A Local Branching Heuristic for Mixed-Integer Programs with 2-Level Variables, with an Application to a Telecommunication Network Design Problem

Matteo Fischetti, Carlo Polo, Massimo Scantamburlo

DEI, University of Padova, Via Gradenigo 6/A, 35100 Padova, Italy

e-mail: matteo.fischetti@unipd.it

April 2003; Revised, October 2003

Abstract

Effective heuristic solution methods for general Mixed-Integer Programs (MIPs) are strongly required in many practical applications, and have been the subject of an intensive research effort in the recent years.

Fischetti and Lodi [6] recently proposed an exact solution technique based on the use of branching conditions expressed through (invalid) linear inequalities called *local branching* cuts. In the concluding remarks of their paper, these authors anticipated the possibility their method be used to design a genuine MIP metaheuristic framework akin to Tabu Search (TS) or Variable Neighborhood Search (VNS), based on an external MIP solver.

In the present paper we introduce and analyze computationally a specific implementation of the above idea. In particular, we address MIPs with binary variables, and propose a variant of the classical VNS scheme that we call *Diversification, Refining, and Tight-refining* (DRT). The new approach is intended to be of high generality, but exploits the specific structure of some MIPs where the set of binary variables partitions naturally into two levels, with the property that fixing the value of the first-level variables produces an easier-to-solve (but still hard) subproblem. This is often the case, e.g., in hard facility location problems arising in telecommunication network design.

Our method detects automatically the presence in the MIP model of the first-level binary variables, if any, according to simple heuristic criteria. This information is then exploited during the intensification phase of the local search, so as to explore nested solution neighborhoods defined by local branching cuts affecting the first- and second-level variables in a different way.

Computational results for a hard facility location problem arising in telecommunication UMTS network design [7] are presented, with a comparison among

different heuristic methods: the general-purpose commercial ILOG-Cplex 7.0 MIP code, the original Fischetti-Lodi local branching heuristic scheme, the new DRT method, and (for some instances) ad-hoc TS and VNS heuristics. The results show that the DRT method, though very simple to implement, provides better heuristic solutions than the alternative methods on all the instances of our test bed.

Key words: Mixed-Integer Programs, Heuristics, Facility Location Problems

1 Introduction

Effective heuristic solution methods for general Mixed-Integer Programs (MIPs) are strongly required in many practical applications, and have been the subject of an intensive research effort in the recent years; see [2], [4], [8], [9], [10], [11], [12], [15], [16], and [18], among others.

Fischetti and Lodi [6] recently proposed an exact solution technique based on the use of branching conditions expressed through (invalid) linear inequalities called *local branching* cuts. They considered a generic MIP with 0-1 variables of the form:

$$
\begin{aligned}
(P) \qquad & \min c^T x && && (1)\\
& Ax \geq b && && (2)\\
& x_j \in \{0,1\} && \forall j \in \mathcal{B} \neq \emptyset && (3)\\
& x_j \geq 0, \text{ integer} && \forall j \in \mathcal{G} && (4)\\
& x_j \geq 0 && \forall j \in \mathcal{C} && (5)
\end{aligned}
$$

where the variable index set $\mathcal{N} := \{1, \cdots, n\}$ is partitioned into $(\mathcal{B}, \mathcal{G}, \mathcal{C})$. Here $\mathcal{B} \neq \emptyset$ is the index set of the 0-1 variables, whereas the possibly empty sets $\mathcal{G}$ and $\mathcal{C}$ index the general integer and the continuous variables, respectively.

Given a feasible *reference solution* $\bar{x}$ of (P) and a nonnegative integer parameter k, the *k-OPT neighborhood* of $\bar{x}$ is defined as the set of the feasible solutions of (P) satisfying the additional *local branching* constraint:

$$
\Delta(x, \bar{x}) := \sum_{j \in \mathcal{B}: \bar{x}_j = 1} (1 - x_j) + \sum_{j \in \mathcal{B}: \bar{x}_j = 0} x_j \leq k \tag{6}
$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{x}$) either from 1 to 0 or from 0 to 1, respectively.

The local branching constraint is used in [6] as a branching criterion within an enumerative scheme for (P). Indeed, given the incumbent solution $\bar{x}$, the solution space associated with the current branching node can be partitioned by means of the disjunction

$$
\Delta(x, \bar{x}) \leq k \quad \text{or} \quad \Delta(x, \bar{x}) \geq k + 1 \tag{7}
$$

where the neighborhood-size parameter k is chosen appropriately. The approach alternates high-level strategic phases where the local branching cuts are used to

define 'promising' solution regions, and low-level tactical phases where these regions are implicitly enumerated through a classical branching-on-variable strategy. The result is a completely general exact scheme aimed at favoring early updatings of the incumbent solution, hence producing high-quality solutions at early stages of the computation.

In the concluding remarks of their paper, Fischetti and Lodi anticipated the possibility their method be used to design a genuine MIP metaheuristic framework akin to Tabu Search (TS) [10] or Variable Neighborhood Search (VNS) [17]. Indeed, all the main ingredients of these metaheuristics (defining the current solution neighborhood, dealing with tabu solutions or moves, imposing a proper diversification, etc.) can easily be modelled in terms of linear cuts to be dynamically inserted and removed from the model. The metaheuristic scheme is then built on top of a general-purpose MIP solver, which is used as a black box to explore solution neighborhoods defined by the original MIP model amended by suitable local branching cuts. This leads naturally to a completely general TS or VNS framework for MIPs, which is easy to implement and hopefully as effective as (and sometimes even better than) ad-hoc TS or VNS methods designed for specific problems.

In the present paper we introduce and analyze computationally a specific implementation of the above idea. Our aim is to show how the local branching paradigm can be specialized for important classes of problems, so as to obtain a (still quite general) heuristic scheme with improved performance. In particular, we propose a variant of the classical VNS scheme that we call *Diversification, Refining, and Tight-refining* (DRT). The new approach is particularly suited for those MIPs in which the set of binary variables can be partitioned into two sets (called *levels*), with the property that fixing the value of the first-level variables produces an easier-to-solve (but non trivial) subproblem.

In particular, our method applies to any network design problem which involves two levels of decisions: a choice of facilities to install on arcs or nodes of a network, and the determination of routings connecting different commodities; see [3] for a recent annotated bibliography on network design. A familiar example is the Facility (or Plant) Location Problem (FLP) with hard side constraints, where the first-level binary variables correspond to the choice of the facilities to be activated, whereas the second-level binary variables refer to link activation between facilities and customers. These kinds of models are very important in practice [13] and arise, e.g., in UMTS telecommunication network design (see [1, 7, 19, 20], among others). A second example arises for MIPs involving big-M coefficients for the first-level variables, whose negative effect in the quality of the LP relaxation vanishes as soon as these variables are fixed to either 0 or 1.

