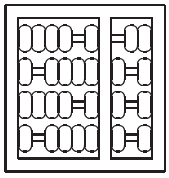

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Hybrid Lagrangian algorithms for optimal vertex separators

Victor F. Cavalcante *Cid C. de Souza*

Technical Report - IC-09-09 - Relatório Técnico

March - 2009 - Março

Abstract

In this paper we propose a Lagrangian relaxation framework to solve the vertex separator problem (VSP). This framework is based on the development of relax-and-cut algorithms which embed the separation of valid inequalities for the VSP discussed in [3] in the subgradient method. These relax-and-cut algorithms are then used as a preprocessing phase in a hybrid algorithm which combines them with branch-and-cut algorithms proposed in [12]. This is done basically by feeding the branch-and-cut algorithms not only with the primal bound but also the cuts separated during the preprocessing phase. Computational results obtained with benchmarks from the literature showed that the hybrid algorithm developed here outperforms the best exact algorithm available for the VSP to date.

Keywords: Lagrangian relaxation, cutting planes, Integer Programming, relax-and-cut algorithms, vertex separator.

1 Introduction

A *vertex separator* in an undirected graph is a subset of the vertices, whose removal disconnects the graph in at least two nonempty connected components. Recently, Balas and de Souza [3, 12] studied the vertex separator problem (VSP) which can formally be stated as follows.
`INSTANCE`: a connected undirected graph $G = (V, E)$, with $|V| = n$, an integer $1 \leq b \leq n$ and a cost c_i associated with each vertex $i \in V$.
`PROBLEM`: find a partition of V into disjoint sets A, B, C, with A and B nonempty, such that *(i)* E contains no edge (i, j) with $i \in A$, $j \in B$, *(ii)* $\max\{|A|, |B|\} \leq b$, *(iii)* $\sum_{j \in C} c_j$ is minimized.

The sets A and B are called the *shores* of the separator C. A separator C that satisfies (i) but violates (ii) is termed *infeasible*; one that satisfies (i) and (ii) is *feasible*; and a separator that satisfies (i), (ii), (iii) is optimal. Unless otherwise specified, the term separator is used here to denote a feasible one. The VSP is $\mathcal{NP}$-hard and has widespread applicability in network connectivity. Further discussion on applications appears in [3].

In that paper Balas and de Souza also conducted the first polyhedral investigation on the VSP. They introduced several classes of strong valid inequalities for the polytope associated to the problem. In a companion paper to that study, the same authors reported extensive computational experiments with a branch-and-cut (B&C) algorithm based on those inequalities. In [6] Borndörfer *et al* considered a generalization of the VSP where the partitioning of the vertex set has to be done in two or more subsets. However, contrarily to the VSP, solutions where one of the shores remains empty are allowed.

Based on the Integer Programming (IP) model and on the strong valid inequalities introduced by Balas and de Souza, we propose an algorithm that combines Lagrangian relaxation with cutting plane techniques to solve the VSP. Our method belongs to a class of Lagrangian relaxation algorithms where constraints of certain families of inequalities may only be explicitly dualized when they become violated at some Lagrangian relaxation solution. These so-called Relax-and-Cut (R&C) algorithms appear as a promising alternative approach to strengthen Lagrangian relaxation bounds as reported in several recent works in the literature [7, 16, 17, 18, 19, 20]. These algorithms use a dynamic inequality dualization scheme that renders viable the application of Lagrangian Relaxation to models with an exponential number of inequalities. Indeed, a similar approach for the traveling salesman problem [2] date from the early 80's.

Furthermore, we describe a framework that proposes a hybridization between our R&C algorithm and a modified version of the B&C algorithm presented in [12], to our knowledge, the best exact algorithm available for the VSP. Basically, this hybridization consists in using our R&C as a preprocessing subroutine of the B&C algorithm and we denote it by HYBRID. Similar hybrid approaches were already tried on other optimization problems [9, 10, 11]. However this work presents the first attempt to use it in the exact computation of VSP instances. The experiments conducted here show that different versions of the HYBRID method outperform the B&C algorithm when used alone.

The paper is organized as follows. Section 2 presents the IP formulation for the VSP given in [3, 12] and used here. Section 3 briefly reviews the Lagrangian relaxation technique and the subgradient method (SM) and gives a general description of R&C algorithms. The elements of the R&C algorithm we developed for the VSP are presented in Section 4. This section includes details of the Lagrangian relaxations considered, descriptions of the separation routines implemented and of the primal heuristic we devised. Section 5 discusses how we integrated Lagrangian relaxation with other Integer Linear Programming techniques to design an exact algorithm to solve the VSP. The setup of our test environment is detailed in Section 6. Section 7 describes the structure of our algorithm and reports on the computational results obtained for test instances gathered from the literature. Finally, in Section 8, we draw some conclusions and point out some possible extensions of this study.

2 An IP formulation for the VSP

We describe here the mixed IP formulation presented in [3, 12] on which our Lagrangian relaxation is based. For every vertex $i \in V$, two binary variables are defined: $u_{i1} = 1$ if and only if $i \in A$ and $u_{i2} = 1$ if and only if $i \in B$. For $S \subseteq V$ and $k \in \{1, 2\}$, let $u_k(S)$ denote $\sum(u_{ik} : i \in S)$, and $u(S) = u_1(S) + u_2(S)$. An IP model for the VSP is given by

$$\max \sum_{i \in V} c_i(u_{i1} + u_{i2})$$

$$u_{i1} + u_{i2} \leq 1, \qquad \forall\ i \in V \tag{1}$$

$$u_{i1} + u_{j2} \leq 1,\ u_{j1} + u_{i2} \leq 1, \qquad \forall\ (i,j) \in E \tag{2}$$

$$u_1(V) \geq 1, \tag{3}$$

$$u_2(V) \leq b, \tag{4}$$

$$u_1(V) - u_2(V) \leq 0, \tag{5}$$

$$u_{i2} \geq 0,\ u_{i1} \in \{0, 1\}, \qquad \forall\ i \in V. \tag{6}$$

Inequalities (1) force every vertex to belong to at most one shore. Inequalities (2) prohibits the extremities of an edge to be on distinct shores. Inequalities (3) to (5) limit the size of the shores and, at the same time, reduce the symmetry of the model by forcing the size of shore A to be bounded by that of shore B. As observed in [12], if the u_{i1} variables are integer for all $i \in V$, the integrality of the u_2 variables can be dropped from the formulation. Though this observation is not taken into account by our Lagrangian relaxation, it is relevant for IP solvers.

3 Relax-and-Cut (R&C) algorithms

For completeness, we briefly review the basics on Lagrangian relaxation and relax-and-cut algorithms that are relevant to us. Denote by X a subset of $\mathbb{B}^n = \{0,1\}^n$ and let

$$Z = \max \{cx : Ax \leq b,\ x \in X\} \tag{7}$$

be a formulation for a $\mathcal{NP}$-hard combinatorial optimization problem. In association with (7) one has $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$, where m and n are positive integral values representing, respectively, the number of constraints and the number of variables involved. Let Z' denote the formulation obtained after removing constraints $Ax \leq b$ from (7). Also, assume that Z' can be solved faster than Z (typically in polynomial or pseudo-polynomial time in the problem size).

A Lagrangian relaxation of (7) is obtained by bringing the term $\lambda(b - Ax)$ into the objective function of Z', where $\lambda \in \mathbb{R}^m_+$ is the corresponding vector of Lagrange multipliers. The resulting *Lagrangian relaxation Problem* ($LRP(\lambda)$) is

$$Z(\lambda) = \max \{cx + \lambda(b - Ax) :\ x \in X\} = \max \{(c - \lambda A)x + \lambda b :\ x \in X\}. \tag{8}$$

It is a known fact that $Z(\lambda) \geq Z$ and, therefore, the tightest possible upper bound on Z, attainable through $LRP(\lambda)$, is given by an optimal solution to the *Lagrangian dual problem* (LDP) $Z_D = \min_{\lambda \in \mathbb{R}^m_+} \{\max \{(c - \lambda A)x + \lambda b : x \in X\}\}$. In the literature, several methods exist to compute the LDP. Among these, due to its simplicity and the acceptable results it returns, the subgradient method (SM) is the most widely used [5]. A brief review of that method follows since the R&C algorithm we suggest here for the VSP is deeply based on SM.

SM is an iterative procedure which solves a succession of LRPs like the one in (8). It starts with a feasible vector λ^0 of Lagrangian multipliers and, at iteration k, generates a new feasible vector λ^k of multipliers and an associated LRP. Usually, the algorithm stops when a given limit on the number of iterations is reached.

At iteration k, let $\bar{x}^k$ be an optimal solution to (8) with cost $Z(\lambda^k)$ and let z^k_{LB} be a known lower bound on (7). An associated subgradient vector (for the m relaxed constraints) is then computed as $g^k_i = (b_i - a_i \bar{x}^k),\ i = 1, 2, \ldots, m$. That vector is then used to update λ^k. To that order, a *step size* θ^k is computed. The following formula is commonly applied to perform this calculation [5]

$$\theta^k = \frac{\pi^k (Z(\lambda^k) - z^k_{LB})}{\sum_{i=1}^m (g^k_i)^2}. \tag{9}$$

Typically, the real parameter π^k is set to an initial value (π^0). Along the iterations, it is reduced to a fraction of its current value whenever an *a priori* fixed number of LRPs have been solved without improving the upper bound on Z. Finally, once θ^k is obtained, λ^k is updated as

$$\lambda^{k+1}_i = \max \{0; \lambda^k_i - \theta^k g^k_i\},\ i = 1, 2, \ldots, m. \tag{10}$$

Notice that the straightforward use of formulas (9-10) may become troublesome when a *huge* number of dualized inequalities exist. An alternative may be to modify SM according to the R&C scheme discussed below.

In the literature two strategies to implement R&C algorithms are discussed. They differ, basically, on the moment at which the new inequalities are identified and dualized. In a Delayed

Relax-and-Cut (DR&C), several executions of SM are made. The search for violated cuts is performed solely at the end of each such execution and, if some of them are encountered, they are dualized and a new execution of SM starts. In a Non Delayed Relax-and-Cut (NDR&C), typically a single SM execution is done and cuts are dualized along the iterations as they are found (see [7, 16, 18, 19, 20] for details). In a comparison carried out in [19], NDR&C performed better than DR&C. However, in our work, we decide to implement both strategies in order to compare them in the context of the VSP. Also, we propose a third strategy which combines ideas borrowed from the previous ones. We denote it by *Postponed (non-delayed) Relax-and-Cut* (PR&C). As for NDR&C, in PR&C the cuts are separated at each SM iteration. However, these cuts are not immediately dualized. Instead, they are stored in a buffer. Similarly to what happens in DR&C, the SM is executed several times. In the beginning of each execution, the buffer is emptied and all its cuts are dualized for the next SM round.

Clearly, if there are exponentially many inequalities in (7), the use of traditional Lagrangian relaxation becomes impracticable. Alternatively the R&C scheme proposes a dynamic strategy to dualize inequalities. In this process, one should be able to identify inequalities that are violated by $\bar{x}^k$. To do so, likewise polyhedral cutting-plane generation, a separation problem must be solved at every iteration of SM. Thus, one tries to find at least one inequality violated by the current LRP solution. The inequalities thus identified are candidates to be dualized. It is worth noting that separation problems arising in R&C algorithms may be easier than their polyhedral cutting-plane algorithm counterparts. That applies since LRP normally has integral valued solutions (cf. [20]).

4 Relax-and-cut algorithms for the VSP

Different Lagrangian relaxations can be devised from the formulation given in section 2. During this work we evaluated some of them, always considering the trade-off between two aspects: `(a)` the strength (sharpness) of the resulting Lagrangian dual bounds and `(b)` the difficulty of solving the Lagrangian primal and dual problems, which influence on the amount of computation required to obtain the bounds. With this in mind, we considered three relaxations, all of which can be easily seen to satisfy the integrality property. Then, in all three cases, the best dual bound attainable is equal to the value of the VSP linear programming relaxation. Therefore, what prevailed in our choice of the Lagrangian relaxation to be used was the computational effort involved in solving LRP and LDP.

We decided to start with a simple relaxation where the constraint sets (1) and (2) are dualized by means of the vector multipliers $\lambda \in \mathbb{R}_+^{|V|}$, $\beta^1 \in \mathbb{R}_+^{|E|}$ and $\beta^2 \in \mathbb{R}_+^{|E|}$, respectively. Also, observe that symmetry is not of primary concern for the Lagrangian relaxation. Thus, we consider an alternative IP formulation where the inequalities (3) and (4) are replaced, respectively, by $1 \leq u_l(V) \leq b$, with $l = 1, 2$, and inequality (5) is dropped. Accordingly, the resulting LRP is given by

$$\begin{aligned} \mathrm{LRP}(\lambda, \beta^1, \beta^2) = \max \{ & \sum_{i \in V} (\bar{c}_{i1} u_{i1} + \bar{c}_{i2} u_{i2} + \lambda_i) + \sum_{\substack{(i,j) \in E \\ i<j}} (\beta^1_{i,j} + \beta^2_{i,j}) : u_{kl} \in \{0,1\}, \\ & \forall\, k \in V \text{ and } l = 1,2, \text{ satisfying } 1 \leq u_l(V) \leq b \ \} \end{aligned} \tag{11}$$

where $\bar{c}_{k1} = c_k - \lambda_k - \sum_{\substack{(k,j) \in E \\ k<j}} \beta^1_{k,j} - \sum_{\substack{(i,k) \in E \\ i<k}} \beta^2_{i,k}$ and $\bar{c}_{k2} = c_k - \lambda_k - \sum_{\substack{(i,k) \in E \\ i<k}} \beta^1_{i,k} - \sum_{\substack{(k,j) \in E \\ k<j}} \beta^2_{k,j}$ (for each k in V) are the Lagrangian costs of, respectively, u_{k1} and u_{k2}. Notice that (11) can be

solved in $O(|V|\log|V|)$ time by sorting the variables according to their Lagrangian costs and after performing a few simple calculations.

The second relaxation we experimented with is very similar to the first one, differing only by the fact that inequalities (1) are not dualized anymore. The resulting LRP is thus

$$\begin{aligned}\mathrm{LRP}(\beta^1,\beta^2) = \max\{ &\sum_{i\in V}(\bar{c}_{i1}u_{i1} + \bar{c}_{i2}u_{i2}) + \sum_{\substack{(i,j)\in E\\ i<j}}(\beta^1_{i,j}+\beta^2_{i,j}) : u_{kl}\in\{0,1\},\\ &\forall\, k\in V \text{ and } l=1,2, \text{ satisfying (1)}, 1\le u_l(V)\le b\}\end{aligned} \tag{12}$$

where $\bar{c}_{k1} = c_k - \sum_{\substack{(k,j)\in E\\ k<j}}\beta^1_{k,j} - \sum_{\substack{(i,k)\in E\\ i<k}}\beta^2_{i,k}$ and $\bar{c}_{k2} = c_k - \sum_{\substack{(i,k)\in E\\ i<k}}\beta^1_{i,k} - \sum_{\substack{(k,j)\in E\\ k<j}}\beta^2_{k,j}$ (for each k in V) are the Lagrangian costs of, respectively, u_{k1} and u_{k2}. It is possible to devise a simple dynamic programming algorithm that solves $\mathrm{LRP}(\beta^1,\beta^2)$ in $O(|V|^3)$.

The third relaxation comes from the observation that a matrix formed by the coefficients of the set of constraints described in (1) and (2) is totally unimodular. Thus, when all but these constraints are dualized, the resulting LRP is a well-solved problem that can be computed in polynomial-time using a specialized network flow algorithm or an interior point method for linear programming. Now, given the vectors of Lagrangian multipliers $\theta\in\mathbb{R}^1_+$, $\eta\in\mathbb{R}^1_+$ and $\gamma\in\mathbb{R}^1_+$, the resulting LRP is

$$\begin{aligned}\mathrm{LRP}(\theta,\eta,\gamma) = \max\{ &\sum_{i\in V}(\bar{c}_{i1}u_{i1} + \bar{c}_{i2}u_{i2}) - \theta + \eta b :\\ &u_{kl},\ \forall\, k\in V \text{ and } l=1,2, \text{ satisfy (1), (2) and (6)}\}\end{aligned} \tag{13}$$

where $\bar{c}_{k1} = c_k + \theta - \gamma$ and $\bar{c}_{k2} = c_k - \eta + \gamma$ (for each k in V) are the Lagrangian costs of u_{k1} and u_{k2}, respectively.

