at the top node has been fixed at 30. Whereas in the enumeration tree it has been fixed at 5. We describe now each step of a pass.

- Cut deletion. Because the number of violated dicuts can be large, keeping all of them in the matrix during all the passes can be too expensive. Therefore, we eliminate the non binding cuts from the matrix at the beginning of each pass. Because some of the deleted cuts may be violated later, we perform cut pool separation at the beginning of the cut generation step. No cuts are deleted from the cut pool.

- Shrinking. In order to reduce the size of the graph on which we search for "interesting" subsets, the graph is shrunk based on the current linear programming solution. Specifically, whenever $\bar{y}_{ij} > 0.99$ and $\bar{x}_{ij} > 10^{-6}$, the two end nodes i, j are contracted into one super node. The demand of the new super node is the sum of the demands. Only nodes are contracted, so the resulting reduced graph typically contains multiple arcs and loops. This shrinking procedure is heuristic. Arcs with $\bar{y}_{ij} > 0$ but $\bar{x}_{ij} = 0$ are not used for shrinking because the addition of the dicut inequalities often forces $y_{ij} > 0$ artificially in the linear programming relaxation, even when there is never flow in the corresponding arc.

- Subset generation. The dicut inequalities are based on "node subsets". Therefore finding good subsets can reduce the number of iterations of cut generation, and also lead to a better top node reformulation. In our implementation, three greedy procedures are used to generate subsets for a given fractional solution $(\bar{x}, \bar{y})$. They differ in the choice of the initial node and in the quantity used to enlarge the current set. Below, we describe these choices:

  - Initialize $S = \{i_0\}$ for $i_0 \in V_S \cup V_D$. Enlarge S using $\max\{\bar{y}_{ij} \; : \; (i,j) \in \delta^-(S) \; , \; \bar{y}_{ij} \in (0,1)\}$.
  - Initialize $S = \{i_0\}$ for $i_0 \in V_S \cup V_D$. Enlarge S using $\max\{\bar{x}_{ij} - b(S)\bar{y}_{ij} \; : \; (i,j) \in \delta^-(S) \; , \; \bar{y}_{ij} \in (0,1)\}$.
  - Given an arc $a = (i_0, j_0)$ such that $\bar{y}_{i_0 j_0}$ is fractional, two candidate sets S are built. In the first case, we start with a set S such that $j_0 \in S$, $b(S) > 0$ and $a \in \delta^-(S)$, and we expand S using the criterion $\max\{\bar{x}_{ij} - b(S)\bar{y}_{ij} \; : \; (i,j) \in \delta^-(S) \; , \; \bar{y}_{ij} \in (0,1), b(S) + d_i > 0\}$. In the second case, we start with a set $\tilde{S} = V \setminus S$ such that $i_0 \in V \setminus S$, $b(V \setminus S) < 0$, and $a \in \delta^+(V \setminus S)$ and we expand $V \setminus S$ using the criterion $\max\{\bar{x}_{ij} - |b(V \setminus S)|\bar{y}_{ij} \; : \; (i,j) \in \delta^+(V \setminus S) \; , \; \bar{y}_{ij} \in (0,1), b(V \setminus S) + d_j < 0\}$. The procedure stops either when the maximum number of nodes allowed is reached or when a violated inequality can be generated.

  The three procedures are called sequentially. The sets generated are stored in the set pool with the status "active", and the frequency parameter initialized at zero.

- Cut generation. We start by performing cut pool separation. Then for each active set S, three dicut inequalities can be generated:

  a) Simple dicut. if $\sum_{a \in \delta^-(S)} \bar{y}_a < 1 - 0.015$, then the inequality

  $$\sum_{a \in \delta^-(S)} y_a \geq 1$$

  is added to the cut pool.

  b) Simple inflow-outflow inequality. define $C = \{a \in \delta^-(S) \; : \; \bar{x}_a > \alpha_a(S)\bar{y}_a\}$ and if $\sum_{a \in \delta^-(S) \setminus C} \bar{x}_a + \sum_{a \in C} \alpha_a(S)\bar{y}_a < b(S) - 0.015$, then the inequality

  $$\sum_{a \in \delta^-(S) \setminus C} x_a + \sum_{a \in C} \alpha_a(S) y_a \geq b(S)$$

  is added to the cut pool. The coefficient $\alpha_{ij}(S)$ is computed using breadth-first search to determine the sets $V^+_{ij,S}$ and $V^-_{ij,S}$ defined in proposition 3.3. Then, $\alpha_{ij}(S) = \min\left\{b(S), \sum_{k \in V^+_{ij,S}} b_k, \sum_{k \in V^-_{ij,S}} |b_k|\right\}$.

c) Mixed dicut with outflow. Define $C^- = \{a \in \delta^-(S) \ : \ \bar{x}_a > b(S)\bar{y}_a\}$, $r(S) = U - b(S)$, and $C^+ = \{a \in \delta^+(S) \ : \ \bar{x}_a > r(S)\bar{y}_a\}$ and if $\sum_{a\in\delta^-(S)\setminus C^-} \bar{x}_a + \sum_{a\in C^-} b(S)\bar{y}_a < b(S) + \sum_{a\in C^+}\{\bar{x}_a - r(S)\bar{y}_a\} - 0.015$, then the inequality

$$\sum_{a\in\delta^-(S)\setminus C^-} x_a + \sum_{a\in C^-} b(S)y_a \geq b(S) + \sum_{a\in C^+}\{x_a - r(S)y_a\}$$

is added to the cut pool.

If a cut is added to the cut pool the frequency parameter is increased by 1. Otherwise, decrease the frequency parameter by 1 and look at the next set.

Whenever the frequency parameter of a set is less than -3, the set is declared inactive. Once, all the active sets have been visited, the cuts generated are added into the matrix.

- Reoptimization. If violated inequalities have been found, the linear program is reoptimized. If the number of passes is less than the maximum, go to the next pass. Otherwise go to the enumeration phase.

4.3 Primal Heuristics

Given the structure of UFC, we have examined the possibility of developing effective primal heuristics to find good feasible solutions rapidly. After several attempts, motivated by [18], [12], [38], [16] , two heuristics have been retained:

- Slope scaling [22]. This algorithm is based on the idea that there exists a linear program,

$$(P(\bar{c}))\min\left\{\sum_{(i,j)\in A}\bar{c}_{ij}x_{ij} \ : \ \sum_{j\in V_i^-} x_{ji} - \sum_{j\in V_i^+} x_{ij} = b_i \ \forall\, i\in V,\ 0\leq x_{ij}\leq U \ \ \forall\,(i,j)\in A\right\}$$

that has the same optimal solution as the original mixed integer problem, or in other words, that there exists $\hat{c}$ such that $v(UFC) = v(P(\hat{c}))$.

To find such a $\hat{c}$, a sequence $\{\bar{c}^k\}_{k=1}^K$ of slopes are constructed, such that $\bar{c}^K$ is not far from $\hat{c}$. Let x^k be an optimal solution of $P(\bar{c}^k)$. The slope at iteration $k+1$ is computed as follows:

$$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij} + \frac{f_{ij}}{x_{ij}^k} & \text{if } x_{ij}^k > 0 \\ g(x^k, x^{k-1}, ..., x^1) & \text{otherwise} \end{cases}$$

where, c and f are the original variable and fixed costs, and $g(\cdot)$ is a function that depends on the solutions of the previous iterations. In our implementation, the first objective function is computed from the solution obtained after the cut generation phase. Indeed, let $(\bar{x}, \bar{y})$ be such a solution. The first objective function is given by:

$$\bar{c}_{ij}^{1} = \begin{cases} c_{ij} + \frac{f_{ij}}{\bar{x}_{ij}} & \text{if } \bar{x}_{ij} > 0 \\ c_{ij} + \frac{f_{ij}}{U} & \text{otherwise} \end{cases}$$

Then, at iteration $k+1$ we define the cost function as follows :

$$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij} + \frac{f_{ij}}{x_{ij}^k} & \text{if } x_{ij}^k > 0 \\ \lambda\bar{c}_{ij}^{r} + (1-\lambda)(c_{ij} + \frac{f_{ij}}{U}) & \text{otherwise} \end{cases}$$

where $\lambda \in (0,1)$ and $r \in \{1,..,k-1\}$ is the last iteration in which $x_{ij}^r > 0$ and the cost assigned was $\bar{c}_{ij}^r$.

If $x^{k+1} = x^k$, then stop. Otherwise, go to the next iteration. If the maximum number of iterations is attained, we stop. If no solution has been found, we apply a rounding heuristic. Let $A' = \{a \in A : \bar{y}_a > 0\}$. Find a feasible flow x^* using just the arcs of A'. Set $y_a^* = 1$ if $x_a^* > 0$. (x^*, y^*) is the heuristic solution.