The method we propose detects automatically the presence in the MIP model of the first-level binary variables, if any, according to simple heuristic criteria. This information is then exploited during the intensification phase of the local search, so as to explore nested solution neighborhoods defined by local branching cuts constraining the variation of the first- and second-level variables in a different way. In particular, during the refining phases the first-level variables are (almost) fixed to their current value in the reference solution, so as to allow for an effective redefinition

of all other variables.

The paper is organized as follows. The basic DRT method is described in Section 2. In Section 3 we discuss how to partition automatically the binary variables into the two level sets required by the DRT method. Section 4 presents computational results on the UMTS telecommunication network design problem addressed in [7]. We compare different heuristic methods: the general-purpose commercial MIP solver ILOG-Cplex 7.0, the original Fischetti-Lodi local branching heuristic scheme, the new DRT method, and (for some instances) ad-hoc TS and VNS heuristics. The results show that the DRT method provides better heuristic solutions than the alternative methods on all the instances of our test bed. Some conclusions are drawn in Section 5, whereas a DRT pseudo-code and the parameter files used in the computational tests are given in the Appendix.

2 The DRT method

Let us consider the MIP problem (P) defined in the introduction, and assume the binary variable index set $\mathcal{B}$ has been partitioned into $(\mathcal{B}_1, \mathcal{B}_2)$, where the (possibly empty) sets $\mathcal{B}_1$ and $\mathcal{B}_2$ correspond to the first- and second-level variables, respectively.

Given a current target solution $\bar{x}$ of problem (P), we aim at exploring as effectively as possible the solution neighborhood obtained by (almost) fixing the first-level variables to their current value in $\bar{x}$. In this phase, called *refining*, we start by simply adding the following constraint

$$\Delta_1(x, \bar{x}) := \sum_{j \in \mathcal{B}_1 : \bar{x}_j = 1} (1 - x_j) + \sum_{j \in \mathcal{B}_1 : \bar{x}_j = 0} x_j \leq k_1 \tag{8}$$

to the current MIP, where parameter k_1 is fixed to a very small value (e.g., $k_1 = 2$ or even $k_1 = 0$). We then apply a general-purpose MIP solver in the attempt to solve the resulting model–the input upper bound for the MIP solver is set to the value of the best feasible solution so far.

If the model is not solved to proven optimality within a given time limit, we enter a *tight-refining* heuristic phase where the first-level variables are still constrained by (8), but we also limit the variation of the second-level variables through local-branching constraints of the type

$$\Delta_2(x, \bar{x}) := \sum_{j \in \mathcal{B}_2 : \bar{x}_j = 1} (1 - x_j) + \sum_{j \in \mathcal{B}_2 : \bar{x}_j = 0} x_j = k_2 \tag{9}$$

for increasing values of the neighborhood size $k_2 = 0, k_2^{step}, 2k_2^{step}, \cdots, k_2^{max}$, where k_2^{step} and k_2^{max} are input parameters to the DRT procedure (typically, $k_2^{step} = 2$ and $k_2^{max} = 10$). In this way we explore through the general-purpose MIP solver (with an appropriate time limit) a sequence of second-level neighborhoods, all of which are contained in the first-level neighborhood defined by (8). The tight-refining phase ends when k_2 reaches its maximum allowed value k_2^{max}, or when the overall time limit for this phase is reached.

We then remove constraints (8) and (9) from the current MIP, and proceed by considering a different setting for the first-level variables. This *diversification* phase is implemented according to the VNS idea that the new setting for the first-level variables should be (different but) sufficiently close to the current one. To be specific, we add the diversification constraint

$$k_1^{min} \leq \Delta_1(x, \bar{x}) \leq k_1^{max} \tag{10}$$

to the current model, and look for any (in principle, random) feasible solution of it. As in [6], this is obtained through the black-box MIP solver, which is run with no input upper bound and aborted as soon as the first heuristic solution is found. This hopefully produces a good-quality (but still almost random) solution to be used as the reference point $\bar{x}$ for the subsequent iteration. If no solution is found within a given time limit, parameters k_1^{min} and k_1^{max} are iteratively modified so as to define larger and larger neighborhoods, according to a simple rule described in the Appendix.

It should be observed that the above scheme can lead us to consider twice a same first-level variable setting. To avoid this risk, before replacing the current reference solution $\bar{x}$ with the new one, we follow the TS paradigm and add the *tabu constraint*

$$\Delta_1(x, \bar{x}) \geq 1 \tag{11}$$

to the current MIP, in a static way (i.e., this constraint will never be removed from the model).

The DRT method requires a clever choice of the first-level variables. This choice can be left to the model designer who can provide explicitly the list of the first-level variables, based on his/her understanding of the MIP problem at hand. Simple criteria will be described in the next section to derive a fast automatic procedure that works well in many practical cases arising in network design problems.

Moreover, it should be observed that our DRT method works correctly even in case no first-level variables exist (i.e., when $\mathcal{B}_1 = \emptyset$), as in this case it becomes a descent method based the exploration of a sequence of solution neighborhoods defined by the local branching constraints (9).

The overall DRT method is outlined in the figure below. In each step, the so-far best feasible solution is implicitly updated. On the first execution of the repeat-until loop, a diversification (rather than a refining) step is immediately performed, as the first reference solution $\bar{x}$ found by some heuristic is likely to be of good quality, hence fixing its first-level variables typically does not result into a significant improvement.

The way we choose the DRT parameters–in particular, the time limit for each phase–can lead to different results. Setting a large time limit for the intensification phase will lead us to exploring in a deep way just a few first-level variable configurations. In some cases, better results are expected by allowing for a larger number of diversifications, even if a short time limit is imposed for each intensification. We have therefore implemented a DRT variant, called DRT2, where we try to prevent

Algorithm 1 : The overall DRT method

find heuristically a "good" starting reference solution $\bar{x}$, e.g., by applying the MIP solver with a short time limit;

repeat

add statically the tabu constraint (11) to the current MIP;

add temporarily the diversification constraint (10) to the current MIP, and apply the MIP solver (with no upper bound) to find a first feasible solution that replaces $\bar{x}$;

remove the diversification constraint, and (almost) fix the first-level variables to their value in $\bar{x}$ by adding temporarily constraint (8) to the current MIP;

solve heuristically the resulting MIP through the MIP solver, possibly adding a sequence of temporary local branching constraints (9) for nested-neighborhood exploration;

remove all temporary constraints

until the overall time limit is exceeded

spending too much computing time in exploring first-level configurations not leading to a significant improvement of the current solution. To this end, as soon as we detect the risk of stalling we force the algorithm to make a *big diversification* step in which a larger number of first-level variables is required to change. Parameters of this strategy are (a) the percentage of improvement used to detect the situation of stalling, (b) the maximum number of no-improvement iterations before a big diversification, (c) the number of first-level variables to change in a big diversification phase, and (d) the maximum number of big diversifications allowed.