Among the three relaxations discussed above, the first one provided the best trade-off between the strength of dual bounds and the computation time required to solve the Lagrangian subproblem. For this reason, it was the one adopted in the final configuration of our relax-and-cut algorithm.

4.1 Classes of valid inequalities and separation problems

The relax-and-cut algorithms developed here are based on two families of valid inequalities introduced by Balas and de Souza in their polyhedral study of the VSP [3]. Inequalities in both families have dominators as part of their support graphs. The first is related to minimal connected dominators and the inequalities belonging to it are called *CD inequalities*. The second family is associated to minimal but not necessarily connected dominators and has its strength increased through a tricky lifting procedure. The latter inequalities are termed *LD inequalities*.

The CD and LD inequalities are described below. In the discussion that follows, P is defined as the convex hull of the integer solutions of the IP model given in section 2, i.e., $P := \mathrm{conv}\{u\in\{0,1\}^{2|V|} : u \text{ satisfies (1)–(6)}\}$. The point $\bar{u} = (\bar{u}_1,\bar{u}_2)$, to which we apply our separation routines, refers to an optimal solution of the LRP currently under consideration. Also, given $G=(V,E)$, for any $S\subset V$, $\mathrm{Adj}(S)$ refers to the set of all vertices in $V\backslash S$ which are adjacent to at least one vertex in S (when $S=\{i\}$ we write $Adj(i)$ to denote $Adj(\{i\})$). Similarly, for a certain $k\in V\backslash S$, we denote $Adj_s(k) := \{i\in S : (i,k)\in E\}$.

```
CD-Separation(G)
1. Construct G_ū = (W, F);
2. Determine n_CC, the number of connected components of G_ū;
3. if n_CC = 1 then                          /* G_ū is connected */
4.   if V ⊆ (W ∪ Adj(W)) then                /* W is a dominator of V */
5.     Turn W into a minimal CD;
6.     return the CD inequality u(W) ≤ |W| − 1;
7.   else return FAIL;          /* no new cut is returned for dualization */
```

Figure 1: Separation routine for CD inequalities.

4.1.1 CD inequalities

Balas and de Souza [3] call a valid inequality for VSP *symmetric* if, for all $j \in V$, the coefficients of the variables u_{j1} and u_{j2} in the inequality are the same. Besides, they show that vertex separators are intimately related to vertex dominators. A vertex dominator is a subset of vertices of the graph such that all the remaining vertices are adjacent to at least one of them. The dominator is said to be connected if the subgraph induced by its vertices is connected. Balas and de Souza then stated the following property: every separator and every connected dominator have at least one vertex in common. From this observation, they derived a class of symmetric inequalities associated with connected dominators, the so-called CD inequalities. If $S \subset V$ is a connected dominator, the CD inequality for S is given by

$$u(S) \leq |S| - 1. \tag{14}$$

Inequality (14) is clearly valid for the VSP polytope P. It is non dominated only if S is *minimal* with respect to vertex removal. Notice that minimality here applies to both the dominance and the connectivity properties. Though necessary and sufficient conditions for CD inequalities to define facets are not known in general, they are shown in [12] to be very effective in computations.

A valuable characteristic of our R&C algorithms is the fast separation routine that looks for violated CD inequalities at $\bar{u}$. A high level description of our procedure is given in Figure 1. The routine starts by constructing the subgraph $G_{\bar{u}} = (W, F)$ of the input graph $G = (V, E)$ which is induced by the vertices $i \in V$ with $\bar{u}_{i1} + \bar{u}_{i2} \geq 1$. It is easy to see that, if W is a dominator and $G_{\bar{u}}$ is connected then the CD inequality associated to W is violated by $\bar{u}$. Unfortunately, the converse is not true in general. It holds when constraints (1) are satisfied, in which case, as cited before, LRP can be solved by dynamic programming. Appendix A presents a thorough discussion regarding the complexity of separating CD inequalities and appendix B describes a dynamic programming algorithm to solve LRP.

Thus, our separation routine can be viewed as a heuristic. Step 5 of the algorithm tries to strengthen the inequality since the minimality of the dominator is a necessary condition for a CD inequality to be facet defining. It checks if the removal of a limited number of vertices preserves the connectivity of the graph induced by W and the dominance property. The separation routine implemented has a worst-case complexity of $O(|V|(|V| + |E|))$. But, in general, the size of minimal connected dominators decreases with graph density and the hardest VSP instances correspond to graphs of relatively high densities. In such cases, the algorithm behaves more like a $O(|V| + |E|)$ algorithm.

In our R&C algorithm the separation procedure is called at every SM iteration. Since we

implemented two greedy ways to obtain minimal CD inequalities, at most two cuts are produced per iteration. Every new cut separated is stored in a *pool* and dualized in a Lagrangian fashion. The relaxation in (11) is then modified to incorporate this constraint. As a result, the term $\sum_{k=1}^{|pool|} \mu_k(|S_k| - 1 - u(S_k))$ is added to the cost function of (11), where $\mu \in \mathbb{R}_+^{|pool|}$ is the vector of multipliers of the CD inequalities that are currently dualized and S_k $(k = 1 \dots |pool|)$ corresponds to the connected dominator associated to the CD inequality at position k in the *pool*.

4.1.2 Conditional (CD) Cuts

According to de Souza and Balas, in [12], for unit costs, one can adapt the separation routine to search for more stringent CD inequalities. These inequalities are valid for all vectors $u \in P$ satisfying $u(V) \geq z_{LB} + 1$, but chop off several feasible solutions with smaller costs. Their usage preserves optimality and is conditioned to the existence of a lower bound z_{LB}. We call them *conditional cuts*, in an analogy to what is done for the set covering problem in [4]. For the VSP, these cuts are obtained computing $\alpha = max\{z_{LB} - b + 1, 1\}$ and searching minimal dominators that cover at least $k = |V| - \alpha + 1$ vertices (k-dominators). Thus, given a lower bound z_{LB} for the optimum, the separation routine can be changed to identify minimal connected k-dominators. Obviously, the interesting situation occurs when $z_{LB} > b$, meaning that not all $|V|$ vertices need to be covered. Conditional cuts are used both in the B&C algorithm in [12] and in the R&C algorithm presented here. In our implementation, conditional CD cuts are considered already along the execution of the R&C algorithm. When a conditional CD cut is identified, it replaces any CD inequality it dominates.

4.1.3 LD inequalities

Let $S \subset V$ be a dominator of V. For $i \in S$, $P(i) = \{k \in V \backslash S : Adj_s(k) = \{i\}\}$ is the set of pendent vertices of i. Also, if S is minimal and $P(i) = \emptyset$, for some $i \in S$, the presence of i in S is needed only to dominate i itself. We call such a vertex a *self-dominator*. Now, take $S \subset V$ a minimal dominator of G, not necessarily connected. Then, the inequality

$$u_1(S) \leq |S| - 1. \tag{15}$$

is trivially valid[1] for the VSP polytope P and is facet defining only under some special conditions, according to the following proposition:

Proposition 4.1 (Balas and de Souza[3]) *The inequality (15), where S is a minimal dominator of G, defines a facet of P if and only if the following conditions are satisfied: (a) $V \backslash S = \bigcup_{i \in S} P(i)$; (b) S contains no self-dominator, and (c) S is an independent set.*

Balas and de Souza [3] propose two forms of lifting the inequality (15) when some of the conditions in proposition 4.1 are not satisfied. In the R&C algorithm designed here, we apply the first lifting devised by them, which alters the coefficients of the variables associated to the assignment of vertices of S to the shore B. It applies when the dominator S is not an independent set. Since the resulting inequalities are associated with minimal dominators and a with a lifting procedure, they were called LD (Lifting Dominator) inequalities.

[1] We assume that $|S| \leq b$, for otherwise (15) would be implied by (4), hence redundant.

Now, let S be a minimal dominator that is not an independent set. Further, let $S_1, S_2, \ldots, S_k$ be the vertex sets of the components of $G[S]$ (the graph induced by S in G) such that $|S_l| > 1, l = 1, \ldots, k$. According to [3], for each component $G[S_l]$, one must build an ordered set of vertices $I_l = \{v_1, v_2, \ldots, v_q\}$ having the following properties: `(c1)` I_l is an independent set of $G[S_l]$; `(c2)` for all $i \in \{2, \ldots, q\}$, v_i is at (edge) distance two from the vertex set $\{v_1, v_2, \ldots, v_{i-1}\}$ and `(c3)` I_l is maximal. Such a set always exists and is usually not unique. Balas and de Souza designed an algorithm to find such a set which computes a spanning tree T_l of $G[S_l]$ as follows.

Initially all the vertices in S_l are unmarked. The algorithm starts by arbitrarily choosing $v \in S_l$ as the root of $T_l = (V_{T_l}, E_{T_l})$ and mark v. Also, all the vertices $w \in Adj(v)$ in $G[S_l]$ and all the edges joining them to v in $G[S_l]$ are put into T_l. Then, for each $w \in S_l \backslash T_l$, $w \in Adj(V_{T_l})$ in $G[S_l]$, the following steps are repeated until all the vertices of S_l have been included in T_l: (i) w is marked and put into T_l by joining it through an edge from $G[S_l]$ to some (arbitrarily chosen) unmarked vertex of T_l; (ii) using edges from $G[S_l]$, add to T_l all the vertices in $(S_l \backslash T_l) \cap Adj(w)$ (the adjacency here is defined over $G[S_l]$).

It is not hard to see that the vertices marked in T_l form an ordered set satisfying the conditions defined earlier for I_l. Moreover, because of the freedom one has to choose the unmarked vertex of T_l to which a newly marked vertex is joined by an edge to T_l, the tree is not unique. Figure 2 shows an example of component $G[S_l]$, along with two distinct ordered sets satisfying conditions `(c1)`, `(c2)` and `(c3)`. The spanning trees corresponding to each of the ordered sets are also depicted. Besides, the marked vertices and their degrees are highlighted.

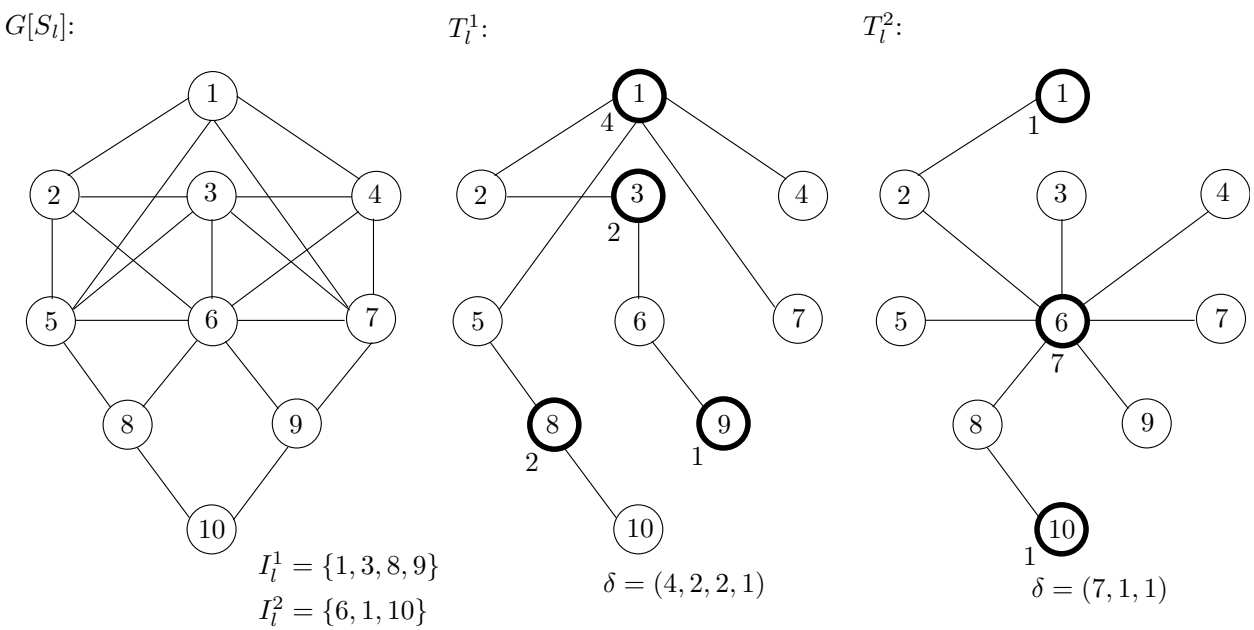

Figure 2: Two ordered sets satisfying conditions `(c1)`, `(c2)` and `(c3)` and the corresponding spanning trees associated with each one of them.

```
LD-Separation(G)
1. Construct G_ū = (W, F);
2. Determine n_CC, the number of connected components of G_ū;
3. if n_CC ≤ 2 then              /* G_ū is connected or has at most two components */
4.   if V ⊆ (W ∪ Adj(W)) then    /* W is a dominator of V */
5.     Turn W into a minimal dominator with n_CC components of G[W];
6.     for l = 1, .., n_CC do
7.       Construct T_l and identify an independent set I_l = {v_1, v_2, ..., v_q} ⊂ V_{T_l};
8.       Determine δ_{v_j} = δ(v_j) for all v_j ∈ I_l. Set δ_{v_j} = 0 for all v_j ∈ V_{T_l} \ I_l;
9.     return the LD inequality u_1(W) + Σ_{v_j ∈ W} δ_{v_j} u_{v_j 2} ≤ |W| − 1;
10. return FAIL;                 /* no new cut is returned for dualization */
```

Figure 3: Separation routine for LD inequalities.

Suppose that the algorithm above executed for each component $G[S_l]$ resulting in a spanning tree T_l and an ordered set I_l. Balas and de Souza [3] prove that the inequality

$$u_1(S) + \sum_{j \in S} \delta_j u_{j2} \leq |S| - 1 \tag{16}$$

is valid and facet defining for the polytope P, where each δ_j is equal to the degree of v_j in T_l if this vertex is marked and is null otherwise.

To identify LD inequalities we implemented a heuristic separation routine which uses this first lifting procedure from Balas and de Souza. The procedure is detailed in Figure 3. Like in the CD separation, the routine starts by constructing the subgraph $G_{\bar{u}} = (W, F)$ of G which is induced by the vertices $i \in V$ with $\bar{u}_{i1} + \bar{u}_{i2} \geq 1$. Then, to save computation time, in step 3 we restrict the separation routine to the cases where $G_{\bar{u}}$ has at most two connected components. Though restrictive, this procedure allows us to generate LD cuts both for connected and non connected dominating sets, contrarily to the CD inequality case. Thus, the lifting of variables in the smaller set is produced with the aid of at most two spanning trees (steps 6–8). In our final implementation, the selection of vertices to mark and to connect each newly marked vertex is done in increasing order of vertex labels. It is worth mentioning that we also experimented to select the vertex with the highest degree in $G[S_l \backslash V_{T_l^{(i)}}]$. In principle this may generate LD constraints with smaller supports resulting in lighter LPs. However, since no actual gain was observed and some additional computation was required, this strategy was abandoned. Finally, a LD inequality associated to the dominator W that cuts off $\bar{u}$ is built. The computational complexity of the separation routine for LD inequalities is the same as that of the CD inequalities, i.e., $O(|V|(|V| + |E|))$. As said before, in practice, for dense graphs, the separation routine is quite fast because minimal dominators are obtained from already small W dominators.

Similarly to what is reported by de Souza and Balas in [12], CD inequalities showed, experimentally, to be much more effective than LD cuts. Moreover, we noticed that the LD inequalities over connected dominator often produced better dual bounds than those over not connected dominators. Thus, in our final experiments, we decided to separate LD inequalities only when our CD separation is turned on and just considering connected dominators. In our experiments, these choices resulted in time savings during LD separation since they reduced its execution to steps 1, 2 and, for $n_{CC} = 1$, to steps 7 to 9 of the algorithm in Figure 3.