- Min cost Flow. Here we solve a minimum cost flow problem on the graph defined by the open arcs of the current fractional solution. In other words, given a fractional solution $(\bar{x}, \bar{y})$, we define the graph $G' = (V, A')$ where $A' = \{(i,j) \in A \ : \ \bar{y}_{ij} > 0\}$. The objective function is defined as:

$$\bar{c}_{ij}(\bar{x}) = \begin{cases} c_{ij} + \frac{f_{ij}}{\bar{x}_{ij}} & \text{if } \bar{x}_{ij} > 0 \\ \\ c_{ij} + \frac{f_{ij}}{u_{ij}} & \text{otherwise} \end{cases} \qquad \text{for all} \quad (i,j) \in A'$$

  The solution to this problem is obtained with the network simplex implementation MCF [25]. The basis is stored and reused as initial basis in the next call.

The slope scaling procedure is called at the top node, whereas the Min cost flow heuristic is called at 10 consecutive nodes every 100 nodes in the branch-and-bound tree.

4.4 Branching Rules

Whereas many specialized branch-and-cut codes use simple variable branching rules such as most fractional, most costly, etc, commercial MIP systems use pseudo-costs based on dual variable estimates. As UFC falls somewhat between the general and the special purpose, we have attempted to compare some of the simple branching rules.

Below, we briefly present the variable and constraint branching rules that we have tried.

- Variable Branching. The rules we have studied are :
  - Most fractional. Branch on the variable that maximizes $w_a = \max\{\ \bar{y}_a\ ,\ \ 1-\bar{y}_a\}$ over the set of arcs $a \in A$ such that $\bar{y}_a \in (0,1)$.
  - Least fractional. This criteria is similar to the previous one. Branch on the variable that minimizes $w_a = \max\{\bar{y}_a, 1-\bar{y}_a\}$.
  - Maximum fixed charge. Branch on the variable for which the fixed cost f_a is maximum among those with $\bar{y}_a$ fractional.
  - Maximum remaining fixed charge. Branch on the variable that maximizes $w_a = (1-\bar{y}_a)\cdot f_a$ among those with $\bar{y}_a$ fractional.
  - 2-strong branching (2-st). Select the two arcs, a^1, a^2 with the biggest and second biggest fixed charge. Then compute (using five iterations of dual simplex) a lower bound of their possible offsprings, namely, z_0^1, z_1^1 for a^1 and z_0^2, z_1^2 for a^2 when the variable is fixed to zero and one respectively. Then, compute $w^1 = \min\{z_0^1, z_1^1\}$ and $w^2 = \min\{z_0^2, z_1^2\}$, and branch on the variable that maximizes w^i.
- Constraint Branching. Branching can also be based on linear inequalities. Here we consider the possibility of branching on subtour constraints, motivated by the fact that optimal solutions of uncapacitated problems do not contain cycles (Proposition 2.1). These inequalities have the following form:

$$\sum_{a \in E(S)} y_a \le |S| - 1 \quad \forall S \subset V.$$

Given the solution at the current node, determine a fractional subtour, i.e. a set S for which there exists at least one arc $a \in E(S)$ with $\bar{y}_a$ fractional. Then

- If the subtour is encountered for the first time, compute its value, i.e. $l(S) = \sum_{a \in E(S)} \bar{y}_a$. If this value is less than $|S| - 1$ and greater than or equal to one, then the branching constraints chosen are $\sum_{a \in E(S)} y_a \leq \lfloor l(S) \rfloor$ and $\sum_{a \in E(S)} y_a \geq \lceil l(S) \rceil$. The node to be solved in the next iteration is one of the successors of the current node.
- Otherwise, if the subtour has already been used, select the most fractional variable in $E(S)$ to branch on.

4.5 Pruning Criteria and Variable Fixing

When the variable and fixed costs are non negative, there exists an optimal solution that is cycle-free (Proposition 2.1). This allow us to prune some nodes of the enumeration tree, and also provides a test allowing us to fix an arc variable y_a to zero, if arc a plus the arcs fixed to one form a cycle. Both these tests are carried out before solving the linear program at each node of the enumeration tree.

5 A set of Test Problems

Several problem classes have been used to test our implementation. Here, we describe the test instances and how they have been generated. Some of the problems are from the literature and the others have been randomly generated. Problems are classified according to the number of source nodes and the structure of the underlying graph. The different classes of problems are shown below.

$$\text{single-source} \begin{cases} \text{Grids} \\ K_n \\ K_n + 1 \\ \text{Steiner} \\ \text{Multi-segment} \\ \text{Multi-level LS} \end{cases} \qquad \text{multi-source} \begin{cases} \text{Grids} \\ \text{Series-parallel} \\ K_n \\ \text{Planar} \\ \text{Random} \end{cases}$$

In all cases, the formulation we have used is based on (1)-(4) from Section 2. For the single source cases, the structure of the optimal extreme solutions says that for every node, only one inflow are can have a positive flow. This gives us the following additional tree constraint: $\sum_{j \in V_i^-} y_{ji} \leq 1$ for all $i \in V$.

Now we explain how the different instances have been created, or indicate their origin.

- Grids. Here the graph is a two dimensional rectangular grid. The parameters used to generate a grid problem are the width and the height (in number of nodes), the total demand, the number of source and demand nodes, and bounds on the costs. Demand and supply nodes are selected randomly as well as the fraction of the demand/supply assigned to a node. The random number generator is that of NETGEN [23]. The variable and fixed costs are uniformly generated over the specified interval using the C/C++ random number generator.

- Complete, K_n. The graph is complete. The number of nodes, the bounds on the costs and the total demand are the parameters needed to specify an instance. The source and demand nodes, the value of the demand and supply, and the costs are chosen in the same way as for grid graphs.

- Random. Here the number of nodes, arcs, source nodes, demand nodes, total demand, and costs intervals are specified. The procedure iteratively selects an arc that has not already been chosen and then assigns costs. Finally, we define the source and demand nodes using the same strategy as before.

- Planar. To generate planar graphs, we have used the planar graph generator included in LEDA [29]. The position and the number of the source and demand nodes are determined with the uniform random number generator from LEDA. The cost function is computed as for Grids.

- Series-Parallel. Here we also use the series-parallel graph generator from LEDA. The rest of the parameters, number of source and demand nodes, total demand and costs are calculated as for planar graphs.

- Hochbaum, $K_n + 1$. These are single source instances taken from [19]. To clarify the description, suppose node 1 is the source node and $\{2, .., n\}$ are the other nodes. The set of arcs consist of $(1, j)$ for every $j > 1$ and (i, j) for every $i, j > 1$, $i \neq j$. So the nodes $\{2, .., n\}$ induce a complete directed graph. The demand of each node is uniformly selected from $\{0, 1, 2, .., 10\}$. The fixed cost is a multiple of the variable cost. We use the same factor 50 as in the paper. To generate the variable costs, we randomly select n points in the plane that are associated to each node. The variable cost of an arc is then calculated as the square of the distance between the points associated to each end node, in other words, if (w_i, h_i) and (w_j, h_j) are the coordinates associated to the nodes i and j the cost of the arc (i, j) is given by $c_{ij} = (w_i - w_j)^2 + (h_i - h_j)^2$.

- Steiner. These undirected instances have been obtained from the Steiner problem Library at ZIB `ftp://ftp.zib.de/pub/Packages/mp-testdata/index.html`. A complete description of all these problems can be found in [24]. In our tests we have selected a small subset of the instances:

  - Beasley. Problems 1,2 and 3 of series C described in [21]. Instances from series B were not considered as they are too easy.
  - X: Instances brasil and berlin. These have complete graphs and Euclidean weights.
  - Mc7, Mc8 and Mc11. These instance are described in [24].

  The formulation we have used is again (1)-(4) from Section 2. An arbitrary node is defined as root node with supply equal to the number of terminal nodes minus one. Each terminal node has a demand of one unit. Undirected edges have been modeled with two directed arcs.

- Multi-Segment. These are single source problems with a concave piecewise linear objective function, see [15] for a survey on this kind of model. One of the instances `beavma`, that comes from a practical harvesting problem in Chile, involves a planar graph and the objective function has at most two segments per arc. The others are randomly generated instances. `mtest4ma` has at most 4 segments. `g150x1100`, `k15x420` and `p50x576` have 2 segments. The other three instances have three segments. In order to model multiple segments, we use a directed multigraph. The formulation used is given by: $x_{ij} = \sum_k x_{ij}^k$, $x_{ij}^k \leq u_{ij}^k \; y_{ij}^k$ and $y_{ij} = \sum_k y_{ij}^k \leq 1$.

  In order words, each arc with more than one segment is repeated as many times as the number of segments in the objective function. The constraint that at most one of the arcs can have a positive flow is then added.

- fixnet6 is a single source problem from the MIPLIB [7], .

- Multi-Level Lot-sizing. The production in series lot-sizing problem [34] can be formulated as a single source UFC problem. The parameters required to generate these instances are the number of periods, the number of levels, the maximum demand and the intervals for the costs. The fixed and variable costs are uniformly selected from the given interval, except that the fixed costs are zero on the stock arcs. The demand is uniformly selected from zero to the maximum demand. The supply is the negative sum of all the demands. Note that this problem is polynomialy solvable by dynamic programming. However though many strong valid inequalities are known that generalize single-level inequalities, a complete description of the convex hull is not known.