3 Detecting first-level variables automatically

In many important cases, the logical structure of a MIP model implies a self-evident two-level variable structure. We have addressed two different situations, associated with the presence into the model of (a) "logical dependencies" between binary variables, or (b) big-M coefficients controlled by binary variables.

As an example of the first case, consider the well-known Facility Location Problem (FLP) [13]. We are given a set N of customers along with a set M of possible locations for activating the facilities. Let c_{ij} be the cost for connecting customer $i \in N$ to the facility active in location $j \in M$, and let f_j be the fixed set-up cost to install a facility in location $j \in M$. The problem consists in determining the locations where the facilities have to be activated as well as the way to assign the customers to them, with the goal of minimizing the overall cost for installing the facilities and for connecting them to the customers. A simple MIP model for FLP then reads:

$$minimize \sum_{i \in N} \sum_{j \in M} c_{ij} x_{ij} + \sum_{j \in M} f_j y_j$$

subject to

$$\sum_{j \in M} x_{ij} = 1 \quad \forall i \in N \tag{12}$$

$$x_{ij} \leq y_j \quad \forall i \in N, j \in M \tag{13}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in N, j \in M \tag{14}$$

$$y_j \in \{0,1\} \quad \forall j \in M \tag{15}$$

where $y_j = 1$ iff a facility is activated in location $j \in M$, and $x_{ij} = 1$ iff customer $i \in N$ is assigned to the facility in location $j \in M$. Constraints (12) state that each customer has to be assigned to exactly one facility location, whereas constraints (13) impose that customers can only be assigned to locations where a facility has been installed.

The structure of the model above leads to a sort of "logical dependency" of the x-variables from the y-variables, in the sense that, because of (13), setting $y_j = 0$ forces to zero all the variables x_{ij} for $i \in N$. Hence the y and the x variables qualify to play the role of first- and second-level variables, respectively.

In some cases, the logical dependency between binary variables can be hidden somehow in the model. This arises, e.g., when the model contains "aggregated" constraints as in the capacitated version of FLP. Here each facility location $j \in M$ has a given capacity $K_j > 0$ whereas each customer $i \in N$ has a certain facility request $k_i \geq 0$. This introduces into the model constraints of the form:

$$\sum_{i \in N} k_i x_{ij} \leq K_j y_j, \quad \forall j \in M \tag{16}$$

As before, a simple analysis of the constraint shows that setting $y_j = 0$ implies $x_{ij} = 0$ for $i \in N$ with $k_i > 0$. Hence, the presence into the MIP model of constraints of the type (16) can be used to detect automatically first-level variables, as outlined in the sequel.

A second practical case in which the distinction between first- and second-level variables can hopefully be automatized arises for MIP models where certain binary variables are used to activate/deactivate linear constraints as, e.g., in the constraint below:

$$\sum_i w_i x_i \leq w_0 + My \tag{17}$$

where y is a binary variable and M is a large positive value. The logic underlying this kind of constraints is that one wants to either impose the linear constraint $\sum_i w_i x_i \leq w_0$, or else requires $y = 1$ so as to pay a certain cost in the objective function (or to activate other restrictions associated with case $y = 1$). It is however well know that the LP relaxation of the model is only marginally affected by constraints of the form (17), since the LP solution (x^*, y^*) can have $\sum_i w_i x_i^* > w_0$ without the need

of setting $y^* = 1$, since the (very small) fractional value $y^* := \sum_i (w_i x_i^* - w_0)/M$ already suffices to fulfill (17). This behavior makes constraints of the form (17) almost useless in the LP relaxation, which then produces very poor lower bounds for the MIP problem. In this respect, fixing variable y is likely to have a strong effect on the solvability of the resulting MIP problem, which suggests defining y as a first-level variable.

Detecting automatically (a) logical dependencies between variable pairs, and (b) the presence of binary variables used to activate/deactivate some constraints, essentially consists in performing a sensitivity analysis of the LP relaxation of the MIP model. Indeed, a binary variable x_j qualifies for playing the role of a first-level variable in case (a) setting $x_j = 0$ forces the setting of other integer variables in the LP relaxation, or (b) setting $x_j = 1$ (or $x_j = 0$) makes a specific constraint redundant in the LP relaxation. In order to avoid a time-consuming analysis of this type, however, in our implementation we used the following very simple heuristic criteria.

We initialize $\mathcal{B}_1 = \emptyset$, and possibly include some predefined variables. We then consider, in sequence, each constraint

$$\sum_j \alpha_j x_j \leq \alpha_0 \qquad (\text{or} \sum_j \alpha_j x_j = \alpha_0) \tag{18}$$

in the MIP model that involves at least two integer variables (one of which binary). Our first criterion for adding a binary variable to the current first-level set $\mathcal{B}_1$ is related to the concept of "logical dependency" between variables. Assuming that each variable x_j is bounded by $LB_j \leq x_j \leq UB_j$, the minimum value for the left-hand side term in (18) is obtained when all variables with a positive coefficient α_j are set to their lower bound LB_j, whereas all variables with a negative coefficient α_j are set to their upper bound UB_j. With this variable setting, the total slack for the constraint attains its maximum value, namely:

$$\delta_{max} := \alpha_0 - \big(\sum_{j:\alpha_j>0} \alpha_j LB_j + \sum_{j:\alpha_j<0} \alpha_j UB_j \big) \tag{19}$$

Our first condition to insert a binary variable x_j into the first-level set $\mathcal{B}_1$ is:

$$\alpha_j < 0 \ : \ |\alpha_j|(UB_j - LB_j) \geq \delta_{max} \tag{20}$$

as in this case setting $x_j = LB_j (= 0)$ implies that all other variables with a nonzero coefficient α_j are forced to keep their (upper or lower) bound value as in (19).