In our R&C algorithm the LD separation routine is called at every SM iteration. It produces at most two cuts per iteration and the lifting procedure is called even when the basic LD inequality (15) is not violated by the solution of the current Lagrangian subproblem.

Every new cut separated is stored in a pool and dualized in a Lagrangian fashion. The relaxation in (11) is then modified to incorporate this constraint. As a result, the expression $\sum_{k=1}^{|pool|} \varphi_k(|S_k| - 1 - u_2(S_k) - \sum_{j \in S_k} \delta_j u_{j1})$ is added to the cost function of (11), where $\varphi \in \mathbb{R}_+^{|pool|}$ is the vector of multipliers of the LD inequalities currently dualized.

Notice that, due to the inequality dualization scheme within relax-and-cut algorithms, the same cut may be repeatedly identified by the separation routines. Managing the cut pools of CD and LD inequalities is quite simple and is restricted to redundancy checks, i.e., a new inequality is inserted only if it is not identical to another inequality already in the pool or in the original formulation. The use of suitable data structures and standard hashing techniques render our implementation of redundancy verification very fast.

4.2 A Lagrangian primal heuristic

The generation of good primal bounds is important for the computation of the step size (9) in the SM and to assess the duality gap along the iterations of the algorithm. In order to compute lower bounds for the VSP, we devise a simple greedy heuristic whose steps are summarized in Figure 4. Initially, the set L containing the vertices that are candidates to be part of the shores is built. This excludes the universal vertices, i.e., those which are adjacent to all the other vertices, which clearly belong to any separator. The heuristic chooses arbitrarily two nonadjacent vertices of L and assigns them to different shores so that, in the end, they will not be empty. It proceeds by assigning vertices to shores, prioritizing the assignments corresponding to the variables with higher weighted Lagrangian costs. The choice of the weighting method is controlled by the parameters $\bar{\rho}(k) \in \{0, 1\}$, where k stands for the shore indices, i.e., $k = 1, 2$. It is implemented by multiplying or dividing the Lagrangian cost of the variable associated to a vertex v by the degree of v, $\delta(v)$, as seen in step 5, and can be distinct for variables associated to the same vertex but to different shores. This allows us to distribute the vertices between the two shores according to their costs and degrees. Since universal vertices are always in a separator, our intuition was that, in an optimal solution, vertices with high degrees are less likely to belong to a shore. This would count if favor of cost division. However, to our surprise, preliminary tests with a subset of instances showed that only multiplying the costs produce slightly better solutions than the other combinations. Hence, in our default setting, we fixed $\bar{\rho}(1) = \bar{\rho}(2) = 1$.

Notice that, in the heuristic, all the assignments of vertices to shores are made so as to maintain the viability and to respect the maximum size of the shores. As a final step, a local search subroutine may be called in an attempt to improve on the solution produced by the heuristic. The decision on whether or not the local search is executed works as follows. Let z be the cost of the current solution and $\gamma(z)$ the number of solutions having cost z found so far throughout the R&C execution. The local search is executed only when $\gamma(z) < \Gamma$, where Γ is a parameter that specifies a limit on the number of improvements trials over solutions having the same cost.

The local search routine is described in Figure 5. It starts by enlarging the current separator C with as many vertices of the shores belonging to its adjacency as possible (steps 1 to 5). Then vertices are transferred from the new separator C' back to the shores in step 6 in an arbitrary order. However, the choice of the destination shore is made so as to increase the chances of future

Lagrangian heuristic $(G = (V, E), c, \overline{c}, \bar{\rho}, \Gamma)$
1. $L \leftarrow V \backslash \{$universal vertices in $G\}$;
2. $v_0 \leftarrow \{$any vertex in L that maximizes $\bar{c}(u_{i1})\}$;
3. Initialize shore A: $A \leftarrow \{v_0\}$, $L \leftarrow L \backslash \{v_0\}$ and $L' \leftarrow L \backslash \mathrm{Adj}(v_0)$;
4. Initialize shore B: $B \leftarrow \{v_1 \in L' : \delta(v_1) \geq \delta(v), \forall v \in L'\}$ and $L \leftarrow L \backslash \{v_1\}$;
5. `for` $k = 1, 2$ `do`:
   `for` all $i \in L$, compute $w_{u_{ik}} \leftarrow \bar{c}(u_{ik}) * [\bar{\rho}(k) * \delta(i) + (1 - \bar{\rho}(k))/\delta(i)]$;
   Let S_k be the list of variables u_{ik} sorted non increasingly by $w_{u_{ik}}$;
   `for all` $j \in Adj(v_{2-k})$ `do` $S_k \leftarrow S_k \backslash \{u_{jk}\}$;
6. `while` $|A| < b$ `or` $|B| < b$ `do`
   $f_1 \leftarrow \{$vertex corresponding to the first variable in $S_1\}$;
   $f_2 \leftarrow \{$vertex corresponding to the first variable in $S_2\}$;
   `if` $\bar{c}(u_{f_1,1}) > \bar{c}(u_{f_2,2})$ `then`
      $A \leftarrow A \cup \{f_1\}$; $S_1 \leftarrow S_1 \backslash \{u_{f_1,1}\}$;
      `for` all $j \in \mathrm{Adj}(f_1)$ `do` $S_2 \leftarrow S_2 \backslash \{u_{j,2}\}$;
   `else`
      $B \leftarrow B \cup \{f_2\}$; $S_2 \leftarrow S_2 \backslash \{u_{f_2,2}\}$;
      `for` all $j \in \mathrm{Adj}(f_2)$ `do` $S_1 \leftarrow S_1 \backslash \{u_{j,1}\}$;
   `if` $|A| = b$, $\bar{c}(u_{f_1,1}) \leftarrow -\infty$; /* avoids new vertices in A */
   `if` $|B| = b$, $\bar{c}(u_{f_2,2}) \leftarrow -\infty$; /* avoids new vertices in B */
7. Compute the separator: $C \leftarrow V \backslash \{A \cup B\}$
8. `if` $\gamma(\sum_{j \in C} c_j) < \Gamma$, `call Local Search`$(G, A, B, C, c)$;
9. `return` (A, B, C)

Figure 4: Lagrangian heuristic.

moves from the separator to the shores. This is evaluated via the simple computations in steps 6.i to 6.m. The overall complexity of the Lagrangian heuristic, including the local search procedure, is $O(|V| \log |V| \times |E|)$.

5 Integrating R&C and B&C

An alternative to be more effective in solving VSP problems to optimality is to devise a hybrid approach that combines Lagrangian relaxation with Integer Linear Programming (IP), in the style suggested in [7]. We denote this hybridization of R&C and B&C algorithms by HYBRID. In such combination, optimization is split in three steps: *(i)* the LR phase, based on our relax-and-cut framework, whose output are pools of valid inequalities and a primal bound; *(ii)* a remodelling phase, where the IP formulation is tightened according to the information gathered during the first phase and, subsequently, *(iii)* the LP phase where a branch-and-cut code is executed over the new IP model. Among the cuts used in this last phase, we include those cuts separated throughout the execution of the R&C algorithm in the initial phase. We call them the *Lagrangian cuts*.

The execution flow of the algorithm is depicted in Figure 6. The two first phases are generically termed as the *preprocessing phase* of our hybrid algorithm. Below we describe the three phases of the hybrid algorithm in more detail.

Local Search (G, A, B, C, c)

/ initializations */*

1. Let A_C be the vertices in A that have neighbors in C;
2. Let B_C be the vertices in B that have neighbors in C;
3. `if` $A = A_C$ `then` $A_C \leftarrow A_C \backslash \{$arbitrarily chosen vertex of A$\}$;
4. `if` $B = B_C$ `then` $B_C \leftarrow B_C \backslash \{$arbitrarily chosen vertex of B$\}$;
5. $A' \leftarrow A \backslash A_C$; $B' \leftarrow B \backslash B_C$; $C' \leftarrow C \cup A_C \cup B_C$;

/ main loop */*

6. `for` every vertex $v \in C'$ `do`:

6.a `if` $|A'| = b$ and $|B'| = b$ `then break`;

6.b `if` $|\mathrm{Adj}(v) \cap A'| \neq \emptyset$ and $|\mathrm{Adj}(v) \cap B'| \neq \emptyset$, `then continue`;

6.c $C' \leftarrow C' \backslash \{v\}$;

6.d `if` $\mathrm{Adj}(v) \subset C'$ `then`

6.e `if` $|A'| = b$ `then` $B' \leftarrow B' \cup \{v\}$;

6.f `else`

6.g `if` $|B'| = b$ `then` $A' \leftarrow A' \cup \{v\}$;

6.h `else`

6.i $n_A \leftarrow 0$; $n_B \leftarrow 0$;

6.j `for` all $w \in \mathrm{Adj}(v)$ `do`

6.k $n_A \leftarrow n_A + |\mathrm{Adj}(w) \cap A|$; $n_B \leftarrow n_B + |\mathrm{Adj}(w) \cap B|$;

6.l `if` $n_A > n_B$ `then` $A' \leftarrow A' \cup \{v\}$;

6.m `else` $B' \leftarrow B' \cup \{v\}$;

6.n `else`

6.o `if` $\mathrm{Adj}(v) \subset A' \cup C'$ and $|A'| < b$ `then` $A' \leftarrow A' \cup \{v\}$;

6.p `else` /* $\mathrm{Adj}(v) \subset B' \cup C'$ */

6.q `if` $|B'| < b$ `then` $B' \leftarrow B' \cup \{v\}$;

7. `if` $\sum_{i \in C} c_i > \sum_{i \in C'} c_i$ `then` $A \leftarrow A'$, $B \leftarrow B'$, $C \leftarrow C'$.

Figure 5: Primal heuristic: the local search procedure

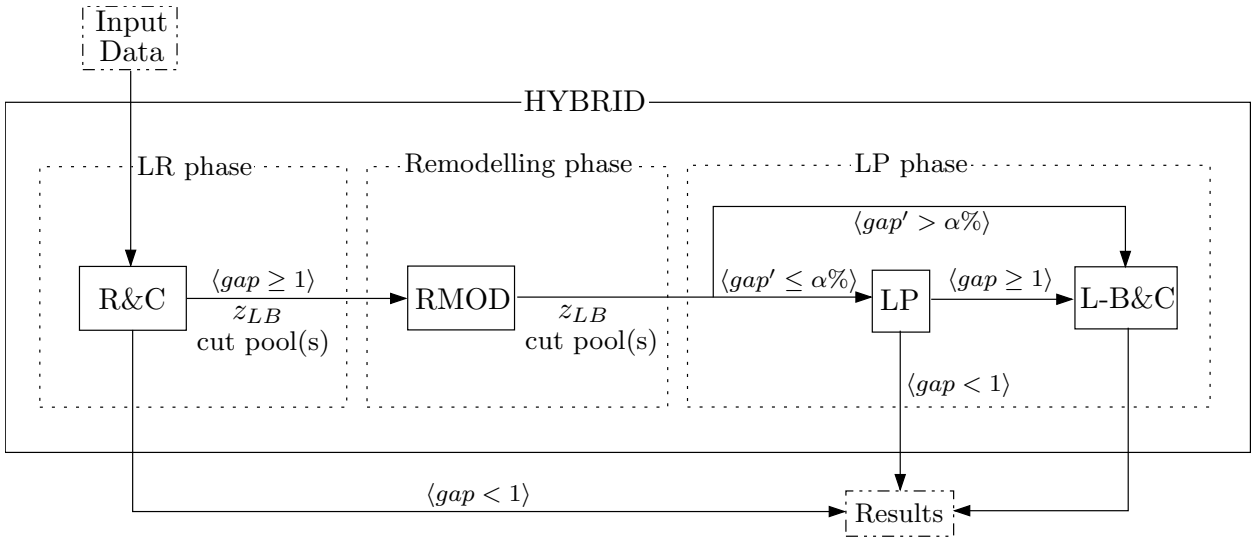

Figure 6: Flow Diagram of the HYBRID algorithms.

5.1 The LR phase

The LR phase is comprised of an R&C module. It corresponds to one of the implementations of the relax-and-cut algorithm described in Section 4 and is the core of our framework. During its execution, valid CD and/or LD inequalities are identified and inserted into the corresponding pool.

After completing the execution of the relax-and-cut algorithm in this module, the final duality gap is verified. If the problem is not solved during the R&C run, some information are passed as the input of the next phases. This includes not only the cut pools, but also the best primal solution and its cost, i.e., the best lower bound found so far.

5.2 The remodelling phase

In addition to CD and LD inequalities identified in the previous phase, some constraints may be added or adapted to strengthen the original formulation presented in section 2.

The first constraint considered comes from the observation that universal vertices must belong to any separator. Thus, given the input graph $G = (V, E)$ and $U = \{i \in V : |Adj(i)| = |V| - 1\}$, the constraint $\sum_{i \in U}(u_{i1} + u_{i2}) = 0$ is trivially valid for the problem. This constraint was not used during the LR phase because it resulted in some degradation of the R&C performance in terms of dual and primal bounds yielded. Also, this constraint is not taken into account in [3]. However, when dealing with high density graphs, the occurrence of universal vertices is very frequent. In practice, the benefits with the addition of this constraint to the IP model justified its inclusion as part of our remodelling phase.

Now, we concentrate on how to use the lower bound yielded by the R&C module to tighten our IP model. To this end, we focus on unit cost instances, i.e., those for which $c_i = 1, \forall i \in V$. We do so because these instances often occur in practical applications.

Assume that z_{LB} is the cost of the best known solution computed in this case. Since $u_2(V) \geq u_1(V)$, we can deduce that $u_2(V) \geq \lfloor \frac{z_{LB}}{2} \rfloor + 1$ must be satisfied by any solution with cost higher than

```
Separation Strategy(sep, CDP, LDP)
    /* call table look up routines to obtain Lagrangian cuts */
1.   runLagrangianSeparation(CD,CDP);          /* L-CD is always ran */
2.   if sep ∈ {⟨L-CD,CD,L-LD,LD⟩, ⟨L-CD,CD,L-LD⟩, ⟨L-CD,L-LD⟩} then
2.a       runLagrangianSeparation(LD,LDP);     /* add L-LD cuts */
    /* test and call de Souza and Balas' routines to generate cuts */
3    if sep ∉ {⟨L-CD,L-LD⟩} and there is no Lagrangian cut violated then
3.a       runCDSeparation();                        /* call CD separation */
3.b      if sep = ⟨L-CD,CD,L-LD,LD⟩ then
3.c            runLDSeparation();                    /* call LD separation */
```

Figure 7: The separation strategy executed in the L-B&C module.

z_{LB}. For any such solution, it is also straightforward to conclude that if $z_{LB} - b > 1$, constraint (3) can be replaced by the stronger inequality $u_1(V) \geq z_{LB} - b$.

Although the previous modifications rely on rather simple arguments, in this phase we incorporate them to the model. As a matter of fact, except for the last change, preliminary experiments we carried out with these modifications in the IP model revealed an improvement in the performance of our modified branch-and-cut algorithm.

5.3 The LP phase.

The LP phase has as its input the cut pools, the best solution and the best primal bound from the LR phase and the new IP model from the remodelling phase. It has two modules that we discuss below.

Linear Programming Solver (LP). This module solves the LP corresponding to the relaxation of the IP model coming from the remodelling phase, appended with the cuts present in the cut pool. This model is computed only if the (relative) Lagrangian gap resulting from the R&C module, given by $gap' = 100 \times (z_{UB} - z_{LB})/z_{LB}$ is lower than a threshold value α. The purpose here is to use linear programming to avoid running the B&C module unnecessarily. It is well-known that, in practice, computing dual bounds within R&C algorithms commonly produce meager values than the linear relaxation optimum value. Thus, this module is a possible workaround to bypass some numerical difficulty in closing the integrality gap.

Branch-and-Cut with Lagrangian cuts (L-B&C). This module runs only if the R&C (LR phase) and/or the LP solver fail to prove optimality (i.e., $gap \geq 1$). Recall that the preprocessing phase yields as outputs the sets of (conditional) CD cuts and/or LD inequalities which are candidates to be added to the formulation given as input of L-B&C. Moreover, L-B&C is also given the values of z_{LB} (best incumbent), which may help to prune the enumeration earlier.