An instance is named according to the graph type and its size or origin. In the first case, a typical instance name is: `tnxm` where `t` is the type of the graph, `n` is the number of nodes and `m` is the number of arcs. Graph types are: `g` grid, `p` planar, `k` complete, `sp` series-parallel, `r` random, `l` multi-level lot-sizing, `h` hochbaum. Instances coming from the literature keep their original name.

Our test set consists of 38 multi-source instances and 45 single source instances. The complete list and some of the principal characteristics of the instances are given in Annex A.

6 Computational Experiments

The computational results reported below can be viewed as an an attempt to answer the following questions:
Q1. How well do standard commercial mixed integer programming systems perform on this class of problems ?
Q2. What is the effect of tightening the formulation of UFC with dicut inequalities and their variants?
Q3. Can the specific structure of UFC be used to improve features of the branch-and-cut process such as primal heuristics, branching strategies, etc ?
Q4. Can anything be said about the complexity of different instances of UFC as a function of the graph structure, the number of source nodes, or the cost structure?
Q5. For specific classes of UFC, how effective is the general approach of reformulating with dicuts as compared with algorithms developed for the specific problem class?

Specifically in subsection 6.1 we report on the behavior of three MIP systems on our test set, and use the results to obtain an initial classification of the difficulty of the 83 test instances. In Subsection 6.2 we first present the results of the preliminary tests carried out to define a default strategy for `bc-nd`. At the end of this section the results on the complete test set are reported. Then, in Subsection 6.3 we rerun the three MIP systems after adding dicut inequalities at the top node. In Section 6.4 we look at the effect of the variations in some of the parameters, such as the graph structure, the number of source nodes, and the cost ratio. Additionally, comments on specialized codes for the different special cases are presented at the end of this section.

6.1 Instance Classification

The three MIP systems used to solve the test instances were Cplex 6.6 [20] on a Sun Ultra 60, 250 Mhz, 256 MB RAM running Sun Solaris 2.6. `mp-opt` v11.50o [14] and `bc-opt` [11] on a Pentium II, 400 Mhz, 128 MB RAM running Windows NT 4.0. All the numerical tolerances have been fixed to 10^{-6}. The maximum time per instance has been fixed to be 1800 CPU seconds.

Based on the results obtained, we have classified the test instances into three classes `Easy`, `Medium` and `Hard`. An instance is `Easy` when at least two of the three systems solve it to optimality. It is `Medium` when only one of the system solves it to optimality and `Hard` when no system solves it. Among the 83 instances 43 are `Easy`, 9 are `Medium` and 31 are `Hard`. In 6 hard instances the best integer solution has been found, but optimality has not been proved. Table 1 summarizes the results for each system. The first line, # solved, is the number of instances solved to optimality, the second <gap> is the average gap of non solved problems, where gap = $100(BestIP - BestBound)/(BestIP)$, and the third $<\Delta$LP> is the average lower bound improvement using the cuts of the system, where ΔLP = $100(XLP-LP)/(BestIP-LP)$. Here LP is the value of the linear relaxation, XLP is the value of the LP relaxation after the addition of systems cuts at the top node, $BestBound$ is the value of the lower bound at the end of the enumeration and $BestIP$ is the value of the best integer solution found.

	`Hard` (31 Ins.)			`Medium` (9 Ins.)			`Easy` (43 Ins.)		
param	Cplex	`bc-opt`	`mp-opt`	Cplex	`bc-opt`	`mp-opt`	Cplex	`bc-opt`	`mp-opt`
# solved	0	0	0	1	2	6	41	37	43
$< gap >$	28.23	21.41	15.74	16.46	2.84	9.49	0.05	0.17	0.01
$< \Delta LP >$	50.97	68.51	75.75	36.42	77.26	79.11	60.87	87.15	94.49

Table 1: Instance classification: summary results

6.2 Solving UFC with `bc-nd`

The first step when using a branch-and-cut system is to define the default strategy to be used. In this case, we had to decide the cut generation strategy, the branching strategy and the option of if and when to use the primal heuristic. The results of these preliminary tests are presented in the next subsection. `bc-nd` is running on a Pentium II, 400 Mhz, 128 MB RAM under Windows NT 4.0.

6.2.1 Choosing a default strategy

The first step is the cut generation strategy, namely which inequalities to use, how to separate, which cuts to keep, and when and how often to separate.

To decide which variants of the dicut inequality to include in the final prototype, our first implementation included simple dicut, mixed dicut, simple inflow-outflow and the mixed dicut with outflow inequalities. Evaluating each of them separately on the test set, the best results were obtained with just the simple dicut and the simple inflow-outflow inequalities.

In choosing which cuts to keep, the first unsurprising observation is that simple dicut inequalities involving just the 0-1 arc variables typically produce a much more significant increase in the linear programming lower bound than the various mixed dicut inequalities which include the flow variables. The second is that simultaneously adding simple and mixed dicut inequalities for the same node set seems to lead to LP degeneracy, and increases running times. So the default is for each active set to add a violated simple dicut inequality if possible, and, if not, to try to add just a simple inflow-outflow (mixed dicut) inequality.

How often to separate, when to delete cuts, and when to make cut sets inactive was determined in part by testing and in part by earlier experience with `bc-opt`. The final strategy selected is as follows: 30 passes are made, non-binding cuts are deleted at the beginning of each pass, cut pool separation is performed, and a threshold of -3 is used to discard node sets. This strategy is based on a trade-off between the quality of the resulting lower bound and the execution time, in that the LP relaxation is not too difficult to solve and the memory requirements are reasonable.

The second part concerns the branching rules. The rules tested are least fractional, most fractional, maximum fixed charge, and strong branching with a candidate list of two elements. 13 instances were selected for this experiment, 3 of them Steiner tree instances and 10 multi-source instances. The latter set contains 4 grid graphs, 2 complete graphs, 2 planar graphs and 2 random graphs. The results are summarized in Table 2. The first column gives the rule, the second the number of instances solved within 1800 secs, the third the average number of nodes evaluated and, the fourth gives the average duality gap at the end of the enumeration for the unsolved problems.

Rule	Solved	avg # Nodes	<gap>
least frac	3	11075	1.87
most frac	1	13838	2.55
max f_a	7	7715	1.22
max 2-st	2	3137	1.80
subtour	3	7866	4.35
xpress	7	17253	1.94

Table 2: branching rules

From Table 2, we conclude that, given the time limit, the best strategy is to use the branching selection rule provided by the Xpress library as the number of nodes explored is much larger than for the other strategies, and the number of instances solved to optimality is the greatest. The maximum fixed charge rule seems to be the best, but we finally selected library branching because it was more robust. In addition we chose best bound node selection, in the belief that it is effective when the duality gaps after adding cuts are very small

The third part of this experiment is related to the effects produced by the min cost flow heuristic within the enumeration phase of the algorithm. First we select the instances that need enumeration to be solved, in other words, those that are not solved at the top node by `bc-nd`. That gives 59 instances. Then we use `bc-nd` without and with the heuristic. The rest of the parameters are the same in both runs. The results are summarized in Table 3. The first column is the class, the second is the number of instances used within the class, the next four columns summarize the results without using the heuristic, and the last four the results with the heuristic. The columns # sol and <gap> are defined as before. sol^0 is the average number of nodes for the instances that are solved. unsol is the average number of nodes for the instances that cannot be solved either with or without the heuristic. * appearing as a superscript of <gap> means no feasible solution has been found for one instance.

The results reported in Table 3 showed that 31 instances are solved with the primal heuristic and 27 instances are solved without the primal heuristic. For the instances solved to optimality by both versions, the number of nodes needed to prove optimality is significantly smaller when using the heuristic. Also, for the instances that cannot be solved by either version, the final gap is smaller when using the heuristic. So the default choice is to use the heuristic at a frequency described in Subsection 4.3

		Without Heuristic				With Heuristic			
				<nodes>				<nodes>	
class	# ins.	# sol.	<gap>	sol^0	unsol	# sol.	<gap>	sol^0	unsol
grid	8	2	7.0*	621	8167	4	5.0	423	1399
K_n	6	3	12.0	455	14342	4	3.8	182	37913
plan	10	1	9.0	53	4660	1	4.1	17	7138
rand	5	1	5.0	5661	10200	2	2.3	923	17849
s-p	4	4	0	2073	–	4	0	6	–
stein	8	2	3.0	69	2120	2	2.3	16	986
m-s	5	3	5.0	35	1574	3	0.6	19	473
grid	3	3	0	18	–	3	0	9	–
K_n+1	3	2	0.3	38	1009	2	0.3	21	966
K_n	4	4	0	28	–	3	0	18	–
l-s	1	1	0	3	–	1	0	23	–
plan	1	1	0	6	–	1	0	0	–
Total	57	27				31			

Table 3: with heuristic v/s without heuristic

6.2.2 Using the default strategy: results

Table 4 summarizes the results obtained by `bc-nd` with the default strategy for the set of test instances. The maximum limit time and the tolerances are fixed as before. The measures reported here are the same as in Table 1 with in addition <gap^0 >, the average duality gap at the beginning of the enumeration, where $\text{gap}^0 = 100(firstIP - LP)/firstIP$.

param	`Hard` (31 Ins.)	`Medium` (9 Ins.)	`Easy` (43 Ins.)
# solved	6	8	42
<gap>	4.98	0.01	0.01
<gap^0 >	7.70	0.16	1.17
< ΔLP>	91.83	98.95	97.54

Table 4: `bc-nd` summary results

Comparing this table with Table 1, we observe that for the hard instances the duality gap after 30 minutes is dramatically reduced. Moreover, the medium instances can be solved within the limit time. The average value of ΔLP, the improvement obtained by reformulating the problem using the dicuts, is 95.9 %. In other words, a large part of the gap is directly closed by the dicut inequalities. In fact 25 instances are solved without enumeration. Note that one medium and one easy instance are not solved to optimality. However both have a duality gap of almost zero, and the optimal solution has been found. It is also worth pointing out that the average duality gap at the top node for the 31 `Hard` instances is 7.7%. Table 5 shows <gap^0 > for each class of graph.