Our second criterion for inserting a binary variable x_j in $\mathcal{B}_1$ is aimed at capturing the presence of a big-M coefficient α_j in (18). To this end, we compute the average value α_{AVG} of the nonzero $|\alpha_j|$'s in the left-hand side of (18), and insert a binary variable x_j into $\mathcal{B}_1$ in case $|\alpha_j| > bigMperc \cdot \alpha_{AVG}$, where $bigMperc$ is a predefined parameter (after some testing, we found that $bigMperc = 5$ works well in most applications).

Though very simple, the above procedure proved adequate in detecting automatically the first-level variables, in particular for the MIP models that typically arise in telecommunication network design. More sophisticated approaches can however be designed, which also take into account "logical dependencies" related to setting certain binary variables to value 1, make a more clever sensitivity analysis of the LP relaxation of the model, etc. For the most difficult cases, the model designer himself can integrate the procedure by providing on input to the DRT method a (partial or complete) list of forced first-level binary variables.

4 Computational results

We have performed tests on several instances of a hard FLP problem arising in UMTS (standing for Universal Mobile Telecommunication System) network planning, as addressed in [7]. Runs were made on a PC Pentium III/1GHz with 380MB RAM, Microsoft Visual C++ 6.0 compiler, and Windows XP OS. For each run we imposed a limit time of 3 hours (10,800 seconds). ILOG Cplex 7.0 [5] was used as the black-box external MIP solver.

4.1 The problem

The basic architecture of a UMTS network includes the following devices:

- Mobile Terminal (MT) of different types (e.g., phone, fax, video, computer, etc.).
- Base Transceiver Station (BTS) interfacing mobile users to the fixed network; a BTS handles users' access and channel assignment. Due to the inherent BTS flexibility, different network topologies can be undertaken: the BTS can be either directly connected to the switching equipment (smart BTS) or linked to a BTS controller (CSS).
- Cell Site Switch (CSS), which is a switch connected to several BTS's on one side and to a single Local Exchange (see below) on the other side; each CSS is devoted to the management of local traffic inside its controlled area, as well as to the connection of the controlled BTS's to the Local Exchange; for technical reasons, CSS's are often of two types (type 1 and 2) and differ, e.g., for the cost and maximum controlled traffic.
- Local Exchange (LE), which is a switch connecting the BTS's to the network, either directly or through CSS's.

In UMTS jargon, BTS's are often called *terminals* while CSS's and LE's are called *concentrators*. The network with BTS-CSS or BTS-LE links is known as *access network* and the one with CSS-LE links is the first-level *fixed network*. They both can be star-shaped or generic networks. The design of a UMTS network can then be subdivided into three phases: (1) decide the number and locations of the concentrators and the assignment of terminals to the concentrators; (2) design the ac-

cess network (terminals-concentrators); and (3) design the first-level fixed network (concentrators-concentrators).

In the specific problem we consider, a certain number of potential CSS's and LE's sites is given, among which the planner has to choose those to be actually activated. We consider a three level star-shaped UMTS architecture, defined by an upper layer made up of active LE's (chosen in the given set of potential LE's), a middle layer made up of active CSS's (also chosen in the given set of potential CSS's), and a lower layer made up of the given BTS's (each of which is required to play the role of a leaf in the star-type structure). The problem then consists in choosing the CSS and LE to be activated, and the way to connect them to the BTS's and between each other, so as to produce a feasible three-level network of minimum cost; see Figure 4.1 for an illustration. Moreover, a number of hard side-constraints are imposed on the number of devices and on the traffic that can be associated to each concentrator, etc.

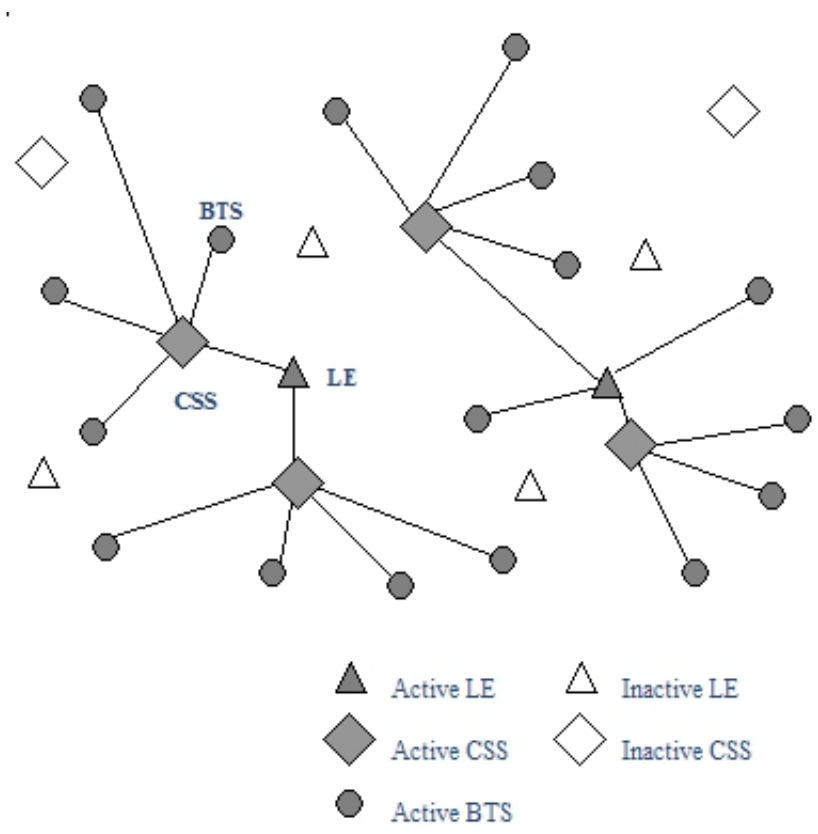

Figure 1: UMTS network structure

The binary variables of our MIP model include:

$$y_j^{CSS-h} = \begin{cases} 1 & \text{if a CSS of } h \text{ type is installed in location } j \text{ assigned to} \\ & \text{a concentrator installed at location } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_k^{LE} = \begin{cases} 1 & \text{if a LE is installed in location } k \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij}^{BTS-CSS} = \begin{cases} 1 & \text{if a BTS in location } i \text{ is assigned to} \\ & \text{a CSS installed in location } j \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ik}^{BTS-LE} = \begin{cases} 1 & \text{if a BTS in location } i \text{ isassigned to} \\ & \text{a LE installed in location } k \\ 0 & \text{otherwise} \end{cases}$$

$$x_{jk}^{CSS-LE} = \begin{cases} 1 & \text{if a CSS in location } j \text{ isassigned to} \\ & \text{a LE installed in location } k \\ 0 & \text{otherwise} \end{cases}$$

For the sake of brevity, we do not report here the MIP model we work with; the interested reader is referred to [7]. With respect to this formulation, the first-level binary variables (automatically detected by the DRT code) are those concerning the activation of the LE's and CSS's, namely $y_j^{CSS-h} \in \{0,1\} \forall j = 1,..,m; h = 1,2$ and $y_k^{LE} \in \{0,1\} \forall k = 1,\ldots,p$.