Figure 7 shows the separation strategy adopted at each node of the enumeration tree during the execution of the L-B&C module. This strategy is fixed according to the contents of the ordered sequence denoted by `sep`. The elements of `sep` are taken in the set {`CD, LD, L-CD, L-LD`}. The meanings of these strings are: `CD` and `LD` correspond to the separation routines for CD and LD

inequalities as implemented in [12], while `L-CD` and `L-LD` are the separation routines for Lagrangian cuts implemented by a table look up scheme. This scheme consists basically of algorithms that scan linearly the cut pools, trying to identify (Lagrangian) inequalities that may cutoff the current LP optimal solution. As per this notation, `sep=<L-CD,CD>` means that the separation of a fractional solution is first made by the table look up procedure for Lagrangian CD cuts and then by de Souza and Balas' routine for CD cuts. Lagrangian CD and LD cuts are stored in pools CDP and LDP, respectively.

The calls to `runCDSeparation` and `runLDSeparation` in lines `3.a` and `3.c` of the algorithm refer to the separation routines from [3].

It was experimentally observed that, L-B&C performance is very sensitive to the way in which CD inequalities are added during the branch-and-cut execution. Thus, several experiments were performed in order to determine the maximum amount of cuts to be added per node. The most promising settings took into account the density of the graphs underlying the instances (see Table 1 for details).

6 Test Environment Setup

This section describes the setup of the environment under which our tests were carried out. The algorithms were coded in `C` and `C++`, using resources of the Standard Template Library and prepared to be executed under Linux OS. We used the free compiler `g++` (`gcc` version 4.0.3) with options `-O3` and `-lm` selected. Tests were ran on a Pentium IV machine 2.66 GHz having 1GB of RAM and XPRESS Optimizer 17.01.02 was used as the IP solver.

6.1 Data sets

Our main experiments were made on a subset of instances taken from [12] which can be downloaded from `www.ic.unicamp.br/~cid/Problem-instances/VSP.html`. Additionally, hard instances from the MIPLIB [6] subset were used to perform further tests. Initially, from the more than 140 instances used in [12], we select the ones that required more than a minute of CPU time to be solved by the branch-and-bound (B&B) algorithm of XPRESS in its default configuration. At this point, it is worth noting that XPRESS default configuration implements cut separation routines that would permit us to classify its default algorithm as a branch-and-cut algorithm, rather than as branch-and-bound. However, to distinguish it more easily from the several algorithms we compare throughout our experiments, we will refer to XPRESS default algorithm as being a branch-and-bound (B&B) one.

We end up with 62 instances for our tests, all of which, with cost vector equals to the sum vector. The parameter b delimiting the maximum size of a shore is always set to $\lceil 2n/3 \rceil$ but, for the MIPLIB instances, that value is computed as $\lfloor 1.05 \times n/2 \rfloor$.

The majority of our reports relies on comparing results for instances that were solved by at least one of the algorithms used in our experiments. Thus, among the 62 instances initially selected, only 51 were broadly used in performance comparisons, since 11 instances were not solved by any of the implemented algorithms within the time limit imposed of 30 minutes[2].

A common characteristic of the bulk of these 51 instances is the mid-high ($> 20\%$) density of the graphs underlying them. As already mentioned in [12] and [8], cutting-plane algorithms (especially

[2]Table 8, in appendix C, summarizes some computational results for these 11 not solved instances.

based on CD inequalities) are likely to be more effective for mid-high density graphs. Nevertheless, a few VSP instances arising from low density graphs were kept in our experiments. This allowed us to analyse the behavior of our approach for some hard instances from the MIPLIB benchmark.

As for de Souza and Balas in [12], our results are reported by classes of instances: DIMACS graphs, MATRIXMARKET graphs – divided in three categories, MM-I, MM-II and MM-HD, according to common characteristics used in their construction – and row intersection graphs corresponding to coefficient matrices of some of the MIPLIB instances. Moreover, within each class, the instances are listed in increasing order of graph density.

6.2 Parameter settings

General parameters. The following settings were used for the basic parameters of the subgradient algorithm: (a) the Lagrangian heuristic is called at every SM iteration; (b) the local search heuristic is called just after Lagrangian heuristic execution. However, along the SM execution, the maximum number of improvement trials for solutions with same cost (Γ) was limited to 5. Notice that cost repetition is easily identified in our case since there are only $O(|V|)$ possible values for the cost function; (c) the algorithm stops when the limit of 2000 SM iterations is reached or when $\pi^k \leq 10^{-5}$ in equation (9), whatever occurs first. Moreover, as in [12], the execution time of any algorithm tested in our experiments was limited to 30 minutes.

Algorithm dependent parameters. When the SM is called inside the NDR&C algorithm, π, in equation (9), is initially set to 2 and multiplied by 0.5 each 90 consecutive SM iterations without improvement on the upper bound. Also, the routine responsible for the generation of conditional cuts is called whenever a minimum amount of new CD cuts are added to the pool. In our final tests this upper bound corresponds to 10% of the maximum number of SM iterations. Anyway, provided that a CD inequality is generated along the iterations, we ensure that the routine is called at least once.

When running the PR&C algorithm, however, the Lagrangian dual problem is solved typically several times using SM. We call each complete execution of SM a *pass*. The total number of passes is an input parameter for the postponed relax-and-cut algorithms, denoted by Δ. In our final experiments we adopted $\Delta = 15$. Now, let δ be the number of the current pass. In equation (9), $\pi^0_{(\delta=1)}$ is initially set to 2 and, for the other passes, $\pi^0_{(\delta>1)}$ is computed by the recurrence relation: $\pi^0_{(\delta)} = \pi^0_{(\delta-1)} \times f(\delta)$, where $f(\delta) = 1 - (\delta - 1)/\Delta^2$. Observe that π^0 decreases monotonically and smoothly as δ increases. In our experiments, the small decreases in the initial values of π in equation (9) proved to be beneficial for the computation of tighter dual bounds.

Moreover, along each pass, the π value is update at each 20 consecutive iterations without improvement on the upper bound. Here, similarly to NDR&C strategy, the routine in charge of the generation of conditional cuts is called. In this case, it is done every time a Lagrangian subproblem is solved. Nevertheless, the dualization of inequalities identified along the execution of a pass is done only when the SM terminates.

During L-B&C execution, the amount of CD *Lagrangian* cuts added at each node is mainly determined by the input graph density. Also, it was experimentally observed that adding many cuts at the first node often speeds up the search. Thus after some tuning we ended up with the final configurations displayed in Table 1. Essentially, the graphs were divided into three density ranges and, in each of these groups, we fixed the number of cuts at the root and at the remaining

nodes of the search tree. For instance, when dealing with graphs having density in $(35.6\%, 64.3\%]$, in the first node we put up to 50% of the cuts in the pool. After, for the other nodes, at most 10 `CD` cuts violated are added. In the case of LD inequalities, the amount of *Lagrangian* cuts added at each node followed the tuning used by de Souza and Balas in [12], i.e., 10 cuts per node.

Table 1: Number of Lagrangian `CD` cuts added in the L-B&C algorithm.

Density range	Maximum number of cuts	
	first node	other nodes
$\leq 35.6\%$	10	2
$(35.6\%, 64.3\%]$	$0.5 \times$ pool size	10
$> 64.3\%$	$0.75 \times$ pool size	10

As a final remark, it is worth noting that to determine the settings discussed above, the tunings of the parameter values were carefully performed with a representative subset of instances containing at least one representative of each class.

7 Computational results

In this section we report the computational tests carried out with the several configurations of relax-and-cut algorithms and hybridizations implemented for the VSP.

7.1 Relax-and-cut algorithms: the *preprocessing* phase

The main results of the computational experiments done with the relax-and-cut algorithms developed are documented in Table 2 for the 62 instances selected. Double horizontal lines in these tables split instances from classes DIMACS, `MM-I`, `MM-II`, `MM-HD` and MIPLIB. Also, these tables are divided in five groups of columns. The first group, relative to columns 1–4, describes the instance characteristics: name (`label`), number of nodes (`n`), graph densities (`d`) and the optimum value (`Opt`) or the best known solution value (when it appears underlined). The other four groups of columns report the results, respectively, concerning the non-delayed relax-and-cut (`NDR&C`) and postponed (non-delayed) relax-and-cut (`PR&C`) algorithms developed. These groups have the following format of columns: `ub`, the upper bound obtained; the value of the best solution found (`lb`) and the total time, `t(s)`, required to run each algorithm. Additionally, although not detailed here, some preliminary tests were performed with a delayed relax-and-cut algorithm. However, as suggested by a previous comparison carried out by Lucena ([19]) the results we obtained confirmed that, NDR&C strategies perform better than DR&C ones.

Before analyzing the quality of the dual bounds produced by the Lagrangian methods, let us discuss the linear relaxation bound. In fact, the linear relaxation of the IP model from Section 2 is rather weak. By setting all variables to 1/2 one can satisfy all the constraints provided that b is sufficiently large (which is the case for all instances in our data set). This gives the worst dual bound one could come up with: n ! Thus, poor dual bounds are expected unless strong cuts are added to the formulation. Results reported in [12] show that `CD` inequalities fulfill this requirement. However, a drawback to use such inequalities comes from the fact that the corresponding separation problem is $\mathcal{NP}$-hard in general. The authors had then to resort to a heuristic procedure to perform the task. Their heuristic is of quadratic-time complexity and, in practice, more expensive than

the routine we use to separate integral points which behaves more like a linear-time algorithm (see Section 4.1).

Analyzing the results reported in Table 2 one can see that: (i) in terms of *optimality*, only four instances (with results indicated in bold) have been solved to proven optimality when separating CD inequalities. In this aspect, PR&C seems to have a better performance than NDR&C algorithm; (ii) concerning *dual bounds* we can highlight that: in most of the cases, the algorithms that embed CD inequalities separation produced much stronger dual bounds than LP relaxation bound. Dual bounds produced by NDR&C(LD) and PR&C(LD) are very poor, with values typically near to the linear programming bounds, and are not entered here; (iii) considering the *primal bounds* obtained by our heuristic (lb column), we notice that they have attained the optimum[3] in 65 to 71% of the instances, depending on the relax-and-cut version. Alternatively, if we refer to the best known and extend our analysis to all the 62 instances tested, the rate of success increases a bit further: from 66 to 71%. (iv) the algorithms NDR&C and PR&C cannot be said to dominate one another.

In addition, inspecting the columns corresponding to the total time required by the various configurations we see that, in general, the running times are quite acceptable. Also, in most cases, the use of LD inequalities led to marginal gains and only provoked an increase in CPU time.

As a general remark, contrarily to what happened to other problems, these results do not encourage the application of pure relax-and-cut algorithms to solve VSP instances exactly. However, as shown below, they can be combined with other exact methods in a clever way to form new and efficient algorithms to tackle the problem.

Primal bounds. Though our main focus with the relax-and-cut algorithms was to strengthen the dual bounds, on the primal side, excellent results were achieved. As seen in columns lb of Table 2, in about 70% of the cases our simple Lagrangian heuristic found an optimal solution, with slight variations, depending on the relax-and-cut version. To illustrate the quality of our primal heuristic, consider the results obtained by PR&C(CD,LD) algorithm. In this case, the average error of the heuristic was lower than 1.4% and only for 6 instances this error was higher than 5%. However, the maximum error was 19.3% for miplib.noswot.p, the only instance for which the error exceeded 8.5%.

For a better appreciation of the performance of the Lagrangian heuristic (LR-H), we compare the execution time it spent with the time needed by B&C(CD,LD) primal heuristic (LP-H) to find its best solution. This comparison can be visualized by inspecting the histogram in Figure 8 where, to be able to compare processing times, we restricted ourselves to the 35 cases for which both, LR-H and LP-H, reached a proved optimum. This histogram reveals that LR-H finds optimal solutions much quicker than the LP based heuristic from de Souza and Balas. Besides, it shows that in 80% of the cases, the optimum was found in at most 0.01 seconds and, for all instances, LR-H reached the optimum in at most one second. On the other hand, in 80% (40%) of the cases, LP-H needed at least one (five) second(s) to found an optimum.

7.2 The HYBRID algorithms

Results in Table 2 reveal the good performances of our relax-and-cut algorithms that separate CD inequalities: they often produce good dual and primal bounds rapidly. However, they fail to solve

[3]Entries in column Opt reveal that optimum values are known for 51 of the 62 instances tested.

Table 2: Results for VSP instances: relax-and-cut algorithms NDR&C and PR&C.