Multi-source					Single-source						
grid	K_n	plan	rand	s-p	stein	m-s	grid	K_n+1	K_n	l-s	plan
7.00	10.10	5.80	7.60	0.30	12.50	1.10	1.40	0	0.60	0	0

Table 5: Initial duality gap

Detailed tables containing the solutions of all the instances used in the experiments can be found in Annex B.

6.3 Effectiveness of Dicut Inequalities

Given the classification of Subsection 6.1, we would like to study how much the dicut inequalities alone contribute to the overall improvement. To that aim, the instances are first reformulated using `bc-nd` and then given to the three systems, Cplex 6.6, `mp-opt` 11.50o and `bc-opt` under the same conditions as in Subsection 6.1, namely, time limit 1800 CPU seconds, all tolerances fixed to 10^-6, default system strategy for the cut generation and the enumeration. The reformulation of each instance is obtained using the default strategy of `bc-nd`. The results are summarized in Table 6. The measures reported are # solved the number of instance solved, $<\text{gap}^i>$ and $<\text{gap}^r>$ are the average gap of the non-solved instances using the original formulation and the improved formulation. $<\text{Nodes}^i>$ and $<\text{Nodes}^r>$ are the average number of nodes used to solve the original and the improved formulation. Finally, $<\text{T}^i>$ and $<\text{T}^r>$ are the average time needed to solve the original and the improved formulation.

	`Hard` (31 Ins.)			`Medium` (9 Ins.)			`Easy` (43 Ins.)		
param	Cplex	`bc-opt`	`mp-opt`	Cplex	`bc-opt`	`mp-opt`	Cplex	`bc-opt`	`mp-opt`
# solved	12	9	11	9	8	8	43	43	43
$<\text{gap}^i>$	30.07	20.94	14.55	-	-	-	-	-	-
$<\text{gap}^r>$	6.96	10.23	7.68	0.00	0.00	$1e^{-3}$	0.0	0.0	0.0
$<\text{Nodes}^i>$	-	-	-	427178	47737	33476	50474	6346	9640
$<\text{Nodes}^r>$	-	-	-	15	21	3	6	26	33
$<\text{T}^i>$	-	-	-	1800	764	582	184	215	81
$<\text{T}^r>$	-	-	-	1	3	5	6	26	33

Table 6: reformulated instances: summary of results

It turns out that 5 `Hard` instances are solved by exactly one system, and 9 `Hard` instances by at least 2 systems. Also 8 `Medium` instances are now solved by all three systems. Detailed results for the hard instances can be found in Table 11 in Annex B.

From the above table we observe that all the `Easy` and `Medium` instances can be solved to optimality with the three systems. Moreover, compared with Table 1 the number of nodes and the time required are reduced. In the case of `Hard` instances 14 of the 31 instances are now solved. For the other 17 instances, the final duality gap is considerably reduced.

6.4 Structure, and Comparison with Specialized Codes

In this subsection we look at the subclasses of UFC appearing in our test set. In Table 7 we consider their difficulty based on the classification of Subsection 6.1. The first four columns give the results for the single-source instances and the last four columns for the multi-source instances. In each case, the first three columns give the number of `Hard`, `Medium` and `Easy` instances in the class, and the last one the average improvement obtained with the dicut inequalities at the top node using `bc-nd`.

We see that of the single-source instances 9 are `Hard`, 6 `Medium` and 30 `Easy`, while of the multi-source instances 22 are `Hard`, 3 `Medium` and 13 `Easy`. Below we analyze the results by class of instance.

Single-source instances. The average gap reduction for these instances is 99.07 %, meaning that dicut inequalities are effective when solving single source instances. We now discuss the results obtained for each specific subclass of UFC

	Single-source				Multi-source			
	`Hard`	`Medium`	`Easy`	ΔLP	`Hard`	`Medium`	`Easy`	ΔLP
grid	0	3	3	99.97	6	0	2	90.92
K_n	0	0	5	99.84	3	0	3	89.38
plan	0	0	5	99.99	9	0	1	92.47
rand	–	–	–	–	3	0	2	89.30
s-p	–	–	–	–	1	3	5	99.10
stein	7	1	0	94.83	–	–	–	–
m-s	2	0	6	98.91	–	–	–	–
K_n+1	0	3	5	99.96	–	–	–	–
l-s	0	0	4	99.99	–	–	–	–

Table 7: complexity depending on problem subclass

- Steiner.

  These are the only single source instances for which `bc-nd` performs poorly in that the gap reduction obtained is only 94% as opposed to 99% on the other instances. One possible reason is that the UFC formulation (1)-(4) is unsuitable for the Steiner problem, and in addition that the number of flow and set-up variables is doubled so as to produce a directed network flow problem. However compared to the three general systems, `bc-nd` is much more effective. In Table 9, we see that on the 7 hard instances, the final duality gap with `bc-nd` is 2.54%, while for Cplex, `bc-opt` and `mp-opt`, it is 49.21%, 43.14% and 24.94% respectively. However if the dicut-tightened formulation is given to the three systems, 6 of these 7 instances are solved at least once.

  This is one of the most studied special cases of UFC. Recently Koch and Martin [24] have developed a successful branch-and-cut code for the undirected Steiner tree problem. The Steiner instances in the test set are all solved to optimality without branching by their specialized code within 20 seconds. However as the authors point out, these remarkable solution times are probably due in part to the specialized preprocessing that leads to significant reductions in the size of the network. For example *brasil* initially has 58 nodes and 1653 edges, whereas after preprocessing it has 39 nodes and 113 edges, and the number of terminal nodes is reduced from 25 to 10. Similar reductions are obtained for the other instances.

- $K_n + 1$.

  Among the 8 instances in the test set, `bc-nd` solves 6 without enumeration, and the gap is less that 0.5 % for the other two instances. The main difficulty encountered is that the number of variables involved in each dicut is fairly large, and thus the LP relaxation becomes difficult to solve. In fact the LP relaxation of these instances is quite degenerate even without adding the dicuts. These difficulties are only significant for instances with more than 50 nodes.

  In Hochbaum and Segev [19] where these instances were introduced, two Lagrangian relaxations were studied. The first is obtained by relaxing the forcing constraints (3) and the second by relaxing the flow balance constraints (2). They also incorporate three primal heuristics into their algorithm. They report a final average gap (defined as in Section 6.1) of 3 % with CPU times going from 10 to 80 seconds. The average gap when the first primal solution is found is around 8%.

  Additionally, they define a measure of the complexity of an instance depending on the cost function. The parameter is defined as the ratio between the maximum demand and twice the ratio between the fixed and variable cost. One of their empirical conclusions is that when the parameter lies in the interval [0.05,0.1] the instances seems to be hard. For the instances in our test set, the value of this parameter is 0.1.

- Lot-sizing.

  All these instances turn out to be easy, with `mp-opt` being fastest. We observe that the strengthened lower bound XLP obtained with dicuts using `bc-nd` is better than that obtained with the path inequalities from `bc-opt`. One possible reason is that for uncapacitated lot-sizing problems, appropriately chosen inflow-outflow inequalities resemble path inequalities and give a complete description of the convex hull of solutions in the single item case, and provide strong valid inequalities for multilevel problems.

- Multi-segment.

  `bc-nd` seems to be highly effective on this class. For example the two instances beavma and mtest4ma are both solved without enumeration. Neither could be solved with the versions of Cplex and `mp-opt` available two years ago, though `bc-opt` was able to solve both instances making extensive use of the path inequalities. Now `mp-opt` also solves them both at the top node, and even more rapidly than `bc-nd`.

- Grids, planar and K_n.

  8 of the 16 instances are solved without enumeration, and for the other instances the average gap reduction at the top node is 99.93 %. For identical graph and cost structure, the single-source instances appear to be much easier than in the multi-source case. For such instances, no specialized code is known.

- Other single source instances.