4.2 Test bed

Table 1 reports the main characteristics of the MIP instances in our test bed. For each instance we report the number of BTS's, CSS's and LE's, the number of constraints (rows) and variables (columns) in the associated MIP model, and the number of first- and second-level variables found automatically by the DRT method. We also report the cost of the best feasible solution found during the present study (Best), in Euro, the value of the root-node LP relaxation of the MIP model (LB), and their percentage gap (%Gap) computed as $100(Best - LB)/Best$.

percentage gap

NAME	BTS	CSS	LE	n. rows	n. cols	$\|\mathcal{B}_1\|$	$\|\mathcal{B}_2\|$	Best	LB	%Gap
UMTS-1	50	4	2	473	334	10	308	10,760,529	10,646,775	1.06
UMTS-2	55	6	3	681	498	13	455	11,595,031	11,554,350	0.35
UMTS-3	60	5	4	809	614	12	562	12,393,173	12,339,360	0.43
UMTS-4	65	6	4	972	738	14	676	13,376,204	13,317,114	0.44
UMTS-5	100	8	4	1741	1316	20	1232	19,731,006	19,705,894	0.13
UMTS-6	105	9	5	2119	1628	23	1515	21,950,121	20,570,225	6.29
UMTS-7	110	9	6	2394	1836	24	1704	22,969,378	21,610,413	5.92
UMTS-8	115	10	6	2660	2046	26	1900	23,978,221	22,701,546	5.32
UMTS-9	150	12	6	3739	2946	30	2772	30,089,580	29,177,750	3.03
UMTS-10	155	13	7	4376	3406	33	3191	31,081,092	30,259,843	2.64
UMTS-11	160	13	8	4710	3706	34	3464	31,984,753	31,234,374	2.35
UMTS-12	165	14	8	4984	4002	36	3742	32,834,623	32,063,919	2.35
UMTS-13	175	15	8	6428	4424	38	4145	34,711,383	34,104,825	1.75
UMTS-14	185	17	9	7034	5313	43	4963	36,580,069	36,130,049	1.23
UMTS-15	200	20	10	8271	6650	50	6200	39,181,447	38,855,799	0.83
UMTS-16	215	22	10	10051	7595	54	7100	41,905,498	41,779,804	0.30
UMTS-17	225	24	10	11217	8429	58	7890	45,038,257	43,670,898	3.04
UMTS-18	235	25	10	11890	9036	60	8475	46,861,037	45,594,865	2.70
UMTS-19	250	26	10	13227	9843	62	9260	49,613,181	48,536,040	2.17
UMTS-20	300	28	10	16213	12307	66	11680	58,663,150	58,046,858	1.05

Table 1: Test bed characteristics.

4.3 Preliminary analysis

Table 2 is taken from Polo [19] and provides a comparison between the very preliminary implementation of the DRT method presented in [19] and three alternative heuristics: the problem-specific TS algorithm described in [14], a general-purpose VNS scheme based on ILOG-Cplex 7.0 (as described in [19]), and the general-purpose MIP solver ILOG-Cplex 7.0 (with its MIPEmphasis parameter set to 1 for better heuristic performance). It should be stressed that the problem-specific TS heuristic used in the comparison is a quite sophisticated one, and its implementation required about 6,000 lines of C++ code. The other two codes are instead more general, and required less than 2,000 lines of code each (of course, not including the MIP solver).

The preliminary test was performed on a PC Pentium III/350MHz with 128MB RAM, Microsoft Visual C++ 6.0 compiler, Windows 98 OS.

The table reports the objective function *improvement* of the DRT method when compared with an alternative heuristic, say HEU, computed as:

$$\delta := HEU\_objective\_value - DRT\_objective\_value$$

As we are considering minimization problems, a positive value means that the DRT method performed better than the alternative heuristic HEU.

	VNS	Cplex 7.0	Tabu Search
UMTS-1	0	0	0
UMTS-2	0	0	0
UMTS-3	-5,734	-2,471	-3,829
UMTS-4	0	0	68,650
UMTS-5	5,885	0	271,924
UMTS-6	1,745	2,110	201,801
UMTS-7	1,853	291	129,522
UMTS-8	1,940	19,311	21,980
UMTS-9	49,546	53,975	50,823
UMTS-10	40,918	1,660	56,396
UMTS-11	37,251	47,660	72,439
UMTS-12	23,195	33,629	63,288
UMTS-15	64,462	99,748	59,173

Table 2: Solution improvements obtained through a preliminary implementation of the DRT method with respect to VNS, ILOG-Cplex 7.0, and TS. A positive entry indicates a better performance of the DRT method.

The table shows that, even in its preliminary implementation, the DRT method compares very favorably with the compared heuristics, and in particular with the problem-specific TS method.

4.4 Computational analysis

Table 3 compares the performance of our DRT method with four alternative heuristics: (1) the Local Branching method with the UMTS parameter setting suggested in [6]; (2) the MIP solver ILOG-Cplex 7.0 with its MIPEmphasis parameter set to 1 so as to deliver improved heuristic solutions; (3) the DRT variant DRT2 described at the end of Section 2; and (4) the DRT method with no first-level variables. The first-level neighborhood parameter in (8) was set to $k_1 = 0$, i.e., all the first-level variables are fixed during each refining phase. More details on the parameters used in the experiments are reported in the Appendix.

The table reports the absolute improvement of DRT over each alternative heuristic, namely $\delta := HEU\_objective\_value - DRT\_objective\_value$. In addition, we report the percentage DRT improvements with respect to the initial MIP gap, computed as:

$$\delta\ \% := \frac{100\ \delta}{DRT\_objective\_value - LB}$$

where LB is the root-node MIP lower bound reported in Table 1. As before, a positive δ (or δ %) means that DRT performed better than the alternative algorithm.