Instance				NDR&C (CD)			PR&C (CD)			NDR&C (CD,LD)			PR&C (CD,LD)		
label	n	d	Opt	ub	lb	t(s)	ub	lb	t(s)	ub	lb	t(s)	ub	lb	t(s)
dim.DSJC125.1	125	0.09	90	122	89	4.12	122	88	4.37	122	88	13.23	124	89	11.28
dim.games120	120	0.09	102	121	99	2.14	120	99	1.67	120	99	2.79	120	99	4.64
dim.myciel7	191	0.13	156	193	155	3.71	188	155	3.86	192	153	5.19	188	155	5.84
dim.myciel6	95	0.17	76	90	75	1.43	89	75	1.18	92	73	2.05	89	75	2.59
dim.queen12_12	144	0.25	97	131	97	6.69	131	97	7.57	133	97	21.71	132	97	25.83
dim.queen11_11	121	0.27	81	109	81	5.56	108	81	5.97	110	81	17.76	109	81	20.23
dim.queen10_10	100	0.30	67	88	67	4.04	88	67	4.47	89	67	12.28	89	67	16.74
dim.queen8_12	96	0.30	65	85	65	4.02	85	65	3.69	86	65	13.26	86	65	18.30
dim.queen9_9	81	0.33	55	69	55	2.73	70	55	3.21	70	55	8.84	71	55	13.57
dim.queen8_8	64	0.36	43	53	43	1.67	53	43	2.10	54	43	7.28	54	43	9.53
dim.miles1000	128	0.40	110	119	109	4.06	119	109	3.98	120	110	8.23	120	109	9.04
dim.queen7_7	49	0.40	31	40	31	0.90	40	31	1.25	40	31	3.17	40	31	6.36
dim.DSJC125.5	125	0.50	74	101	74	5.11	101	74	6.11	101	74	13.49	102	74	17.00
dim.DSJC125.9	125	0.90	22	63	22	5.73	62	22	6.20	63	22	6.63	62	22	6.61
mat.can96	96	0.20	72	89	72	1.78	87	72	2.33	89	72	6.52	89	72	9.09
mat.can73	73	0.25	53	65	53	1.60	64	53	1.69	66	53	6.91	66	53	7.76
mat.rw136	136	0.07	121	136	120	2.49	133	119	1.62	136	120	5.17	135	119	18.66
mat.gre__115	115	0.09	95	114	91	2.98	113	93	3.45	114	90	9.01	114	91	9.82
mat.L125.gre__185	125	0.15	104	120	104	4.64	119	104	4.19	120	104	20.01	122	104	22.37
mat.can__144	144	0.16	126	136	126	5.60	138	126	5.75	138	126	24.88	140	126	24.48
mat.L125.can__161	125	0.16	97	119	95	4.07	118	97	4.10	119	95	14.98	120	97	15.38
mat.lund_a	147	0.26	118	130	116	4.83	129	116	5.17	136	116	17.08	130	116	16.97
mat.L125.bcsstk05	125	0.35	101	108	101	3.48	104	101	3.68	116	101	8.35	107	101	10.30
mat.L125.dwt__193	125	0.38	95	105	95	3.49	102	95	3.68	107	95	8.60	106	95	6.91
mat.L125.fs_183_1	125	0.44	98	135	95	2.32	135	97	2.49	135	95	2.59	134	98	2.70
mat.bcsstk04	132	0.68	84	94	84	4.63	91	84	4.83	91	84	4.61	90	84	5.36
mat.arc130	130	0.93	88	102	88	7.51	100	88	7.80	103	88	12.65	100	88	13.21
mat.L100.steam2	100	0.36	76	82	76	2.85	82	76	2.84	83	76	9.43	83	76	10.76
mat.L120.fidap025	120	0.39	102	110	102	2.57	111	102	2.76	108	102	4.49	110	102	6.08
mat.L120.cavity01	120	0.42	99	120	99	3.36	119	99	2.60	121	99	4.77	122	98	4.88
mat.L120.fidap021	120	0.43	98	115	98	2.84	114	98	2.77	114	98	4.59	116	98	4.76
mat.L120.rbs480a	120	0.46	88	95	88	3.40	96	88	3.60	97	88	6.44	96	88	5.72
mat.L120.wm2	120	0.47	98	127	92	1.73	125	92	2.24	127	92	1.89	125	92	2.37
mat.L100.rbs480a	100	0.52	73	82	73	2.26	82	73	2.46	81	73	2.74	82	73	2.70
mat.L80.wm2	80	0.58	61	84	60	1.37	82	60	1.42	84	59	1.66	80	61	1.96
mat.L100.wm3	100	0.59	77	100	77	2.43	99	74	1.87	103	71	1.96	99	76	2.64
mat.L120.e05r0000	120	0.59	90	108	90	2.39	107	90	2.71	108	90	2.90	108	90	3.20
mat.L100.wm1	100	0.60	74	102	71	2.23	90	73	2.30	102	71	2.39	95	73	3.42
mat.L120.fidap022	120	0.60	84	91	84	3.87	90	84	4.10	92	84	4.70	91	84	4.20
mat.L100.fidapm02	100	0.62	69	70	69	2.39	70	69	2.28	70	**69**	2.79	**69**	**69**	2.70
mat.L120.fidap001	120	0.63	82	88	82	4.08	87	82	4.40	87	82	5.60	87	82	5.48
mat.L100.e05r0000	100	0.64	70	84	70	1.92	85	70	1.99	84	70	2.10	85	70	2.19
mat.L80.fidapm02	80	0.65	53	54	53	1.54	**53**	**53**	0.89	54	**53**	1.73	54	**53**	1.74
mat.L120.fidapm02	120	0.65	86	94	86	3.44	92	86	3.50	93	86	4.94	93	86	4.37
mat.L100.fidap001	100	0.68	64	71	64	2.76	69	64	2.87	73	64	3.06	70	64	3.20
mat.L100.fidap022	100	0.68	62	71	62	2.83	71	62	2.91	71	62	3.12	71	62	3.34
mat.L80.fidap001	80	0.72	54	62	54	1.40	62	54	1.52	62	54	1.47	62	54	1.75
mat.L80.fidap022	80	0.76	41	53	41	1.65	52	41	1.85	53	41	1.95	51	41	2.00
mat.L100.fidap027	100	0.81	69	70	69	2.48	**69**	**69**	1.60	**69**	**69**	1.94	**69**	**69**	2.12
mat.L100.fidap002	100	0.82	66	86	66	1.91	85	66	2.21	86	66	2.05	85	66	2.45
mat.L120.fidap002	120	0.82	68	91	68	3.09	89	68	3.15	88	68	3.41	89	68	3.37
mat.L120.fidap027	120	0.85	83	84	83	3.51	**83**	**83**	3.02	84	**83**	3.83	**83**	**83**	2.33
miplib.noswot.p	182	0.09	167	187	139	3.24	186	146	2.77	189	140	5.27	188	146	4.52
miplib.khb05250.p	100	0.27	75	99	75	1.10	95	75	1.20	99	75	1.20	95	75	1.31
miplib.stein27_r.p	118	0.32	62	116	62	3.78	106	62	4.20	118	62	18.84	110	62	23.56
miplib.10teams.p	210	0.34	120	203	120	10.55	180	120	11.21	205	120	25.06	188	120	25.99
miplib.mod010.p	146	0.38	90	145	88	3.99	126	86	5.26	149	85	6.74	131	85	19.19
miplib.l152lav.p	97	0.40	61	97	60	1.73	79	60	2.34	95	58	2.60	83	60	6.47
miplib.lp4l.p	85	0.46	50	80	47	1.87	63	48	1.95	81	48	1.82	70	49	4.32
miplib.air03.p	124	0.61	75	124	73	3.45	107	73	4.64	123	74	5.83	109	73	7.90
miplib.misc03.p	96	0.63	52	83	52	3.54	72	52	2.81	82	52	9.85	78	52	11.03
miplib.misc07.p	212	0.80	116	218	113	12.64	212	114	11.80	214	115	20.19	212	114	13.92

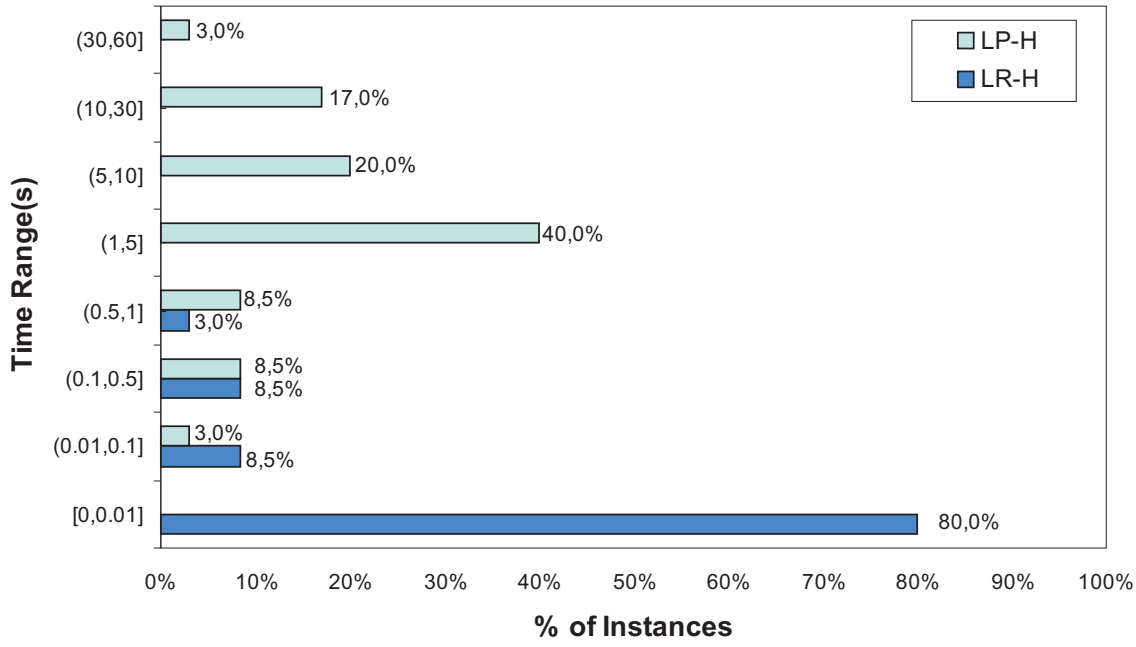

Figure 8: Time to optimum for Lagrangian (LR-H) and LP-based (LP-H) heuristics.

to proven optimality the vast majority of the instances. Moreover, inspecting the behavior of the B&C algorithm developed in [12], which we had access to, we noticed that a couple of CD inequalities needed to be separated and added to the model before good dual bounds are computed. Thus, it would be very helpful if one could quickly generate a set of initial CD cuts.

The HYBRID algorithm discussed in Section 5 appear as a possible option to handle this situation. According to the execution flow depicted in Figure 6, any of our relax-and-cut algorithms could be used to generate cuts that would allow the IP solver work on a tighter VSP formulation.

Before we report on the results achieved by the HYBRID algorithms, we redefine our data set. Initially, from the 62 instances originally selected to be part of the benchmark, only the 58 not solved to optimality by any relax-and-cut algorithms were kept. Later, we eliminate from our analysis the 11 instances that were not solved by any of the algorithms used in the totality of the experiments. We did so because they could introduce spurious information that could have distorted our conclusions. Thus, unless stated otherwise, the next discussions apply only to the 47 instances that are left.

Computational experiments were performed taking into account a considerable variety of HYBRID configurations. The results revealed a floating performance of the algorithms, depending on the density of the input graph. This observation confirms the reports in [12]. There, instances are classified to be of *high* ($\geq 35\%$) or *low* density according to the density of the input graphs. We proceeded similarly in our preliminary tests. However, these initial tests led us to regroup the instances in different classification ranges: mid-high ($> 20\%$) and low density instances, corresponding to a total of, respectively, 38 and 9 instances. The results reported in this section are organized in light of this regrouping. Furthermore, for each these new groups, the comparison measure used as a

basis for the majority of our conclusions was the total amount of time required and nodes explored to solve all the instances in each group.

7.2.1 Results for mid-high density graphs

We now report the main results obtained with mid-high density instances which correspond to the majority of our final test set. Table 3 compares the outcomes of the most promising algorithms for mid-high density instances. A total of three postponed and non-delayed relax-and-cut configurations have their results reported. These configurations prioritize the separation of `CD` inequalities, in accordance to what is suggested in [12]. Also, we display the results returned by two variations of the B&C algorithms from [12] – to our knowledge, the best ones available in literature to solve the VSP – and by XPRESS under default settings.

The number of nodes and the time required for each algorithm are reported. When the time exceeds 1800 seconds, it means that the instance was not solved by the corresponding algorithm within that time bound. The main headings that identify the relax-and-cut configurations whose results are presented in Table 3 have the following meaning: `(i) Only CD Cuts`: corresponds to the usage of our NDR&C(CD) (or PR&C(CD)) algorithm followed by L-B&C, with sep =⟨`L-CD,CD`⟩. This configuration is denoted later by NDHYBRID(CD) (PHYBRID(CD)); `(ii) CD Cuts and L-LD Cuts`: combines the usage of NDR&C(CD,LD) (or PR&C(CD,LD)) as relax-and-cut algorithm with L-B&C, with sep =⟨`L-CD,CD,L-LD`⟩. This configuration is denoted later by NDHYBRID(CD,L-LD) (PHYBRID(CD,L-LD)); `(iii) CD and LD Cuts`: refers to NDR&C(CD,LD) (or PR&C(CD,LD)) preceding L-B&C, with sep =⟨`L-CD,CD,L-LD,LD`⟩. This configuration is denoted later by NDHYBRID(CD,LD) (PHYBRID(CD,LD)). Also, columns `B&C(CD)` and `B&C(CD,LD)` correspond to the algorithm described in [12] separating, respectively, only CD and both, CD and LD inequalities. Finally, XPRESS results are reported on the last two columns.

At the bottom of each column and for each algorithm, three summations are shown. The first of them corresponds to the total time (or total number of nodes explored in the search trees) only for those algorithms that solved to optimality the whole set of instances listed in the table. To understand the other summations, let S' be the subset of instances in Table 3 that are solved by any HYBRID version and any B&C configuration within 30 minutes, i.e., all of them except instance `miplib.misc07.p`. Likewise, let S'' be the subset of instances in S' also solved by XPRESS within the same time limit. (i.e., $S' \backslash S'' = \{$`dim.DSJC125.9`, `mat.lund_a`, `mat.bcsstk04`, `mat.L120.fidap001` and `miplib.air03.p`$\}$). The penultimate (last) line contains the total number of nodes and time needed by each approach to solve all the instances in S' (S'') subset. Unless stated otherwise, our analyses are restricted to instances in S'' only when XPRESS results are also under consideration.

Entries with the symbol "−" correspond to the instances that were solved after the LP module execution, i.e., before branching. These entries permit us to conclude that, besides the four instances already solved during the Lagrangian phase, six more instances were solved to optimality before entering the L-B&C module in Figure 6.

Comparing the last three lines in Table 3, we can conclude that: `(1)` all the six HYBRID proposed outperform the B&C algorithm detailed in [12] over the S' data set; `(2)` over the subset S'' of instances, the HYBRID algorithms also performed better, in terms of time, than the other three approaches. However, the number of nodes of the search tree explored by B&C(CD) is slightly smaller than that of our best HYBRID configurations; `(3)` algorithms based chiefly on `CD` inequalities seem to be the most promising approaches currently available to tackle mid-high

Table 3: Results for VSP instances using HYBRID configurations, B&C algorithms from [12] and XPRESS.

Mid-high density instance		Only CD Cuts				CD Cuts and L-LD Cuts				CD and LD Cuts				B&C [ref.[12]]				B&B (XPRESS)	
		NDHYBRID		PHYBRID		NDHYBRID		PHYBRID		NDHYBRID		PHYBRID		B&C(CD)		B&C(CD,LD)			
label	$d(>20\%)$	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)
dim.queen8_8	0.36	1807	75.02	1707	90.22	1863	85.01	2283	136.40	2541	125.79	2283	136.03	4315	70.51	3143	58.27	23131	126.29
dim.miles1000	0.40	17	12.97	13	11.62	7	10.82	9	14.05	9	11.52	9	14.65	11	18.37	35	26.22	287	83.96
dim.queen7_7	0.40	391	14.11	313	16.73	403	16.56	243	20.48	253	17.13	243	20.89	431	10.77	265	9.93	27833	78.53
dim.DSJC125.9	0.90	4275	596.45	4143	537.01	4229	532.45	6405	722.50	6375	743.80	6417	775.66	33833	1107.29	28475	1291.54	51261	1800.00
mat.can73	0.25	5343	46.15	5505	50.57	4975	64.53	5157	59.63	5367	57.37	5195	59.35	5123	71.50	5615	46.44	33195	147.62
mat.lund_a	0.26	3145	401.39	2715	384.14	2033	454.57	3565	543.09	3345	479.10	3005	511.01	2231	462.03	2709	332.09	27506	1800.00
mat.L125.bcsstk05	0.35	709	85.65	-	4.53	-	7.25	-	8.02	-	7.25	-	8.02	1573	326.20	1625	196.01	1635	211.38
mat.L125.dwt_193	0.38	21	23.11	27	25.07	21	34.49	39	40.78	41	45.06	39	40.66	131	97.31	721	134.80	17767	1301.43
mat.L125.fs_183_1	0.44	29	28.11	21	26.16	29	32.82	746	69.37	29	43.27	27	41.02	25	35.45	29	31.74	1515	182.78
mat.bcsstk04	0.68	13	22.90	-	4.35	13	22.02	-	5.68	31	31.14	-	5.68	133	124.60	247	132.27	16572	1800.00
mat.arc130	0.93	83	160.19	83	163.80	103	186.66	73	231.78	75	246.58	73	235.17	101	370.67	101	329.59	957	926.12
mat.L100.steam2	0.36	45	21.02	41	20.67	45	25.73	79	38.06	77	34.79	69	35.92	149	40.83	241	37.98	11577	229.98
mat.L120.fidap025	0.39	-	2.50	-	2.59	-	3.58	-	4.75	-	3.58	-	4.75	13	12.00	49	17.67	889	107.73
mat.L120.cavity01	0.42	13	9.13	15	10.50	11	9.62	35	18.35	35	14.87	15	11.83	13	16.88	41	21.52	813	91.69
mat.L120.fidap021	0.43	5	5.63	7	5.88	3	7.14	3	6.98	3	7.34	3	7.00	35	24.74	67	36.63	1031	150.22
mat.L120.rbs480a	0.46	125	64.01	141	67.75	123	71.81	75	59.20	75	60.34	75	61.14	367	218.67	3007	249.27	15619	1308.79
mat.L120.wm2	0.47	33	28.04	35	36.68	75	19.47	73	26.55	75	20.25	33	38.57	33	47.82	33	45.71	351	88.71
mat.L100.rbs480a	0.52	59	11.90	67	15.60	61	18.18	45	15.44	49	17.81	45	16.76	63	21.33	91	21.34	2951	189.73
mat.L80.wm2	0.58	9	3.28	11	4.31	13	4.82	51	6.07	13	5.90	13	6.53	13	4.90	15	5.65	379	67.22
mat.L100.wm3	0.59	11	7.63	13	10.48	17	12.00	61	16.24	19	18.64	13	10.66	17	13.26	15	13.34	379	65.70
mat.L120.e05r0000	0.59	5	7.05	3	7.63	7	7.70	7	7.74	5	8.51	7	7.82	9	11.49	43	25.31	2703	543.05
mat.L100.wm1	0.60	19	10.74	13	9.18	17	9.59	71	18.40	27	14.35	17	12.00	25	15.67	35	24.36	877	94.07
mat.L120.fidap022	0.60	77	22.53	17	13.75	81	27.72	43	20.60	41	20.28	43	21.22	53	38.17	81	53.04	13319	1522.86
mat.L120.fidap001	0.63	-	4.94	-	5.07	-	6.71	-	6.66	-	6.71	-	6.66	31	32.57	189	84.70	33120	1800.00
mat.L100.e05r0000	0.64	15	8.39	17	8.17	15	8.75	19	6.46	33	8.50	19	6.59	19	11.19	39	12.49	3559	284.25
mat.L120.fidapm02	0.65	-	2.87	-	2.91	-	4.28	-	3.65	-	4.28	-	3.65	17	24.52	57	55.66	4457	552.37
mat.L100.fidap001	0.68	35	7.10	35	7.54	33	9.00	29	9.87	35	11.46	29	9.82	49	15.96	73	23.21	34321	950.38
mat.L100.fidap022	0.68	109	22.66	105	22.36	99	23.75	54	16.17	63	18.00	54	17.19	171	52.20	93	27.31	57415	1594.48
mat.L80.fidap001	0.72	-	1.55	-	1.55	-	1.61	-	1.75	-	1.61	-	1.75	1	1.76	33	5.20	3523	101.25
mat.L80.fidap022	0.76	197	14.25	159	11.90	135	11.92	57	7.18	55	7.89	57	7.68	173	15.28	45	5.51	19279	308.05
mat.L100.fidap002	0.82	5	3.22	3	3.52	5	3.48	5	4.00	3	3.10	5	4.05	7	4.75	29	10.19	2111	240.58
mat.L120.fidap002	0.82	5	6.88	1	5.73	5	6.60	3	5.56	3	6.72	3	5.58	73	41.59	93	48.59	10415	1284.46
miplib.khb05250.p	0.27	119	3.94	121	5.26	119	4.28	121	5.16	119	4.48	121	5.43	91	3.21	111	4.84	3641	66.37
miplib.l152lav.p	0.40	213	46.27	185	54.66	245	64.35	611	72.76	749	103.57	129	60.48	283	70.12	853	101.98	22885	567.68
miplib.lp4l.p	0.46	275	34.27	271	33.10	321	37.02	2083	93.82	1295	69.37	265	50.19	551	50.10	5965	151.72	27523	409.73
miplib.air03.p	0.61	115	111.94	117	119.98	117	111.50	135	139.86	119	146.10	111	163.87	135	167.35	135	180.01	14215	1800.00
miplib.misc03.p	0.63	2993	111.95	2717	121.67	3293	138.31	2589	168.23	2785	178.78	2225	173.42	4819	138.07	3947	155.22	53417	1794.42
miplib.misc07.p	0.80	173	1280.36	177	1519.38	169	1162.75	1102	1809.86	117	1402.40	147	1704.75	125	1800.00	78	1800.00	2243	1800.00
SUM TOTAL of S		20480	3320.10	18789	3442.02	18685	3258.85	-	-	23861	4007.64	20789	4303.45	-	-	-	-	-	-
SUM TOTAL of S'		20315	2039.74	18621	1922.64	18516	2096.10	24769	2631.29	23744	2605.24	20642	2598.70	47706	3286.19	58345	4007.35	-	-
SUM TOTAL of S''		15912	1303.51	14361	1256.23	14157	1423.42	18229	1756.59	17219	1677.49	14114	1646.83	13574	1854.38	29299	2318.83	427260	17471.88