  During the development of `bc-nd`, we received a new set of 36 instances used by Cruz et al. [12] who implemented a Lagrangian relaxation to solve the single source case of UFC. The instances are randomly generated in the plane with 16 to 32 nodes and 30 to 248 arcs. The objective function is defined using the Euclidean distance between node i and j, such that the ratio between the fixed and variable cost is fixed. The values of that ratio are 1/10, 1, and 10/1. Each instance is solved by `bc-nd` within 2 seconds, several of them without enumeration.

Multi-source instances. Though 23 of the 38 instances can be solved to optimality by `bc-nd`, these instances are considerably more difficult to solve than the single source instances. In particular, the average gap reduction is only 90 %. The exception are the instances defined on series-parallel graphs on which both `bc-nd` and `mp-opt` perform well.

Unfortunately, we do not know any other code developed for this type of model.

6.4.1 Importance of Fixed/Variable cost ratio

In this experiment we took a subset of 27 instances of different kinds from our list of instances and solved them with `bc-nd` (default strategy) for four ratios: 10, 100 , 1000 and ∞ (that means, $f_{ij} = 1, c_{ij} = 0$). A summary of the results is presented in Table 8 that is organized as follows: the first two columns give the class and the number of instances used in the experiment. The next four columns give in the first line # solved, and in the second the <gap>, both defined as before. The last four columns give the average value of ΔLP in the first line, and the average solution time of the solved instances in the second line. The single source instances were put all together in one line S-s, due to the homogeneity of the results obtained. * appearing as superscript of <gap> means one of the instances was not solved due to a numerical problem.

		# solved				$<\Delta \mathrm{LP}>$			
		$<gap>$				$<\mathrm{T}>$			
	# ins	10	10^2	10^3	∞	10	10^2	10^3	∞
S-s	9	8	8	8	7	99.43	88.9	91.54	97.70
		0^*	0.12	0.73	2.1	13.7	277.1	531	873.7
grid	5	4	3	3	0	99.45	96.40	95.91	86.62
		5^{-2}	1.5	5.0	9.0	36.5	258.6	50	–
plan	5	5	4	3	0	99.95	99.17	99.21	85.96
		0	0.2	1.0	9.0	100.2	514.5	6	–
rand	5	4	4	4	0	97.59	97.16	95.99	75.89
		0.015	7.3	9.0	19.0	24.5	43.2	235	–
s-p	2	2	2	2	2	100	100	99.98	99.54
		0	0	0	0	3	16.5	445	51.5

Table 8: Modifying the cost ratio: summary of results

As we might expect, instances become harder when the fixed/variable cost-ratio increases. For the single-source instances the execution time increases when the ratio increases, but the gap reduction at the top node does not show a clear trend. In the multi-source case, we observe that the gap reduction decreases when the cost ratio increases. However, some strange behavior is observed. For example, the time required for the grid graphs and planar graphs is much smaller when the cost ratio is 10^3 than when it is 10^2. The same occurs for series-parallel graphs for ratios 10^3 and ∞. Finally, the number of instances solved to optimality decreases when the cost ratio increases.

7 Conclusions

In spite of the improvements brought about by the introduction of cutting planes into commercial mixed integer programming systems, it appears that the solution of uncapacitated fixed charge network design problems is significantly improved by the use of dicut inequalities either within a special purpose system such as `bc-nd`, or by giving a dicut-strengthened reformulation to an MPS system. Another important step in the development has been the use of a dynamic active node set list as part of the separation heuristic for dicut inequalities. This has been particularly important for certain single-source instances where the number of potential node sets has exploded rapidly.

Further work on the choice of dicut inequalities is probably needed. It would be interesting to design and test separation routines for input-output inequalities, and preliminary tests indicate that the mixing (Günluk and Pochet [17]) of dicut inequalities may be valuable. There remains also the important question of understanding why the duality gaps for multi-source instances are more important, and the need to find new classes of valid inequalities and heuristics for such instances.

Another difficulty concerns actual MIP systems. One reason that `bc-nd` does not more significantly outperform "dicut reformulation + an MIP system" is that the MIP system uses its powerful preprocessor to reduce the tightened formulation, while `bc-nd` keeps the initial tightened formulation as it needs the network structure for cut generation and its primal heuristics. This handicap will only be overcome when the MPS subroutine libraries provide a two way mapping between the original and preprocessed matrices.

Obvious developments are to extend the system to tackle both single commodity *capacitated* fixed charge network design problems, and to *multicommodity* problems. The set X_S (5)-(7) modified with active capacity constraints is the natural starting point as a variety of valid inequalities (flow cover, mixed integer rounding, etc.) can be generated for X_S modified, and the heuristics to generate good node sets S probably need only minor modifications. In addition practically all the inequalities used for multicommodity problems are direct adaptations of single commodity inequalities.

References

[1] Ahuja R.K., Magnanti T.L., and Orlin J.B. *Network Flows.* Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632, 1993.

[2] Atamtürk A. On the network design cut set polyhedra. Draft, Department of Industrial Ingineering and Operations Research, University of California at Berkley, 1999.

[3] Ball M.O., Magnanti T.L., Monma C.L., and Nemhauser G.L. *Network Models*, volume 7 of *Handbooks in OR.* Elsevier Science B.V., 1995.

[4] Barany I., van Roy T.J., and Wolsey L.A. Uncapacitated lot-sizing: The convex hull of solutions. *Math. Programming Stud.*, 22:32–43, 19984.

[5] Barr R.S., Glover F., and Klingman D. A new optimization method for large scale fixed charge transportation problems. *Operations Research*, 29(3):448–463, 1981.

[6] Belvaux Gaëtan. *Modelling and Solving Lot-Sizing Problems by Mixed Integer Programming.* PhD thesis, F.S.A. - Université catholique de Louvain, 1998.

[7] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998. Problems available at *http://www.caam.rice.edu/bixby/miplib/miplib.html.*

[8] Cabot A.V. and Erenguc S.S. Some branch and bound procedures for fixed-cost transportation problems. *Naval Research Logistic Quarterly*, 31:145–154, 1984.

[9] Chopra S., Gorres E., and Rao M.R. Solving a steiner tree problem on a graph using branch and cut. *ORSA Journal on Computing*, 4:320–335, 1992.

[10] Christof T. and Löbel A. *PORTA - A Polyhedron Representation and Transformation Algorithm.* ZIB, Konrad-Zuse-Zentrum für Informationstechnik Berlin.

[11] Cordier C., Marchand H., Laundy R., and Wolsey L.A. `bc-opt`: A branch and cut code for mixed integer programs. *Mathematical Programming*, 86:335–353, 1999.

[12] Cruz F.R.B., MacGregor Smith J., and Mateus G.R. Solving to optimality the uncapacitated fixed-charge network flow problem. *Computers Operations Research*, 25:67–81, 1998.

[13] Dash Associates. *XPRESS-MP Extended Modelling and Optimisation Subroutine Library, Reference Manual,Release 11.* Blisworth House, Blisworth, Northants NN73BX, UK., 1999.

[14] Dash Associates. *XPRESS-MP, Reference Manual,Release 11.* Blisworth House, Blisworth, Northants NN73BX, UK., 1999.

[15] Guisewite G.M. and Pardalos P.M. Minimum concave-cost network flow problems. *Annals of Operations Research*, 25:75–100, 1990.

[16] Günlük O. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, to appear, 1997.

[17] Günlük O. and Pochet Y. Mixing mixed-integer inequalities. DP 9911, CORE, Université catholique de Louvain-la-Neuve, 1998.

[18] Herrman J.W., Ioannou G., Minis I., Nagi R., and Proth J.M. Design of material flow networks in manufactoring facilities. Technical Research Report T.R.94-50, ISRI, University of Maryland, 1994.

[19] Hochbaum D. and Segev A. Analysis of a flow problem with fixed charges. *Networks*, 19:291–312, 1989.

[20] ILOG. *Cplex 6.5, User's Manual*, 2000.

[21] Beasley J.E. An sst-based algorithm for the steiner problem in graphs. *Networks*, 19:1–16, 1989.

[22] Kim D. and Pardalos P. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24:195–203, 1998.

[23] Klingman D., Napier A., and Stutz J. Netgen: A program for generating large scale capacitated assignment, transportation, and minimum-cost flow network problems. *Management Science*, 20:814–820, 1974.

[24] Koch T. and Martin A. Solving steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

[25] Löbel A. *MCF: a network simplex implementation.* Konrad-Zuse-Zentrum für Informationstechnik Berlin, `http://www.zib.de/Optimization/Software/Mcf/`, 2000.

[26] Magnanti T.L. and Wolsey L.A. *Optimal Trees*, chapter 9. Volume 7 of *Handbooks in OR* [3], 1995.

[27] Marchand H., Martin A., Weismantel R., and Wolsey L.A. Cutting planes in integer and mixed integer programming. DP 9953, CORE, Université catholique de Louvain-la-Neuve, 1999.

[28] Martin R.K. and Schrage L. Subset coefficient reduction cuts for 0/1 mixed integer programing. *Operations Research*, 33:505–526, 1985.

[29] Mehlhorn K., Näher S., Seel M., and Uhrig C. *The LEDA User Manual, Version 4.1.* `http://www.mpi-sb.mpg.de/LEDA/MANUAL/MANUAL.html`, 2000.