	Cplex 7.0		**DRT2**		**DRT** ($\|\mathcal{B}_1\| = 0$)		**LocBra**	
problem	δ	δ %	δ	δ %	δ	δ %	δ	δ %
UMTS-1	455	0.40	0	0.00	895	0.79	0	0.00
UMTS-2	24,786	60.93	200	0.49	4,567	11.23	7	0.02
UMTS-3	0	0.00	0	0.00	3,940	7.32	0	0.00
UMTS-4	1,814	3.07	0	0.00	4,158	7.04	22,327	37.78
UMTS-5	729	2.90	6,000	23.89	7,126	28.38	491	1.96
UMTS-6	0	0.00	0	0.00	0	0.00	76,846	5.57
UMTS-7	3,994	0.29	288	0.02	4,393	0.32	818	0.06
UMTS-8	12,399	0.97	110	0.01	53,828	4.22	45,263	3.55
UMTS-9	45,625	4.98	-3,588	-0.39	104,057	11.37	301,436	32.93
UMTS-10	13,443	1.64	712	0.09	86,229	10.50	328,667	40.02
UMTS-11	79,412	10.56	-1,532	-0.20	120,047	15.97	135,307	18.00
UMTS-12	51,678	6.64	-7,904	-1.02	109,940	14.12	125,391	16.10
UMTS-13	61,417	10.13	3,821	0.63	139,919	23.07	53,885	8.88
UMTS-14	2,281	0.51	3,634	0.81	97,169	21.59	29,244	6.50
UMTS-15	100,973	31.01	2,042	0.63	276,331	84.86	159,167	48.88
UMTS-16	191,918	126.28	-26,284	-17.29	330,300	217.33	146,954	96.69
UMTS-17	125,304	8.88	-44,239	-3.13	220,255	15.60	140,390	9.95
UMTS-18	110,685	8.59	-22,878	-1.77	684,995	53.14	264,419	20.51
UMTS-19	67,965	6.24	-11,454	-1.05	623,342	57.26	111,361	10.23
UMTS-20	429,981	69.73	-369	-0.06	882,212	143.06	548,595	88.96

Table 3: Solution improvements of DRT w.r.t. ILOG-Cplex 7.0, DRT2, DRT with no first-level variables, and Local Branching. A positive entry indicates a better performance of the DRT method.

According to the table, the DRT and DRT2 methods outperformed the other heuristics on all the instances of the test bed. The DRT and DRT2 methods can be considered almost equivalent, in that none of the two dominates clearly the other, though DRT2 seems to be preferable for large instances. As expected, the DRT method with no first-level variables exhibits a quite poor performance.

A comparison between ILOG-Cplex 7.0 and Local Branching shows that the latter tends to produce worse solutions. A similar behavior for UMTS network design problems was observed in [6]. Evidently, the default Local Branching scheme in not particularly suited for this kind of MIP instances. An explanation is that this scheme, in its generality, makes no distinction between first- and second-level variables, hence it cannot define very effective neighborhoods for UMTS instances. As a matter of fact, the unsatisfactory performance of the original Local Branching scheme on UMTS instances gave us motivation to investigate a Local Branching variant capable of detecting automatically (and exploiting) certain relevant MIP structures, thus originating the research reported in the present paper.

To better highlight the behavior of the five heuristics we compared, and in particular their capability of providing soon good heuristic solutions, we plot in Figure

2 the heuristic solution values they produced for instance UMTS-13 (for Cplex, the monotonically non-increasing value of the incumbent solution is reported).

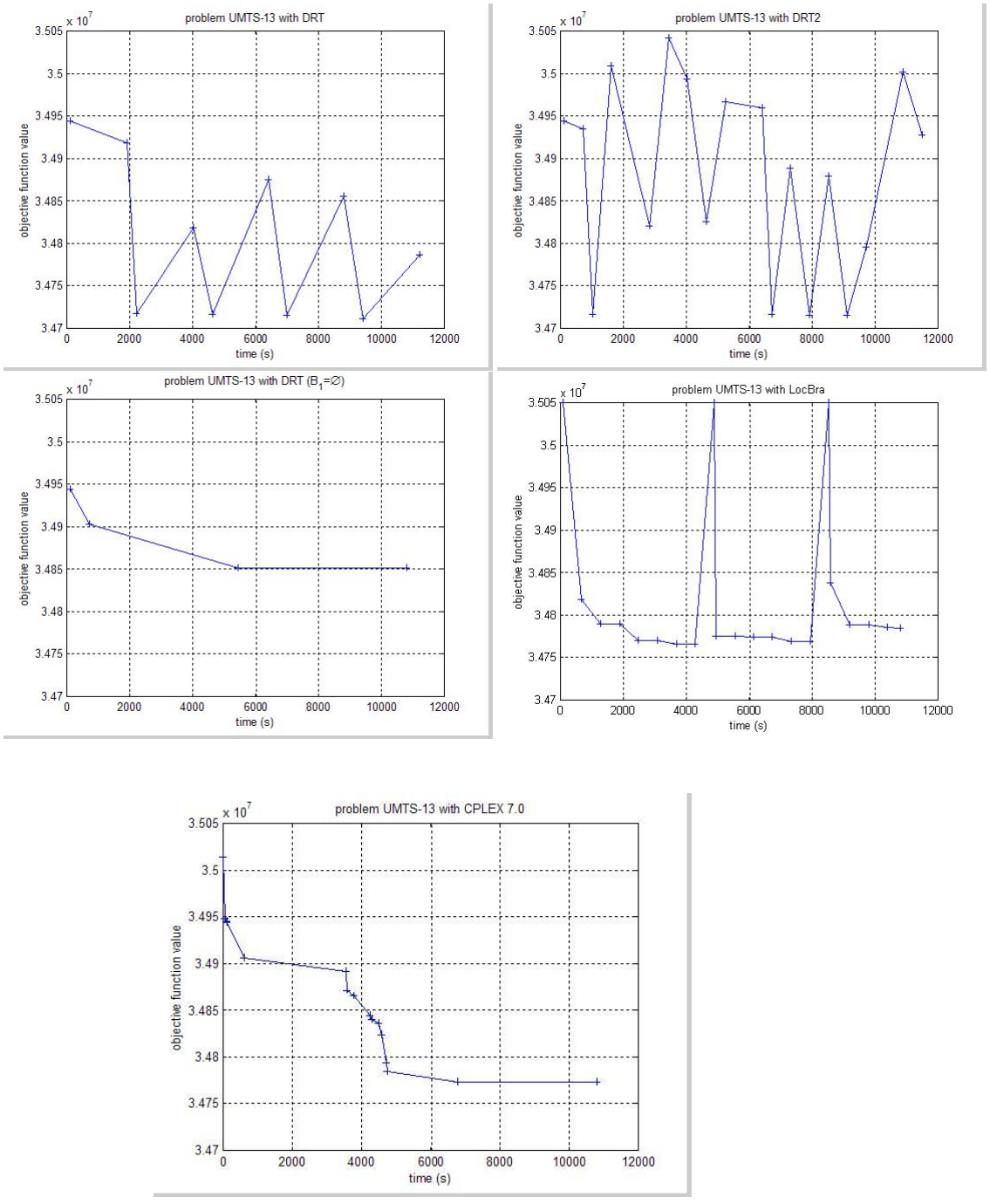

Figure 2: Heuristic trends when solving instance UMTS-13.