density instances. This observation reinforces the conclusions reported in [12] about the use and the strength of CD inequalities to solve high ($\geq 35\%$) density instances. Nevertheless, the outcomes from our experiments showed that, for instances `mat.can73`, `mat.lund_a` and `miplib.khb05250.p`, both hybrid and branch-and-cut algorithms outperform XPRESS. Hence, one could push the frontier of the conclusions of de Souza and Balas to include mid-high density (over 20%) graphs.

Besides, we notice that five out of our six configurations were able to solve the instance `miplib.misc07.p` whereas no other approach tested solved it within 30 minutes. Direct comparison of these five algorithms over the whole set of instances in Table 3 allows us to conclude that our three best configurations come from using only CD cuts and from using NDR&C(CD,LD) as relax-and-cut algorithm combined with L-B&C, with sep $=\langle$`L-CD,CD,L-LD`$\rangle$ denoted, as said before, by NDHYBRID(CD,L-LD).

In addition to that conclusion, for a fairer comparison of the entire set of algorithms, we restrict ourselves to instances in S'' (last row of Table 3). One can see that PHYBRID(CD) is the fastest algorithm. Figure 9 highlights the results obtained with PHYBRID(CD) and B&C(CD), the best branch-and-cut version from [12]. The performance of these algorithms is measured as a percentage of the time required by XPRESS B&B to solve the instances in S''. First, notice that, in general, both algorithms are much faster than XPRESS B&B under default settings. Actually, only for instance `mat.L125.bcsstk05` XPRESS surpassed B&C(CD). Also, on average, PHYBRID(CD) is about two times faster than B&C(CD): the former requires, on average, about 10.6% of the computation time used by XPRESS to solve the instances while the latter needs approximately 19.3% of that time.

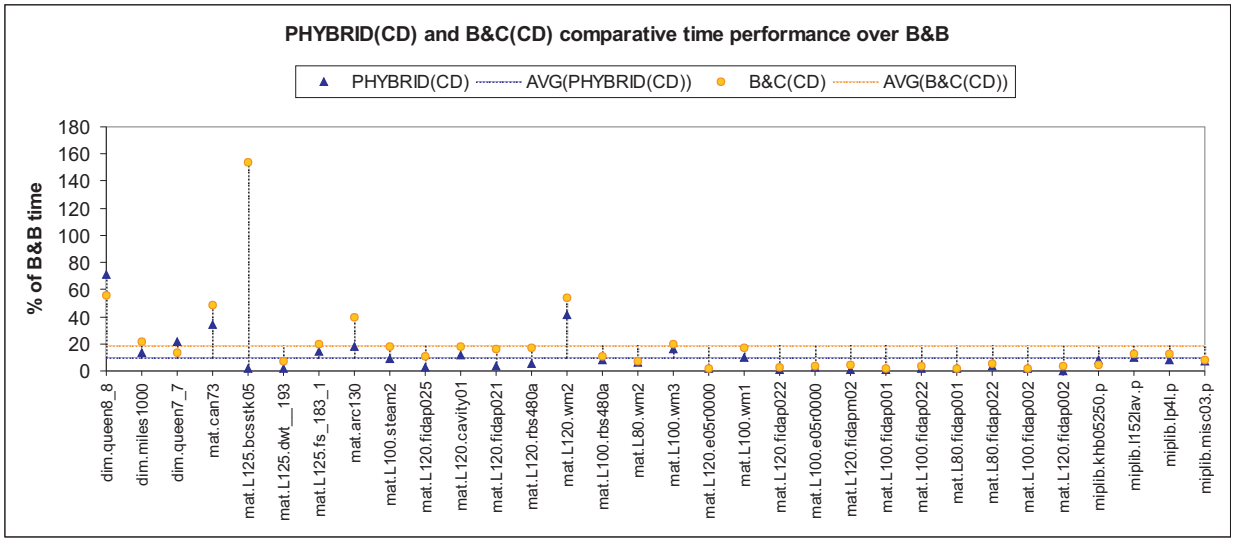

Figure 9: Time performance: PHYBRID and B&C(CD) against XPRESS (default) B&B.

Figure 10 exhibits a graphical comparison of the computation times among the six combined approaches reported in Table 3 against the two best B&C versions. It highlights the big savings obtained by the HYBRID algorithms with respect to the B&C algorithms. For instance, the effect of replacing B&C(CD,LD) by PHYBRID(CD) would be a CPU time reduction of 52%. Also, on average, considering the results for the six variations of our hybrid approach, time savings over B&C(CD,LD) and B&C(CD) are, respectively, of about 42% and 30%.

Back to Table 3, one may argue that instance `dim.DSJ125.9` may have distorted a little the analysis to the detriment of the B&C algorithms. In fact, it had contributed with high amounts to both the total number of nodes and the running time. However, it should be noticed that this instance does not belong to S'' and the superiority of the HYBRID algorithms over this set is also noticeable. For this set, the reduction in time by using PHYBRID(CD) in place of B&C(CD) was of 32.3% (compared to 41.5% for S'). Moreover, PHYBRID(CD) ran faster than B&C(CD) in 92% of the cases, namely, in 35 out of the 38 mid-high density instances tested. This can be easily seem with the help of Figure 12 where the time performance of the two codes are compared with respect to graph densities.

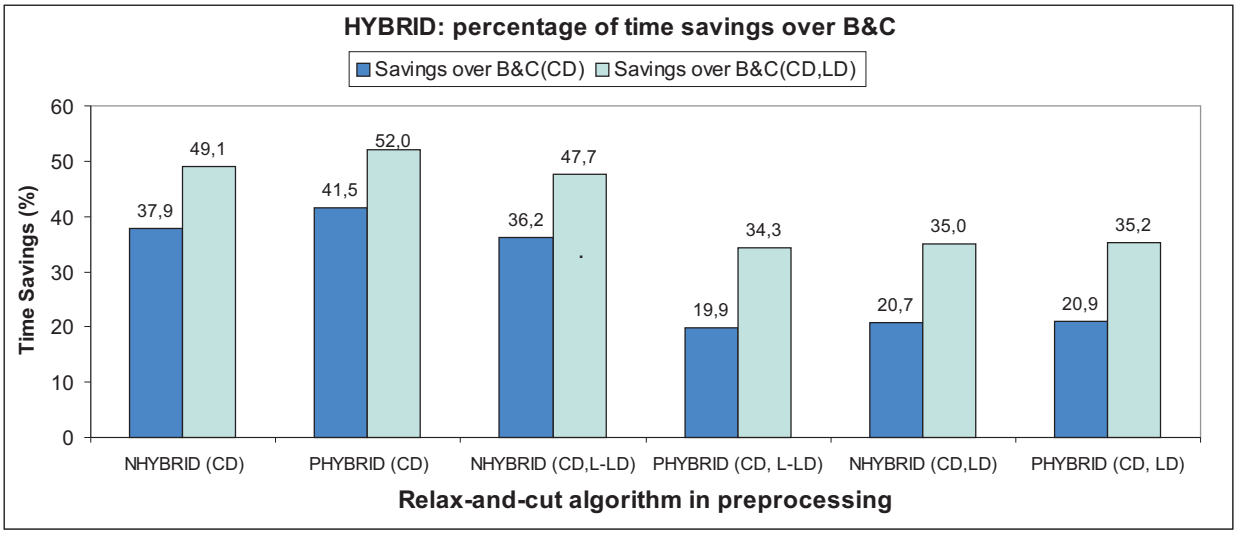

Figure 10: Time performance: HYBRID variants × B&C algorithms described in [12].

Consolidated × instance-based analysis. Although usual in literature, analyses of computational results focused on the total time spent in solving the entire data set may mislead the conclusions. Driven by this observation, we constructed the graphic in Figure 11 that shows an alternative way of composing results on time performance. There, for each HYBRID algorithm, the percentage of B&C(CD) CPU time consumed to solve each instance in S' is computed and an arithmetic mean is taken. Basically, the average expression is given by $AVG(a) = 100 \times \frac{1}{|S'|} \times \sum_{i \in S'} (\frac{t(i,\text{HYBRID}(a))}{t(i,\texttt{B\&C(CD)})})$, $a \in \{1, ..., 6\}$, where a are labels corresponding to the HYBRID algorithms tested and $t(i, A)$ refers to the CPU time algorithm labeled as A takes to solve instance i. Hence, those values refer to the percentage of B&C(CD) execution time one should expect to be spent, on average, by each one of our algorithms. The premise adopted is that, under discrepancies of running times, replacing total time by percentage may lead to a more suitable analysis.

Comparing graphics in Figures 10 and 11 we see, however, that quite similar values were generated, presenting deviation inferior to 4%. For example, the time reduction with NDHYBRID(CD) in Figure 10 is of 37.9% while in Figure 11 it is of 41.3%. Notice that, rather than unexpected, the similarities between both results reinforce what is shown in Figure 10. Taking into account these computational results concerning time performance we can conclude that NDHYBRID(CD) and

PHYBRID(CD) are the best algorithms we developed. To better illustrate that in details, graphic in Figure 12 carries out a comparison between both approaches. Similarly to what happened in Figure 11, we adopted percentages over B&C(CD) running times. The results are presented in increasing order of graph density. One can see that, in general, for instances having input graph density below to 60%, NDHYBRID(CD) outperforms the postponed version. On the other hand, PHYBRID(CD) shows to be more suitable to solve higher density instances.

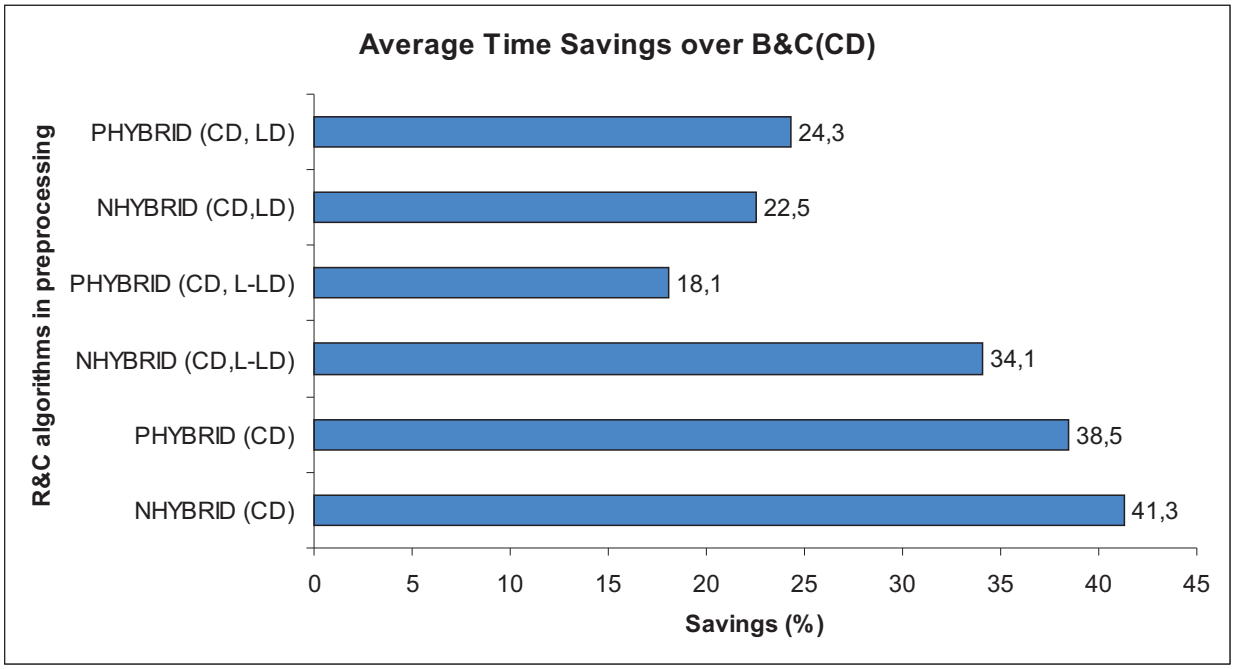

Figure 11: Average time savings of HYBRID algorithms relative to B&C(CD) times for instances in S'.

As seen above, consolidated and instance-based analyses both led to the conclusion that PHYBRID(CD) and NDHYBRID(CD) are the two most promising algorithms to solve instances in S'. However, these results do not render possible to assert that one algorithm excels the other. In an attempt to answer to that question, further investigation is reported below. Percentages over B&C(CD) running times was adopted in order to compare the two algorithms.