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

[31] Goemans M.X. personal communication, December 1998.

[32] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & sons, Chichester, 1988. Wiley Interscience Series in Discrete Mathematics and Optimization.

[33] Palekar U.S., Karwan M.K., and Zionts S. A branch and bound method for the fixed charge transportation problem. *Management Science*, 36(9):1092–1105, 1990.

[34] Pochet Y. and Wolsey L.A. Algorithms and reformulations for lot sizing problems. *DIMACS Series in DM and TCS*, 20:245–293, 1995.

[35] Rardin R.L. and Wolsey L.A. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European J. Oper. Res.*, 71:95–109, 1993.

[36] Schaffer J.E. Use of penalties in the branch and bound procedure for the fixed charge transportation problem. *European J. Oper. Res.*, 43:305–312, 1989.

[37] Stan van Hoesel. *Models and Algorithms fot the single item lot-sizing problem.* PhD thesis, Erasmus University, Rotterdam, 1991.

[38] Sun M., Aronson J.E., McKeown P.G., and Drinka D. A tabu search heuristic procedure for the fixed charge transportation problem. *European J. Oper. Res.*, 106:441–456, 1998.

[39] van Roy T.J. and Wolsey L.A. Valid inequalities and separation for uncapacitated fixed charge networks. *Operations Research Letters*, 4(3):105–112, 1985.

[40] van Roy T.J. and Wolsey L.A. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 33(1):45–57, 1987.

Annex A: Characteristics of the test instances

The next two tables give the principal parameters of the instances we have used to test our algorithm. The columns are: the name of the problem, the number of nodes and arcs, the number of source nodes and demand nodes, total demand, and the last column indicates whether the variable costs are zero or not.

Multi-Source Problems

Type	Name	nodes	arcs	nsrc	ndem	totdem	c=0
Grid	g200x740	200	740	100	10	1500	
	g200x740b	200	740	10	80	1000	
	g200x740g	200	740	100	100	1000	
	g200x740h	200	740	100	100	10000	
	g200x740i	200	740	100	100	200	
	g40x132	40	132	10	15	934	
	g50x170	50	170	13	18	1000	
	g55x188	55	188	17	14	1000	
Complete	k10x90	10	90	6	4	100	
	k14x182	14	182	4	6	1000	
	k14x182b	14	182	10	2	1000	
	k16x240	16	240	10	5	794	
	k16x240b	16	240	8	8	1000	
	k20x380	20	380	10	6	437	
Planar	p100x588	100	588	40	25	900	
	p100x588b	100	588	40	25	900	
	p200x1188	200	1188	40	25	900	
	p200x1188b	200	1188	40	25	900	
	p500x2988	500	2988	80	40	10000	
	p500x2988b	500	2988	80	40	10000	
	p50x288	50	288	10	15	500	
	p50x288b	50	288	10	15	500	
	p80x400	80	400	30	15	800	
	p80x400b	80	400	30	15	800	
Random	r20x100	20	100	8	8	1000	
	r20x200	20	200	8	8	1000	
	r30x160	30	160	10	8	1000	
	r50x360	50	360	30	15	100	
	r80x800	80	800	40	25	944	
Series-Parallel	sp100x200	100	200	14	11	51	
	sp150x300	150	300	11	13	90	
	sp150x300b	150	300	15	23	313	X
	sp150x300c	150	300	18	26	1174	
	sp150x300d	150	300	18	33	1135	X
	sp50x100	50	100	18	21	776	
	sp80x160	80	160	4	10	47	
	sp90x180	90	180	16	16	309	
	sp90x250	90	250	13	14	122	

Single-Source Problems

Type	Name	nodes	arcs	nsrc	ndem	totdem	c = 0
Steiner	beasleyC1	500	1250	1	4	4	X
	beasleyC2	500	1250	1	9	9	X
	beasleyC3	500	1250	1	82	82	X
	berlin	52	2652	1	15	15	X
	brasil	58	3306	1	24	24	X
	mc11	400	1520	1	212	212	X
	mc7	400	1520	1	169	169	X
	mc8	400	1520	1	187	187	X
Multi-Segment	beavma	89	195	70	1	14505	
	mtest4ma	100	975	70	1	3959	
	g150x1100	150	1100	1	50	1000	
	g150x1650	150	1650	1	50	1000	
	k15x420	15	420	1	14	95	
	k15x630	15	630	1	14	95	
	p50x576	50	576	1	30	968	
	p50x864	50	864	1	30	968	
MIPLIB	fixnet6	100	500	1	80	500	
Grid	g200x740c	200	740	1	30	10000	
	g200x740d	200	740	1	100	10000	
	g200x740e	200	740	1	150	10000	
	g200x740f	200	740	1	199	10000	
	g180x666	180	666	1	150	10000	
	g55x188c	55	188	1	30	1000	
$K_n + 1$	h50x2450	50	2450	1	49	277	
	h50x2450b	50	2450	1	49	303	
	h50x2450c	50	2450	1	44	252	
	h50x2450e	50	2450	1	44	253	
	h80x6320	80	6320	1	74	408	
	h80x6320b	80	6320	1	69	354	
	h80x6320c	80	6320	1	71	396	
	h80x6320d	80	6320	1	68	384	
Complete	k15x210	15	210	1	14	95	
	k20x380b	20	380	1	10	89	
	k20x380c	20	380	1	15	100	
	k20x380d	20	380	1	19	91	
	k20x380e	20	380	1	5	100	
Lot-Sizing	l121x232	121	232	1	15	4202	
	l451x885	451	885	1	30	5677	
	l451x885b	451	885	1	30	7677	
	l61x114	61	114	1	10	1810	
Planar	p100x588c	100	588	1	5	673	
	p100x588d	100	588	1	5	673	X
	p200x1188c	200	1188	1	6	140	X
	p500x2988c	500	2988	1	6	606	
	p500x2988d	500	2988	1	6	606	X

Annex B: Results

Table 9 reports the results for the single-source problems and Table 10 for the multi-source problems. Within the tables the problems are grouped by graph type. The first column is the name of the instance, the second column is the class the instance belongs to. The third column gives the value of the LP relaxation. The rest of the columns are grouped by system reporting the value of the LP after adding cuts at the top node, the value of the best IP solution found, the gap and the total time used by the system.

Table 11 reports the results for the hard instances after reformulation. The first column is the name of the instance, the second column is the class the instance belongs to. A superscript (1) means the instance is solved by exactly one system after adding dicuts and (2) means the instance is solved by at least two of the systems after adding dicuts. The next column give the value of the LP relaxation after reformulation. The rest of the columns are grouped by system and reporting the value of the LP relaxation improved with system cuts, the value of the best IP solution, the final duality gap, and the execution time is CPU seconds.