5 Conclusions

Many real-world optimization problems are difficult to solve. Very often, these problems are first approached within the MIP framework, and (sometimes quite sophisticated) models are designed in the hope they can be solved to proven optimality–or within an acceptable error–by a commercial MIP solver. If this is not the case, one often forgets about the developed MIP models and concentrates on ad-hoc local-search heuristics, often based on the enumeration of solution neighborhoods defined by a series of more and more sophisticated problem-specific "moves".

In this paper we have investigated a different approach, where the MIP model plays an important role even in the heuristic phase. This is motivated by the fact that commercial MIP solvers are nowadays quite stable and powerful tools, that we would like to exploit even when they prove not adequate for the solution of the original MIP *as a whole*. A key point of our method is that we use a local-search paradigm where the solution neighborhoods are defined by suitable "locality" linear constraints to be added to the original MIP model, so we can invoke an external MIP solver for their exploration. This allows for a tight and fruitful cooperation between the local-search framework and the MIP solver.

We have proposed a simple-to-implement local search scheme that we called Diversification, Refining, and Tight-refining (DRT). The new approach is particularly suited for those MIPs in which the set of binary variables can be partitioned into two sets (called levels), with the property that fixing the value of the first-level variables produces a easier-to-solve (but nontrivial) subproblem.

Our method detects automatically the presence in the MIP model of the first-level binary variables, if any, according to simple heuristic criteria. This information is then exploited during the intensification phase of the local search, so as to explore nested solution neighborhoods defined by local branching cuts constraining the variation of the first- and second-level variables in a different way. In particular, during the refining phases the first-level variables are (almost) fixed to their current value in a certain reference solution, so as to allow for an effective redefinition of all other variables.

Computational results on a hard UMTS telecommunication network design problem are presented. The outcome is that, on our test-bed, the DRT method clearly outperforms other heuristics from the literature, including an ad-hoc (and quite sophisticated) implementation of a classical tabu-search method based on a series of problem-specific moves.

Future research should investigate the effectiveness of the DRT method on other network design problems. Also to be investigated is the possibility of implementing more powerful procedures to detect first-level variables in a more general setting.

6 Acknowledgements

Work partially supported by M.I.U.R. and C.N.R., Italy, and by the EU project DONET. Special thanks are due to Andrea Lodi who provided us with his implementation of the Local Branching framework.

References

[1] E. Amaldi, A. Capone and F. Malucelli. Planning UMTS Base Station Location: Optimization Models with Power Control and Algorithms. Tehcnical Report, University of Milano, 2002

[2] E. Balas, S. Ceria, M. Dawande, F. Margot and G. Pataki. OCTANE: A New Heuristic For Pure 0-1 Programs. Operations Research 49(2), 207–225, 2001.

[3] A. Balakrishnan, T.L. Magnanti and P. Mirchandani. Network Design, in: M. DellAmico, F. Maffioli, S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, New York, 311–334, 1997.

[4] E. Balas and C.H. Martin. Pivot-and-Complement: A Heuristic For 0-1 Programming. Management Science 26(1), 86–96, 1980.

[5] Cplex. *ILOG Cplex 7.0 User's Manual and Reference Manual*. ILOG, S.A., 2001 (http://www.ilog.com)

[6] M. Fischetti and A. Lodi. Local Branching. Mathematical Programming Ser. B, 98, 23-47, 2003.

[7] M. Fischetti, G. Romanin Jacur and J.J. Salazar Gonzáles. Optimization of the Interconnecting Network of a UMTS Radio Mobile Telephone System. European Journal of Operational Research, 144, 56-67, 2003.

[8] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part I. Journal of Heuristics 2, 343–358, 1997.

[9] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part II. Journal of Heuristics 3, 161–179, 1997.

[10] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publisher, Boston, Dordrecht, London, 1997.

[11] F. S. Hillier. Effcient Heuristic Procedures For Integer Linear Programming With An Interior. Operations Research 17(4), 600–637, 1969.

[12] T. Ibaraki, T. Ohashi and H. Mine. A Heuristic Algorithm For Mixed-Integer Programming Problems. Mathematical Programming Study 2, 115–136, 1974.

[13] M. Labbé and F.L. Louveaux. Location Problems, in: M. DellAmico, F. Maffioli, S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, New York, 261-281, 1997.

[14] A. Laspertini. *Third-generation Mobile Telephone Systems: Optimal Design of the Interconnecting Network.* Master Dissertation, University of Padova, Italy, 1997 (in Italian).

[15] A. Løkketangen. Heuristics for 0-1 Mixed-Integer Programming. In P.M. Pardalos and M.G.C. Resende (ed.s) *Handbook of Applied Optimization*, Oxford University Press, 474–477, 2002.

[16] A. Løkketangen and F. Glover. Solving Zero/One Mixed Integer Programming Problems Using Tabu Search. European Journal of Operational Research 106, 624-658, 1998.

[17] N. Mladenovíc and P. Hansen. Variable Neighborhood Search. Computers and Operations Research 24, 1097–1100, 1997.

[18] M. Nediak and J. Eckstein. Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming. Research Report RRR 53-2001, RUTCOR, Rutgers University, October 2001.

[19] C. Polo. Heuristic Methods for the Optimal Design of the Interconnecting Network in a UMTS Radio Mobile Telephone System. Master Degree thesis, University of Padova, 2002 (in Italian).

[20] H. Yaman. Concentrator location in telecommunication network. Master Dissertation, University of Bruxelles, 2002

A Appendix: the DRT and DRT2 pseudo code

We next outline a possible pseudo-code of the DRT algorithm and of its variant DRT2 (where big diversification steps are allowed).

The main function DRT has been divided in four subroutines. In the main while loop we alternate the big diversification, diversification, refining and tight refining phases. Diversification and big diversification are basically the same–they only differ for the diversification parameters k's, which become k^{big}'s in case of a big diversification. The termination condition is based on the time limit and on the maximum number of diversifications and big diversifications allowed.

Function `solve(parms)` represents a call to the external MIP solver (e.g., ILOG-Cplex); the argument `parms` represents a set of parameters passed to the solver (such as time limit, MIP emphasis, etc.). The function returns the best feasible solution found within the given termination conditions (time limit, etc.); in case no solution if found, the function returns a dummy *empty* solution.