PHYBRID × NDHYBRID**: the Wilcoxon signed-rank (WSR) test.** The Wilcoxon signed-rank test is a well known nonparametric statistical test [5, 15] that has been used in the optimization literature [1, 13, 15, 21] to compare two heuristics. The final outcome of the test is always given in terms of the null hypothesis: we either reject the null hypothesis or fail to reject it. When we reject the null hypothesis, we have only shown that it is highly unlikely to be true – we have not proven it in the mathematical sense. Rejecting the null hypothesis then, suggests that an alternative hypothesis may be true. Alternative hypothesis may be one-tailed or two-tailed. A one-tailed hypothesis claims that a parameter is either larger or smaller than the value given by the null hypothesis. A two-tailed hypothesis claims that a parameter is simply not equal to the value given by the null hypothesis - the direction does not matter.

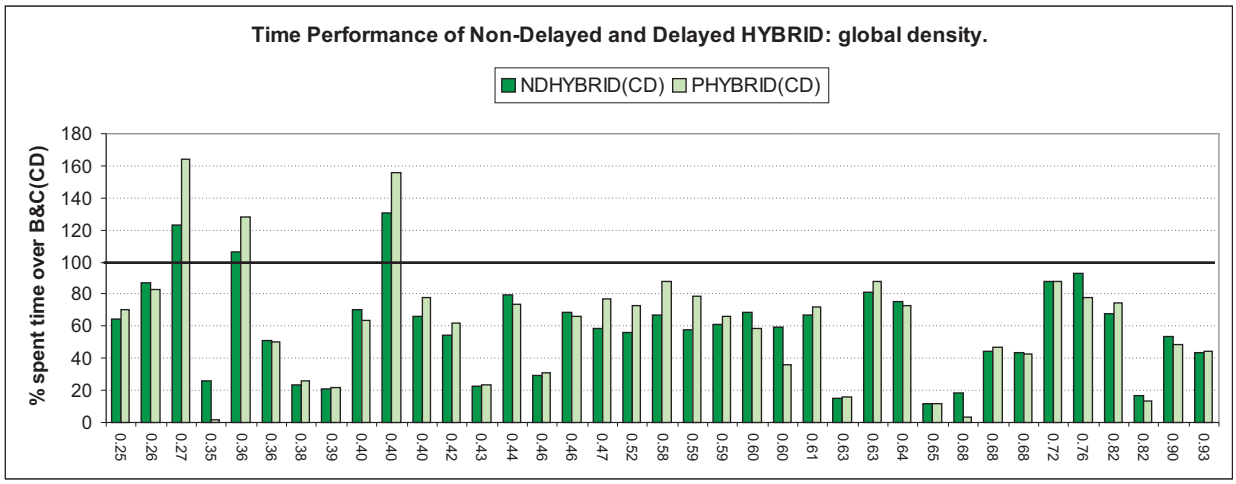

Figure 12: Time performance: postponed × non-delayed versions of HYBRID (grouped by input graph densities).

Here, we apply WSR test to try to infer, statistically, any superiority between PHYBRID(CD) and NDHYBRID(CD) algorithms when solving mid-high density VSP instances (the null hypothesis claims that there is no dominance of one algorithm over the other one). Hence, we applied the test using a directional (one-tailed) hypothesis over the results used to build the graphic in Figure 11.

Let W and W' be, respectively, the sum of the signed ranks and the critical value computed in WSR test. As the number n of instances used in the test increases, the distribution of W tends toward the normal distribution. Furthermore, for $n \geq 10$, the critical value W' can be approximated by $W' = Z(\alpha)\sqrt{n(n+1)(2n+1)/6}$, where $Z(\alpha)$ corresponds to the standard normal quantile such that a proportion α of the area is to the left of $Z(\alpha)$. In fact, α is the term used to express the level of significance we will accept the hypothesis. As an example, for 90% confidence, $\alpha = 0.10$, and testing a hypothesis at the $\alpha = 0.10$ level or establishing a 90% confidence interval are essentially the same thing. In both cases the critical values and the region of rejection are the same.

In our case, the differences between matched pairs of results computed in WSR were obtained subtracting values obtained by NDHYBRID(CD) from values corresponding to PHYBRID(CD) results. Thus, if $W > W'$, the null hypothesis should be rejected at the α significance level [15]. As a consequence, for that confidence assumed when computing W', it would mean that NDHYBRID(CD) has better performance than PHYBRID(CD). Actually, the sum of the signed ranks computed with our results was 181 and, if we set $\alpha = 0.10$ ($Z(\alpha) = 1.282$), we obtain $W' = 169.96$. This tell us that if the null hypothesis is true, then in only 10% of the cases W is expected to exceed 169.96. Hence, we refute the null hypothesis at the significance level of 90% and infer that NDHYBRID(CD) outperforms PHYBRID(CD).

Applying Wilcoxon signed-rank test clearly increases our confidence in the comparison carried out between both algorithms and graphically shown in Figure 11. However, it is noteworthy that we may not be able to reject the null hypothesis if we try to come out with a stronger evidence of superiority of NDHYBRID(CD). A manner to conclude that is using a smaller α value. For instance, if we set $\alpha = 0.05$ ($Z(\alpha) = 1.645$), we get $W' = 218.08$. It implies, according WSR test, that we failed to reject the null hypothesis for 95% confidence, i.e., no dominance is verified at that

significance level.

7.2.2 HYBRID algorithms and sparse graphs

Computational results reported in [12] discourage the use of cutting planes corresponding to CD or LD inequalities when the input graph is sparse. There, in general, the increase in computing time per search tree node resulted in an increase of total computing time. In other words, using XPRESS with default settings was, normally, more advantageous for instances associated to low-density graphs. Despite these reports, we decided to investigate if that conclusion goes on being true when using a more recent XPRESS solver version. Thus, some variations of our hybrid approach were tested, as well as the two best branch-and-cut configurations from [12]. Tables 4 and 5 and Figure 13 document the main results obtained with these computational experiments.

Table 4 shows, essentially, four configurations of our hybrid algorithms: two postponed and two non-delayed relax-and-cut were used as preprocessing. The number of nodes and the total CPU time required are reported for each algorithm. When the time is $\geq$ 1800 seconds, it means that the instance was not solved by the corresponding algorithm. The main headings that identify the relax-and-cut configurations whose results are presented in Table 4 have the following meaning: `(i) Only CD Cuts`: corresponds to the usage of our NDR&C(CD) (or PR&C(CD)) followed by L-B&C, with sep =⟨`L-CD,CD`⟩; `(ii) L-CD Cuts and L-LD Cuts`: combines the usage of NDR&C(CD,LD) (or PR&C(CD,LD)) preceding L-B&C, with sep =⟨`L-CD,L-LD`⟩. `(iii) CD Cuts and L-LD Cuts`: regards to the usage of NDR&C(CD,LD) (or PR&C(CD,LD)) as relax-and-cut algorithm with L-B&C, having sep =⟨`L-CD,CD,L-LD`⟩; `(iv) CD and LD Cuts`: refers to NDR&C(CD,LD) (or PR&C(CD,LD)) preceding L-B&C, with sep =⟨`L-CD,CD,L-LD,LD`⟩.

Complementary, in Table 5, columns `B&C(CD)` and `B&C(CD,LD)` correspond to the algorithm described in [12] separating, respectively, only CD and both, CD and LD inequalities. Also, the results returned by XPRESS under default settings are shown. In both tables, at the bottom of each column, two summations are shown for each algorithm. To understand them, let $\dot{S}$ be the subset of instances with sparse graphs that are solved by any HYBRID and any B&C configuration within 30 CPU minutes, i.e., we exclude instances `mat.can96`, `mat.rw136` and `mat.can_144`. In addition, let $\ddot{S} \subset \dot{S}$ be the subset of instances in $\dot{S}$ also solved by XPRESS within that time limit. The penultimate (last) line contains the total number of nodes and time needed by each approach to solve all the instances in $\dot{S}$ ($\ddot{S}$) subset.

Examining the values corresponding to the variants of our combined approach (Table 4, subset $\dot{S}$) we can deduce that our best performances were attained by NDHYBRID(L-CD,L-LD) and PHYBRID(L-CD,L-LD). This observation lead us to infer that, in this case, cuts discovered during the Lagrangian phase were helpful.

Comparing B&C algorithms via results in Table 5 we can conclude that separating both, `CD` and `LD` cuts, is better than identifying only CD inequalities. So, as opposed to what has been seen to mid-high density instances, B&C(CD,LD) outperforms B&C(CD).

Inspecting the behavior of XPRESS in Table 5, we can see that it was the only algorithm to solve instances `mat.can96` and `mat.rw136`. Also, together with NDHYBRID(CD,L-CD), they were the only ones to solve instance `mat.can_144`. On the other hand, XPRESS was the single approach to fail when trying to solve `dim.myciel7`. Now, restricting ourselves to the other five instances (i.e., $\ddot{S}$ set), results suggest that our three best hybrid algorithms, NDHYBRID(L-CD,L-LD), PHYBRID(L-CD,L-LD) and PHYBRID(CD,L-LD), outperform both, B&C from [12] and XPRESS, when the criterion is

Table 4: Computational results for VSP low density ($\leq 20\%$) instances: NDR&C and PR&C.

Low density instance		Only CD Cuts				L-CD Cuts and L-LD Cuts				CD Cuts and L-LD Cuts				CD and LD Cuts			
		NDHYBRID		PHYBRID		NDHYBRID		PHYBRID		NDHYBRID		PHYBRID		NDHYBRID		PHYBRID	
label	d	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)	nodes	t(s)
dim.games120	0.09	84625	789.82	86767	978.87	96985	740.16	97349	754.08	82677	1066.70	85615	869.15	87667	1071.31	85787	1026.52
dim.myciel7	0.13	1823	304.02	2493	383.73	2499	287.90	2799	270.30	2251	374.36	1991	269.26	2779	442.20	2491	429.75
dim.myciel6	0.17	323	9.32	333	10.86	415	10.30	395	10.03	373	13.44	385	11.03	409	14.47	319	13.15
mat.can96	0.20	191769	1801.32	162842	1801.58	192310	1803.07	157351	1803.77	131426	1803.07	134256	1803.77	159271	1803.07	129012	1803.77
mat.rw136	0.07	72063	1802.28	80603	1801.70	345446	1803.05	155538	1812.90	367218	1803.05	88741	1812.90	334492	1803.05	71316	1812.90
mat.gre_115	0.09	7185	92.91	6471	92.90	6393	93.32	6371	89.77	6623	129.86	6047	111.01	6939	127.35	5993	113.05
mat.L125.gre_185	0.15	1169	94.53	1089	92.68	1655	83.39	1639	77.56	1139	105.65	1255	106.63	1191	105.05	1255	111.04
mat.can_144	0.16	38265	1500.97	39849	1807.43	29193	1818.34	24692	1818.96	26048	1818.34	21465	1818.96	24977	1818.34	21666	1818.96
miplib.noswot.p	0.09	27697	1265.17	27759	1351.65	29227	687.80	29219	715.29	27833	1342.92	17853	740.22	18879	882.22	17783	1012.83
SUM TOTAL of $\dot{S}$		122822	2555.77	124912	2910.69	137174	1902.87	137772	1917.03	120896	3032.93	113146	2107.30	117864	2642.60	113628	2706.34
SUM TOTAL of $\hat{S}$		120999	2251.75	122419	2526.96	134675	1614.97	134973	1646.73	118645	2658.57	111155	1838.04	115085	2200.40	111137	2276.59

Table 5: Computational results for VSP low density ($\leq 20\%$) instances: B&C and B&B.

Low density instance		B&C				B&B (XP)	
		B&C(CD)		B&C(CD,LD)			
label	d	nodes	t(s)	nodes	t(s)	nodes	t(s)
dim.games120	0.09	82963	886.29	85051	980.73	161485	1077.10
dim.myciel7	0.13	3009	562.94	2033	328.17	28881	1800.00
dim.myciel6	0.17	377	14.60	423	11.69	5243	62.32
mat.can96	0.20	107849	1800.00	167804	1800.00	177163	1569.57
mat.rw136	0.07	62049	1800.00	82714	1800.00	10083	81.55
mat.gre_115	0.09	6517	103.87	6539	83.22	37177	295.66
mat.L125.gre_185	0.15	1205	131.53	1603	71.00	15795	273.68
mat.can_144	0.16	19742	1800.00	27700	1800.00	12683	339.10
miplib.noswot.p	0.09	27891	1343.88	17763	845.51	34719	849.71
SUM TOTAL of $\dot{S}$		121962	3043.11	113412	2320.32	–	–
SUM TOTAL of $\hat{S}$		118953	2480.17	111379	1992.15	254419	2558.47

processing time. These results can be more clearly visualized in Figure 13 that compile the results of all approaches, confronting them to XPRESS B&B results. Observe that XPRESS standard B&B code is by far the worst algorithm with respect to the number of nodes explored by the search tree. Similarly, in terms of processing time, it is worse than all approaches, except NDHYBRID(CD,L-LD) version.

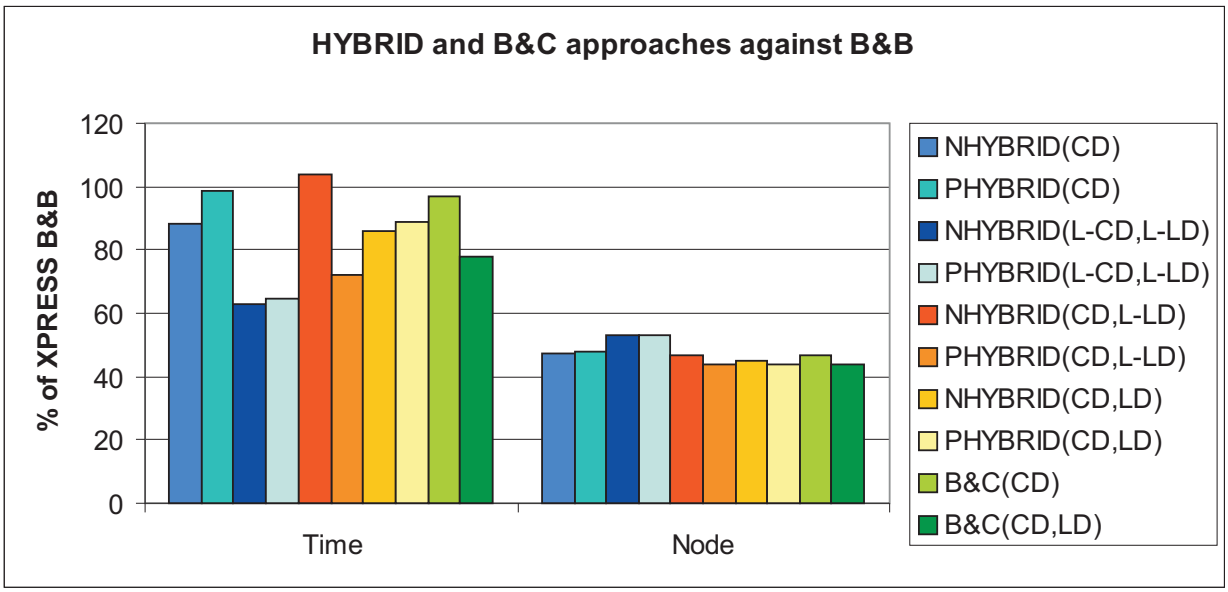

Figure 13: Sparse graphs results for instances in $\ddot{S}$.

7.3 Results on hard MIPLIB instances

A similar problem that can be considered a generalization of the VSP is discussed in [6] by Borndörfer *et al.* There is described a branch-and-cut algorithm that use cutting planes different from [12] and computational results are reported on a vast number of instances. Several instances from MIPLIB are used and a subset of them demonstrated, experimentally, to be very hard to solve (or remained unsolved) within the time limit imposed to the algorithm execution. Three among these instances have already their results reported in tables 3, 4 and/or 5: `miplib.noswot.p` and `miplib.misc03.p` were solved by all the approaches tested whereas `miplib.misc07.p` was solved to proven optimality only by the hybrid framework we proposed.

Computational experiments were carried out aiming to assess the performance of our algorithms when tackling also the other instances from MIPLIB which were not solved in [6]. Instances whose underlying graph is disconnected were discarded, for we are working with `CD` inequalities. As before, B&C algorithms from de Souza and Balas [12] and XPRESS were tested. The results obtained are detailed in Tables 6 and 7. They have the same format as before: columns `R&C` show computation times spent by the relax-and-cut algorithm used as preprocessing and under `gap` columns we list the final gaps concerning each algorithm. Notice that, with the exception of computation times, these results can also be compared with those reported in [6][4].