Instance	Class	LP	Cplex				bc-opt				mp-opt				bc-nd			
Gaps			XLP	IP	gap	Time	XLP	IP	gap	Time	XLP	IP	gap	Time	XLP	IP	gap	Time
beasleyC1	Easy	52	61.62	85	0.00	302.4	69.82	85	0.00	37.95	80.36	85	0.00	7	79.806	85	0.00	31
beasleyC2	Hard	47.33	75.96	187	44.02	1800.07	113.52	145	9.30	1810.81	127.49	144	4.20	1800	128.8664	144	4.72	1902
beasleyC3	Hard	237.99	442.02	925	50.06	1800.08	673.88	1049	35.28	1815.66	731.95	764	3.67	1800	746.5531	781	4.35	1800
berlin	Hard	130.07	637.4	1372	51.04	1800.24	911.03	2612	65.00	1810.52	675.44	2789	75.59	1800	996.4976	1047	2.75	2094
brasil	Hard	2208.71	9727.25	16610	34.55	1800.31	12342.67	30261	59.21	1813.7	10112.07	52534	80.64	1800	13061.32	13918	4.19	1933
mc11	Hard	608.84	8621.21	27330	67.59	1800.09	10169	18017	43.46	1835.32	11448.27	12167	5.77	1800	11689	11689	0.00	274
mc7	Hard	367.76	2941.22	5931	49.43	1800.11	3158.87	4967	35.87	1831.08	3354.18	3437	2.06	1800	3388.678	3436	1.00	2239
mc8	Hard	91.76	951.16	1922	47.84	1800.09	1203.54	2617	53.92	1827.89	1539.58	1584	2.68	1800	1563.332	1577	0.82	2174
beavma	Easy	158460.2	331940.3	383285	0.00	24.61	380789.1	383285	0.00	3.5	382654.2	383285	0.00	1	383285	383285	0.00	1
mtest4ma	Easy	33968.07	47180.99	52186	0.79	1800.03	50126.39	52148	0.00	528.86	51617.96	52148	0.00	13	52148	52148	0.00	155
g150x1100	Hard	34562.12	54477.49	91918	35.58	1800.03	64772.51	92460	29.31	1815.65	67810.05	76436	10.09	1800	71311.47	72349	1.11	2443
g150x1650	Hard	31884.51	49200.4	90750	40.31	1800.06	58078.06	120702	50.69	1818.96	62704.36	95319	32.90	1800	67440.62	68977	1.99	2404
k15x420	Easy	350.08	768.47	819	0.00	2.58	808.3	819	0.00	4.73	780.73	819	0.00	2	814.7716	819	0.00	13
k15x630	Easy	449.89	830.2	936	0.00	6.88	917.85	936	0.00	9.98	884.42	936	0.00	3	927.2721	936	0.00	41
p50x576	Easy	12203.14	17829.41	19407	0.00	91.7	19257.89	19407	0.00	7.7	19148.07	19407	0.00	5	19399.4	19407	0.00	122
p50x684	Easy	11284.68	17422.39	19007	0.00	49.05	18913.57	19007	0.00	4.7	18779.86	19007	0.00	4	19007	19007	0.00	116
fixnet6	Easy	3192.04	3508.31	3983	0.00	1.49	3548.36	3983	0.00	6.19	3725.49	3983	0.00	7	3959.273	3983	0.00	930
g200x740c	Medium	655765.9	662697.8	680409	0.33	1800.03	673235.6	680164	0.51	1813.16	675206.9	680124	0.00	492	680124	680124	0.00	22
g200x740d	Medium	548918.9	571642.9	592507	1.68	1800.04	584243.5	586038	0.00	442	583761	586038	0.00	131	586038	586038	0.00	36
g200x740e	Medium	559469.2	592314.7	600559	0.19	1800.07	599975.3	600394	0.00	35.02	599132.5	601192	0.00	280	600346.2	600396	0.00	28
g200x740f	Easy	572132.2	613640.2	617876	0.00	56.07	617805.3	617872	0.00	17.76	616894.3	618818	0.00	15	617856	617872	0.00	25
g180x666	Easy	496101.2	613930	624650	0.00	16.01	624460.1	624632	0.00	16.24	623998.2	624632	0.00	48	624632	624632	0.00	26
g55x188c	Easy	22549.39	33070.33	35464	0.00	1.73	35035.8	35464	0.00	3.93	35067.07	35464	0.00	1	35455.99	35464	0.00	5
h50x2450	Medium	131027.8	270133.4	643114	49.97	1800.14	365070.2	371887	0.00	12.63	294207.7	1310087	76.38	1800	371887	371887	0.00	33
h50x2450b	Medium	18.6	37.54	162	73.94	1800.17	44.88	47.07	0.00	59.66	40.67	49	9.04	1800	47.0685	47.1	0.00	26
h50x2450c	Easy	3074.15	3077.77	3096	0.00	7.86	3096	3096	0.00	5.19	3096	3096	0.00	1	3093.595	3096	0.00	4
h50x2450e	Easy	3063.51	3063.51	3094	0.00	8.12	3094	3094	0.00	5.99	3094	3094	0.00	1	3094	3094	0.00	7
h80x6320	Easy	4931.11	4932.33	4990	0.00	54.78	4931.11	5209	5.03	1474.05	4988.21	4990	0.00	39	4982.946	5012	0.54	2067
h80x6320b	Easy	4709.11	4709.11	4728	0.00	110.39	4709.11	4825	2.13	1725.19	4728	4728	0.00	7	4728	4728	0.00	54
h80x6320c	Easy	4858.16	4858.16	4880	0.00	59.29	4880	4880	0.00	209.7	4880	4880	0.00	9	4880	4880	0.00	40
h80x6320d	Medium	4958.42	4958.42	5035	0.00	101.94	4963.73	5230	4.24	1349.24	5007.33	5035	0.02	1800	5029.493	5035	0.11	1993
k15x210	Easy	1982.74	15978.14	16128	0.00	0.59	16065.19	16128	0.00	1.57	16097.15	16128	0.00	1	16126.14	16128	0.00	9
k20x380b	Easy	1764.92	11248.76	11343	0.00	0.89	11343	11343	0.00	1.08	11343	11343	0.00	1	11343	11343	0.00	3
k20x380c	Easy	1858.48	16916.58	17159	0.00	1.95	17079.16	17159	0.00	4.76	17015.41	17159	0.00	4	17124.47	17159	0.00	108
k20x380d	Easy	1937.75	20884.58	20979	0.00	0.98	20954.24	20979	0.00	2.52	20944.99	20979	0.00	3	20975.97	20979	0.00	51
k20x380e	Easy	2013.35	6804.97	6904	0.00	0.89	6904	6904	0.00	1.39	6904	6904	0.00	0	6876.918	6904	0.00	0
l121x232	Easy	151280.9	151678.8	152438	0.00	0.33	152420.5	152438	0.00	1.36	152420.5	152438	0.00	1	152438	152438	0.00	0
l451x885	Easy	389467	390149.6	393757	0.00	96.11	392454	393757	0.00	1343.6	393556.9	393757	0.00	3	393757	393757	0.00	25
l451x885b	Easy	519584.4	520271.3	523782	0.00	16.78	522615.2	523780	0.00	1123.99	523552.1	523780	0.00	3	523771.5	523780	0.00	55
l61x114	Easy	51809.12	51936.61	52327	0.00	0.11	52327	52327	0.00	0.3	52327	52327	0.00	0	52327	52327	0.00	0
p100x588c	Easy	97579.11	140682.3	172770	0.00	11.34	156839.8	156840	0.00	1.19	170801.7	172770	0.00	5	172770	172770	0.00	11
p100x588d	Easy	1.9	5	5	0.00	0.41	5	5	0.00	1.37	5	5	0.00	2	5	5	0.00	2
p200x1188c	Easy	5678.61	9897.96	15078	0.00	253.89	13703.34	15078	0.00	130.89	12949.11	15078	0.00	608	15078	15078	0.00	10
p500x2988c	Easy	14122.96	14759.02	15215	0.00	5.99	15032.13	15215	0.00	89.76	14854.12	15215	0.00	5	15214.2	15215	0.00	20
p500x2988d	Easy	2.16	6	6	0.00	2.44	6	6	0.00	54.4	6	6	0.00	27	6	6	0.00	9