All parameters can be set by the user, through an appropriate parameter file, and include:

name	**meaning**
time_limit	maximum computational time for the whole DRT
max_big_div_number	maximum number of big diversifications allowed
max_div_number	maximum number of diversifications allowed
max_div_with_no_improv	maximum number of diversifications without improvement before performing a big diversification
percgap	percentage gap used to check whether a solution leads to a significant improvement
$k_1^{ini}(TYPE)$	starting value for the $TYPE$ (diversification or big diversification) parameter $k_1^{min}(TYPE)$ used in constraint (10)
$k_1^{step}(TYPE)$	step-size for $k_1^{min}(TYPE)$
k_1	used in the refining constraint (8)
k_2^{step}	neighborhood-size step used during the tight refining phase
k_2^{max}	maximum neighborhood-size allowed during the tight refining phase

Additional input parameters include the time limit for each phase (initial search, diversification, big diversification, refining, tight refining), plus some extra parameters possibly required by the MIP solver.

```
function DRT(DRT_parms)
current_solution = solver(start_parms)
if current_solution is optimal then
  output(current_solution)
  return
end if
if current solution is empty then
  output("no starting solution can be found")
  return
end if
num_div=0; num_div_with_no_improv=0
while (execution_time < time_limit) and (num_div < max_div_number) and
(num_big_div < max_big_div_number) do
  add the tabu constraint (11) with respect to the current solution
  if num_div_with_no_improv==max_div_with_no_improv then
    num_div_with_no_improv=0
    num_big_div++
    DIVERSIFICATION(current_solution,BIG)
  else
    DIVERSIFICATION(current_solution,NO_BIG)
  end if
  abort execution if time limit exceeded
  REFINING(current_solution)
  if  current_solution is not guranteed to be optimal for its neighborhood then
    TIGHT-REFINING(current_solution)
  end if
  update best_solution
  remove the refining and tight refining constraints (8) and (9), respectively
  if best_solution is percgap percent better than the one of the previous diver-
  sification then
    num_div_no_improv=0; num_div_no_improv=0
  else
    num_div_no_improv++
  end if
end while
output(best_solution)
```

function DIVERSIFICATION(current_solution, TYPE)
$k_1^{min} = k_1^{ini}(TYPE)$
repeat
  $k_1^{max} = k_1^{min} + k_1^{step}(TYPE)$
  add constraint (10) for diversification with respect to current_solution
  current_solution = solve(diversif_parms)
  $k_1^{min} = k_1^{min} + k_1^{step}(TYPE) + 1$
  remove the constraint for diversification
  num_div++
until (execution_time=time_limit) **or** (current_solution is not empty)
update best_solution

function REFINING(current_solution)
add constraint (8) for refining with respect to current_solution
current_solution = solve(refining_parms)
if current_solution is guaranteed optimal for the current neighborhood **then**
  remove the constraint for refining
end if

function TIGHT-REFINING(current_solution)
$k_2 = 0$
repeat
  add constraint (9) for tight refining with respect to current_solution
  new_solution = solve(tightref_parms)
  if new_solution is better than current_solution **then**
    current_solution=new_solution
    $k_2 = 0$
  else
    $k_2 = k_2 + k_2^{step}$
  end if
  remove the tight refining constraint
until (execution_time > time_limit) **or** ($k_2 > k_2^{max}$)

B Appendix: parameter setting

We next report the parameter files we used for the computational tests.

B.1 DRT setting

```
  Total time limit (hours) :  3
  Maximum number of diversifications :  1000

# Global algorithm Cplex parameters
```

```
   Tree memory limit :  128
   Node storage file indicator :  1

# Parameters for initial general research
   Time limit (minutes) :  60
   Maximum number of nodes :  5000000
   Precedence to optimality (0) or to feasibility (1) :  1
   Which number of admissible solutions before stopping :  4

# Parameters for diversification
   Time limit (minutes) :  30
   Maximum number of nodes :  5000000
   Minimum number of level-1 variable changes of status :  1
   Step increment :  2

# Parameters for refining
   Time limit (minutes) :  5
   Maximum number of nodes :  5000000

# Parameters for tight refining
   Time limit (minutes) :  3
   Maximum number of nodes :  5000000
   Abs tolerance on gap between best solution and lower bound :  1e-009
   Rel tolerance on gap between best solution and lower bound :  1e-009
   Minimum number of level-2 variable changes of status :  1
   Maximum number of level-2 variable changes of status :  4
   Step increment :  2
```

B.2 DRT2 setting

```
   Total time limit (hours) :  3
   Maximum number of diversifications :  10000
   Maximum number of big diversifications :  4

# Parameters for big diversification
   Maximum number of diversifications without improvement :  5
   Minimum Gap of improvement (percentage 0-100) :  5
   Minimum number of level-1 variable changes of status :  5
   Step increment :  2

# Global algorithm Cplex parameters

Tree memory limit :  128
   Node storage file indicator :  1
```

```
# Parameters for initial general research
   Time limit (minutes) :  60
   Maximum number of nodes :  5000000
   Precedence to optimality (0) or to feasibility (1) :  1
   Which number of admissible solutions before stopping :  4

# Parameters for big and small diversification
   Time limit (minutes) :  10
   Maximum number of nodes :  5000000
   Minimum number of level-1 variable changes of status :  1
   Step increment :  2

# Parameters for refining
   Time limit (minutes) :  5
   Maximum number of nodes :  5000000

# Parameters for tight refining
   Time limit (minutes) :  3
   Maximum number of nodes :  5000000
   Abs tolerance on gap between best solution and lower bound :  1e-009
   Rel tolerance on gap between best solution and lower bound :  1e-009
   Minimum number of level-2 variable changes of status :  1
   Maximum number of level-2 variable changes of status :  4
   Step increment :  2
```

B.3 Local Branching setting

```
Total time limit (hours):  3
Node time limit (seconds) :  600
MIP emphasis:  1 (ILOG-Cplex emphasis on feasibility)
MIP presolve:  1 (do apply ILOG-Cplex presolver)
MIP heuristic frequency :  10 (ILOG-Cplex heuristics applied at each 10th
node)
MIP precision:  0 (ILOG-Cplex default precision)
MIP print interval :  10
MIP number of time intervals :  100
Local Branching neighborhood size k :  20
Local Branching type of cuts:  0 (default=symmetric local branching constraints)
Local Branching exact/heuristic flag:  0 (heuristic version)
Video output:  1 (yes)
```