Instance	Class	LP	Cplex				bc-opt				mp-opt				bc-nd			
Gaps			XLP	IP	gap	Time	XLP	IP	gap	Time	XLP	IP	gap	Time	XLP	IP	gap	Time
g200x740	Hard	34077.54	41177.85	46327	8.26	1800.02	43496.47	44362	1.08	1813.41	43589.96	44460	1.35	1800	44148.45	44360	0.33	1915
g200x740b	Hard	145993.9	166186.4	183438	5.83	1800.03	175320.6	179538	1.43	1812.23	175208.2	180054	2.11	1802	178775.6	179279	0.00	1878
g200x740g	Hard	7911.51	31272.16	57962	43.36	1800.03	35142.63	53325	33.83	1817.74	37988.75	52718	27.54	1800	39145.76	45472	13.47	1977
g200x740h	Hard	86698.84	118727.1	163076	26.43	1800.06	124655.1	134775	7.17	1812.85	127127.3	136073	6.31	1800	129122.5	134222	3.59	1939
g200x740i	Hard	2292.47	21839.27	42468	46.68	1800.06	25668.87	40300	36.18	1820.75	26816.26	37916	29.06	1800	27893.84	33733	16.86	2070
g40x132	Easy	13997.27	22156.21	26629	0.00	957.48	25002.84	26629	0.00	238.66	25824.66	26629	0.00	30	26076	26629	0.00	10
g50x170	Easy	10711.59	20603.48	25576	1.54	1800.01	24146.06	25576	0.00	230.16	24546.03	25576	0.00	229	25153.19	25576	0.00	22
g55x188	Hard	9178.6	19154.27	27100	20.40	1800.01	21674.3	26938	15.28	1800.84	22630.61	24706	5.43	1800	23728.47	24487	0.00	1143
k10x90	Easy	399.73	522.39	568	0.00	2.35	554.27	568	0.00	3.63	546.62	568	0.00	1	558.3744	568	0.00	0
k14x182	Easy	3995.51	7245.82	8491	0.00	24.9	7887.18	8491	0.00	65.17	7656.1	8491	0.00	48	7990.55	8491	0.00	9
k14x182b	Easy	4442.84	10066.09	11042	0.00	13.84	10817.82	11042	0.00	5.26	10768.54	11042	0.00	8	10820.31	11042	0.00	4
k16x240	Hard	2769.84	8578.39	10872	11.38	1800.02	8919.04	11203	15.41	1801.41	9077.93	11053	13.45	1800	9136.041	10877	9.27	1804
k16x240b	Hard	3320.77	9056.5	12243	17.74	1800.01	9856.58	12300	14.69	1802.67	9649.04	12099	15.54	1800	10054.16	12078	11.27	1810
k20x380	Hard	952.57	1649.61	1941	1.59	1800.01	1786.8	1951	3.03	1802.79	1741.29	1986	7.05	1800	1829.364	1941	0.04	406
p100x588	Hard	3957.22	7628.41	9505	11.89	1800.02	8279.28	9522	12.00	1806.34	8516.9	9165	5.89	1800	8745.126	9049	2.66	1911
p100x588b	Hard	5554.01	37223.14	60990	35.22	1800.02	43168.64	54055	19.30	1807.6	44226.4	55642	19.88	1800	45001.27	50572	10.20	1999
p200x1188	Hard	5575.13	9089.26	12186	17.73	1800.06	10638.6	12075	11.21	1813.92	10939.92	11865	7.26	1800	11233.36	11496	1.85	2018
p200x1188b	Hard	7860.8	38583.22	73516	42.79	1800.06	50314.68	75057	32.51	1818.55	51062.7	73862	30.55	1800	52931.98	58672	9.40	2107
p500x2988	Hard	59130.67	67046.23	74125	7.56	1800.44	70238.56	72336	2.58	1831.56	70041.6	73646	4.70	1800	71683.29	71845	0.14	2313
p500x2988b	Hard	61188.7	136874.3	489414	71.44	1800.08	155816.2	221307	29.53	1833.35	147778.6	270294	45.08	1800	169919.5	177370	4.02	2559
p50x288	Easy	4178.48	5592.12	6134	0.00	94.85	5911.76	6134	0.00	155.58	6041.18	6134	0.00	132	6113.48	6134	0.00	8
p50x288b	Hard	5818	17829.84	23298	11.33	1800.01	20091.74	21916	3.82	1802.28	20766	21916	2.78	1800	20993.56	21753	0.19	1811
p80x400	Hard	4824.65	7585.43	8752	6.57	1800.03	8082.47	8786	6.61	1803.02	8264.62	8585	2.82	1800	8450.915	8548	0.00	1491
p80x400b	Hard	6418.8	30469.83	45865	28.12	1800.02	34878.82	43318	16.49	1804.64	36190	41916	12.15	1800	37021.15	41442	9.19	1863
r20x100	Easy	6747.22	13166.21	15603	0.00	72.77	15033.57	15603	0.00	11	15103.81	15603	0.00	18	15172.52	15603	0.00	5
r20x200	Hard	5088.26	12084.78	15281	7.66	1800.03	12990.22	14517	3.15	1801.02	13003.46	14783	5.41	1800	13258.61	14877	3.25	1806
r30x160	Easy	10608.07	18731.89	21827	0.00	1049.48	19721.5	21827	0.00	1633.51	20341.18	21827	0.25	1800	20377.88	21827	0.00	479
r50x360	Hard	575.02	1525.51	1786	10.15	1800.03	1598.5	1684	3.29	1806.85	1582.12	1757	8.64	1800	1605.47	1698	3.95	1829
r80x800	Hard	3651.48	4912.19	5569	9.40	1800.13	5090	5551	7.90	1809.45	5062.53	5680	10.40	1800	5190.14	5425	3.92	1907
sp100x200	Easy	18515.86	26357.77	34507	0.00	913.76	31762.62	34507	0.00	56.98	33620.66	34507	0.00	1	34434.59	34507	0.00	34
sp150x300	Medium	12632.31	20331.26	31833	7.44	1800.04	27959.55	34806	14.52	1802.65	30083.1	30918	0.00	9	30918	30918	0.00	9
sp150x300b	Hard	36.77	41.08	57	13.25	1800.03	51.26	56	5.32	1803.44	52.13	57	6.98	1800	55.3001	56	0.00	250
sp150x300c	Medium	400358.1	470816.2	566944	2.89	1800.02	519324.9	549595	0.00	146.73	553873.1	560736	0.00	2	556818.7	560736	0.00	35
sp150x300d	Medium	34.14	49.51	69	11.70	1800.02	58.73	69	5.92	1803.14	68.51	69	0.00	138	68.5855	69	0.00	40
sp50x100	Easy	48292.37	49936.38	50968	0.00	0.05	48174.89	48422	0.00	0.26	50968	50968	0.00	0	50968	50968	0.00	0
sp80x160	Easy	14573.39	16642.59	19549	0.00	2.56	19101.22	19549	0.00	2.03	19549	19549	0.00	0	19549	19549	0.00	0
sp90x180	Easy	57349.9	63233.66	68862	0.00	1.07	66478.8	67570	0.00	1.73	68656.9	68862	0.00	1	68862	68862	0.00	1
sp90x250	Easy	18745.35	20868.32	23571	0.00	3.79	23172.61	23571	0.00	1.71	23571	23571	0.00	0	23571	23571	0.00	3

Table 10: Multi-source instances: detailed results

Instance	Class	LP	Cplex				bc-opt				mp-opt			
			XLP	IP	gap	Time	XLP	IP	gap	Time	XLP	IP	gap	Time
g200x740	Hard(1)	44148.45	44148.56	44316	0.00	863	44154.16	44393	0.39	1817	44165.34	44342	0.25	1800
g200x740b	Hard(2)	178775.61	178778.03	179279	0.00	59	178800.62	179279	0.00	455	178927.7	179279	0.00	201
g200x740g	Hard	39145.76	39148.66	55717	29.38	1800	39213.23	50357	22.00	1841	39547.61	51323	22.71	1800
g200x740h	Hard	129122.52	129150.23	136935	5.49	1800	129154.97	132882	2.64	1817	129577.07	134274	3.35	1800
g200x740i	Hard	27893.84	27894.13	38981	27.96	1800	27933.12	34624	18.98	1850	28106.2	37278	24.36	1800
g55x188	Hard(2)	23728.47	23730.02	24487	0.00	1308	23731.28	24487	0.00	262	23778.12	24487	0.00	400
k16x240	Hard	9136.04	9136.04	10862	8.55	1800	9136.04	10792	10.20	1801	9183.44	10807	10.35	1800
k16x240b	Hard	10054.16	10058.4	11905	9.17	1800	10057.92	11599	7.13	1804	10101.49	11490	6.99	1800
k20x380	Hard(2)	1829.36	1829.36	1941	0.00	156	0	1941	0.00	1024	1833.22	1941	0.00	686
p100x588	Hard	8745.13	8747.17	9071	2.28	1800	8749.11	9145	3.47	1807	8771.74	9122	3.06	1800
p100x588b	Hard	45001.27	45009.88	52741	13.60	1800	45109.5	53146	14.31	1808	45205.22	56021	18.64	1800
p200x1188	Hard	11233.36	11233.56	11637	2.68	1800	NP				11257.82	11578	2.47	1800
p200x1188b	Hard	52931.98	52932.09	62150	14.17	1800	NP				53083.68	71229	25.10	1800
p500x2988	Hard	71683.29	71684	71923	0.18	1800	71685.24	71907	0.24	1827	71684.31	71905	0.23	1800
p500x2988b	Hard	169919.52	169944.71	184979	7.70	1801	169994.4	191745	11.28	1887	169939.89	196432	13.36	1800
p50x288b	Hard(2)	20993.56	20993.56	21753	0.00	593	20993.91	21753	0.00	1247	21050.59	21753	0.00	1219
p80x400	Hard(2)	8446.93	8451.76	8548	0.00	80	8456.25	8548	0.00	400	8482.93	8548	0.00	579
p80x400b	Hard	37021.15	37025.71	41719	8.96	1800	37036.42	41822	9.70	1802	37353.08	43206	12.44	1800
r20x200	Hard	13258.61	13258.72	14803	2.53	1800	13258.75	14796	3.10	1801	13345.65	14796	3.65	1800
r50x360	Hard	1605.47	1605.47	1669	1.36	1800	1606.08	1675	2.12	1804	1608.66	1697	3.50	1800
r80x800	Hard	5190.14	5190.14	5538	5.62	1800	5190.19	5362	2.86	1806	5208.66	5456	4.25	1800
sp150x300b	Hard(2)	55.33	55.37	56	0.00	11	55.43	56	0.00	644	55.86	56	0.00	36
beasleyC2	Hard(1)	128.87	128.99	144	1.40	1800	129.78	144	2.87	1814	136.31	144	0.00	529
beasleyC3	Hard(1)	746.55	746.75	784	4.16	1800	748.03	757	0.99	1840	751.45	754	0.00	1175
berlin	Hard(2)	996.5	1001.72	1045	0.96	1800	1006.25	1044	1.06	1848	1004.59	1068	4.91	1800
brasil	Hard(2)	13061.32	13061.32	14504	8.55	1801	13236.49	13679	1.24	1913	13069.71	13695	4.44	1800
mc11	Hard(1)	11689	11689	11689	0.00	82	11689	11689	0.00	16	11689	11689	0.00	2
mc7	Hard(1)	3388.68	3388.8	3503	2.72	1800	3390.18	3442	1.32	1882	3388.98	3432	0.94	1800
mc8	Hard	1563.33	1563.35	1568	0.18	1800	1563.36	1566	0.09	1858	1563.36	1566	0.11	1800
g150x1100	Hard(2)	71311.47	71311.47	71816	0.00	449	71411.09	71816	0.00	1793	71608.08	71816	0.00	521
g150x1650	Hard	67440.62	67440.95	70386	3.08	1800	67568.73	68936	1.80	1926	67788.26	68997	1.56	1800

NP : numerical problem.

Table 11: Hard instances : reformulated