[4]For comparison purposes, notice that the values concerning the bounds used to compute the gaps reported here

The selection of HYBRID implementations to test these hard MIPLIB instances was based on the results reported in the previous sections. Accordingly, to test mid-high and low density instances we choose, respectively, NDHYBRID(CD) and PHYBRID(L-CD,L-LD) versions. The choice concerning mid-high density instances was immediate from the previous analyses on the results reported in Table 3 and Figure 11. However, due to the very similar performance between NDHYBRID(L-CD,L-LD) and PHYBRID(L-CD,L-LD) reported in Table 4, no natural choice comes out. However, a quick instance-based analysis decided in favor of PHYBRID(L-CD,L-LD).

Results in Table 6 confirm the difficulty found by the various algorithms we tested to solve those instances within the time limit of 30 minutes. However, regarding final gaps (entered as percentages), they reveal that our hybrid algorithm outperforms the other ones in 4 out of the 7 instances. Also, it was able to solve `p0282` in less than 30% of the time needed by the second best algorithm.

Final gaps reported in Table 7 show that, in 12 out of the 14 instances, our combined approach is at least as good as the other algorithms. Notice that our algorithm and the B&C described in [12] were capable to solve the same subset of instances. However, our algorithm did it faster than the latter. On the other hand, the number of nodes yielded by our search trees were usually higher than the amount of nodes generated when applying de Souza and Balas' algorithm.

Table 6: Computational results for MIPLIB open problems: mid-high density ($> 20\%$) instances.

Instance			NDHYBRID(CD)				B&C(CD)			B&B			B&C[6]
label	$d(>20\%)$	n	R&C	total t(s)	nodes	gap(%)	t(s)	nodes	gap(%)	t(s)	nodes	gap(%)	gap(%)
fast0507	20.82	484	24.57	1824.57	134	57.20	1800.00	194	68.49	1800.00	1188	86.39	59.14
stein27_r	32.20	118	4.06	1804.06	22930	34.21	1800.00	20577	36.42	1800.00	89909	32.64	38.71
air05	34.37	408	39.03	1839.03	173	75.76	1800.00	233	83.43	1800.00	317	88.19	77.63
10teams	34.45	210	10.36	1810.36	1937	57.29	1800.00	3450	35.43	1800.00	13827	48.10	39.17
mod010	37.97	146	5.04	1805.04	12336	3.58	1800.00	10146	13.69	1800.00	39346	27.82	13.79
misc05	40.09	266	13.61	1813.61	246	55.57	1800.00	129	70.54	1800.00	5467	60.13	58.67
p0282	40.89	161	4.43	31.58	23	0.00	114.99	73	0.00	1112.33	2399	0.00	10.40

Table 7: Computational results for MIPLIB open problems: low density ($\leq 20\%$) VSP instances.

Instance			PHYBRID(L-CD,L-LD)				B&C(CD,LD)			B&B			B&C[6]
label	d	n	R&C	total t(s)	nodes	gap	t(s)	nodes	gap	t(s)	nodes	gap	gap
set1al	0.78	492	11.98	162.30	13567	0.00	471.28	11439	0.00	1695.38	39175	0.00	1.25
set1cl	0.78	492	11.98	165.72	13567	0.00	464.22	11439	0.00	1731.34	39175	0.00	1.25
set1ch	0.81	477	10.80	153.31	13512	0.00	430.26	11373	0.00	1800.00	30008	0.36	1.08
fixnet3_r	1.10	478	12.91	20.73	77	0.00	21.24	71	0.00	406.00	5399	0.00	0.22
misc06	1.21	696	18.07	1818.07	12426	5.52	1800.00	7363	5.60	1800.00	13862	15.63	10.54
qnet1_o	3.59	369	7.64	60.03	677	0.00	215.86	497	0.00	1800.00	16989	2.38	4.69
qnet1	3.60	407	10.57	212.52	1085	0.00	532.12	1115	0.00	1800.00	19023	8.32	7.26
danoint	4.49	664	52.14	1852.14	130	27.11	1800.00	1513	29.04	1800.00	5114	70.70	38.66
gams	5.13	291	8.42	373.46	5235	0.00	419.52	4869	0.00	1800.00	22587	1.52	2.22
adrud	5.50	795	43.40	1254.74	193	0.00	1325.10	205	0.00	1800.00	1110	2.76	2.88
p0548	7.82	257	5.57	1805.57	125611	18.28	1800.00	24086	10.69	1800.00	41048	13.62	11.54
air04	16.67	782	71.29	1871.29	17	73.54	1800.00	220	87.29	1800.00	18	89.67	82.44
air06	16.82	570	51.00	1851.00	67	70.42	1800.00	184	85.51	1800.00	710	83.49	162.50
stein45_r	19.59	331	30.83	1830.83	88	79.95	1800.00	723	84.61	1800.00	7253	73.29	80.79

Notice that twelve MIPLIB instances remained unsolved after our experiments with our combined approach. For these problems, we kept our focus on the primal side. Thus, we decided to run again one of our relax-and-cut algorithms to attempt to obtain better primal bounds. In order to do it, we rerun PR&C(CD,LD) with different settings of parameters. Thus, we adopted: $\Delta = 25$

represent the size of the union of the shores in each instance. Therefore they are the complements (with respect to the number of vertices) of the separator size, which are the values reported in [6].

(number of passes of the PR&C algorithm) and π^k (see 9) was updated at each 50 consecutive iterations without improvement on the upper bound and the maximum number of iterations was limited to 5000. Besides, at each SM iteration, the primal heuristic was called at most four times, stopping at the first call with success in obtaining an improvement on the best primal bound known so far. At each call, a distinct choice of the parameter $\bar{\rho}$ (see Figure 4) that controls the weighting method is performed. After some preliminary tests, the final sequence of calls adopted was: $\bar{\rho} = \langle 1, 0 \rangle$, $\bar{\rho} = \langle 0, 1 \rangle$, $\bar{\rho} = \langle 1, 1 \rangle$ and $\bar{\rho} = \langle 0, 0 \rangle$. The results obtained are shown in Figures 14 and 15.

The graphic in Figure 14 compares the results produced by our heuristic against de Souza and Balas' primal results reported in [12]. Since the running times of `PR&C(CD,LD)` remained below 90 seconds for all instances used in this test, we imposed this time limit to the execution of de Souza and Balas' algorithm. The graphic shows, for each instance, how much better/worse is the best solution found by both approaches when compared to the bounds reported in [6]. Thus, a point above x axis represents a better solution than the best we know from literature. Observe that, for this subset of instances, the solutions corresponding to our Lagrangian heuristic are, in general, much better and almost dominate those from the heuristic embedded in de Souza and Balas' algorithm. It is worth noting that in 83% of the cases our heuristic produced results as good as the best VSP solution already reported.

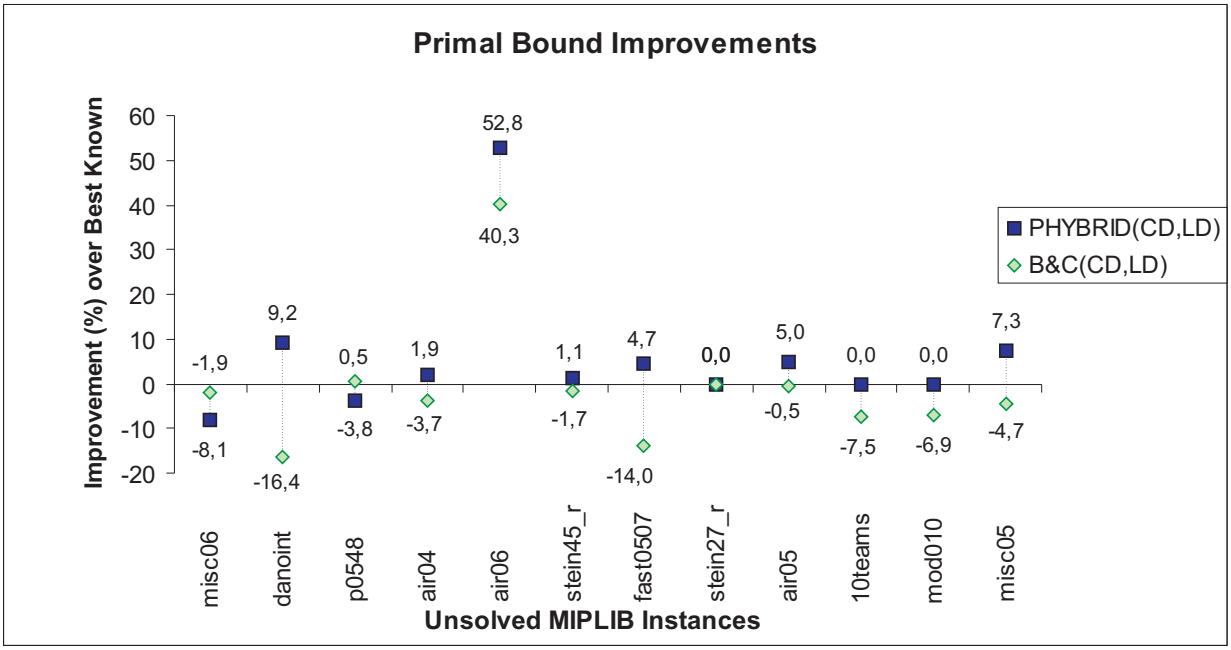

Figure 14: Performance of the heuristics embedded in PHYBRID(CD,LD) and B&C from [12] for hard MIPLIB instances within a 90 seconds running time limit.

Now, regarding the execution time to reach the best solution, we build up the graphic shown in Figure 15. Observe that for all instances but two, `air04` and `air05`, our Lagrangian heuristic needed less time than de Souza and Balas' heuristic to yield their best solutions.

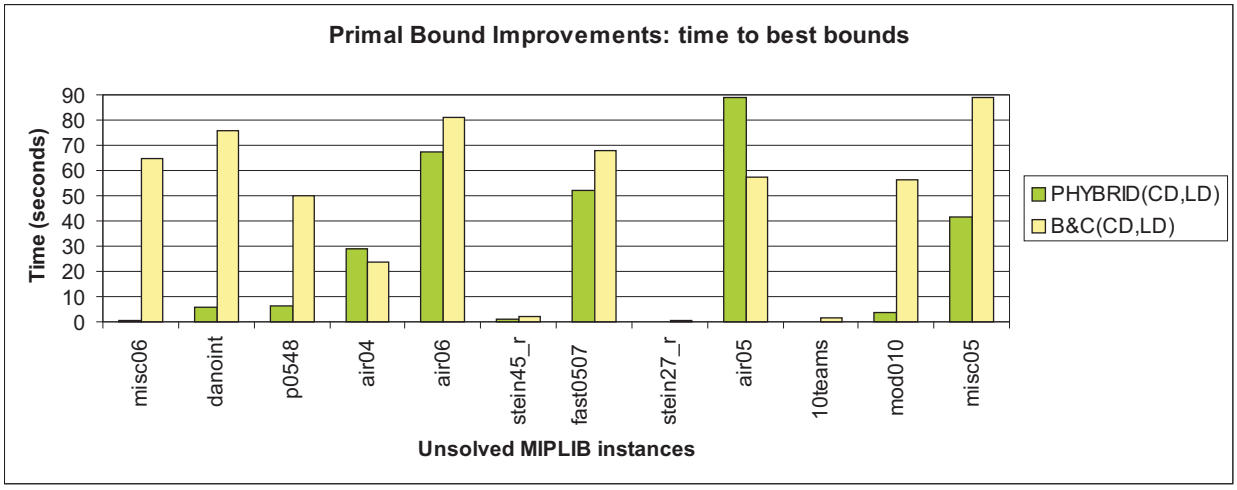

Figure 15: Computation times required by PHYBRID(CD,LD) and by B&C(CD,LD) to produce their best primal bounds.

8 Conclusions and future works

In this paper we investigated the combined usage of Lagrangian relaxation and cutting planes in the development of exact algorithms for the vertex separator problem. Though the pure relax-and-cut algorithms resulting from this combination are usually not strong enough to reach this goal, they proved to be a very effective preprocessing tool for a hybrid exact algorithm. In this algorithmic framework, generically named here as HYBRID, the Lagrangian algorithms are succeeded by a branch-and-cut algorithm. The latter is fed by invaluable outcomes from the (Lagrangian) preprocessing phase. This includes strong primal bounds and cutting planes separated during the relax-and-cut algorithms and corresponding to valid inequalities for the VSP presented in [3].

Computational results were obtained for benchmarks from the literature and compared with the best known results published so far. These experiments show that the best variants of the HYBRID method we developed outperform the pure B&C algorithm introduced by de Souza and Balas in [12], to our knowledge the best exact algorithm available for the VSP. For mid-high density instances, the most difficult ones for the VSP, our algorithms beat the best branch-and-bound code in 92% of the cases tested.

Besides, we show that the Lagrangian phase is a very effective heuristic for the VSP, often producing optimal solutions extremely fast. Moreover, for the MIPLIB instances whose optimal still remains unknown to date, our Lagrangian heuristic in most cases obtained stronger primal bounds than those reported earlier in the literature.

Further developments and implementation issues should be considered to possibly improve the performance of our current framework. This includes the study of different relaxations, the design of more sophisticated primal heuristics and the identification of new valid inequalities for the VSP polytope discussed in [3] to be used as cutting planes in the *relax-and-cut* algorithms.

Acknowledgements. The first author was supported by a doctoral fellowship from CAPES (Brazilian Ministry of Education). The second author was partially supported by grants 301732/2007-8,

472504/2007-0 and 473726/2007-6 from CNPQ (National Council for Scientific and Technological Development from the Brazilian Ministry of Science and Technology). The research also benefited from the *Academic Partnership Program* from DASH OPTIMIZATION, developer of the XPRESS software.

References

[1] T. P. Bagchi. Pareto-optimal solutions for multi-objective production scheduling problems. *Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science*, 1993/2001:458–471, 2001.

[2] E. Balas and N. Christofides. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming*, 21:19–46, 1981.

[3] E. Balas and C. C. de Souza. The vertex separator problem: a polyhedral investigation. *Mathematical Programming*, 103:583–608, 2005.

[4] E. Balas and C. Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization. *Mathematical Programming Study*, 12:37–60, 1980.

[5] J.E. Beasley. *Modern Heuristic Techniques for Combinatorial Problems*, chapter 6, pages 243–303. Mc-Graw-Hill Book Company Europe, London, 1995.

[6] R. Borndörfer, C.E. Ferreira, and A. Martin. Decomposing matrices into blocks. *SIAM Journal on optimization*, 9:236–269, 1998.

[7] F. Calheiros, A. Lucena, and C. C. de Souza. Optimal rectangular partitions. *Networks*, 41:51–67, 2003.

[8] V. Cavalcante and C. C. de Souza. Lagrangian relaxation and cutting planes for the vertex separator problem. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE)*, pages 471–482, 2007.

[9] V. Cavalcante, C. C. de Souza, and A. Lucena. A relax-and-cut algorithm for the set partitioning problem. *Computers & Operations Research*, 35:1963–1981, 2008.

[10] A.S. Cunha. *Árvores ótimas em grafos: modelos, algoritmos e aplicações*. PhD thesis, Engenharia de Sistemas e Computação, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, May 2006. In Portuguese.

[11] A.S. Cunha and A. Lucena. Lower and upper bounds for the degree-constrained minimum spanning tree problem. *Networks*, 50(1):55–66, 2007.

[12] C. C. de Souza and E. Balas. The vertex separator problem: algorithms and computations. *Mathematical Programming*, 103:609–631, 2005.

[13] M.F.F. Filho. *GRASP e Busca Tabu Aplicados a Problemas de Programação de Tarefas em Máquinas Paralelas*. PhD thesis, Departamento de Engenharia de Sistemas, Faculdade de Engenharia Elétrica e Ciência da Computação, University of Campinas (UNICAMP), October 2007. In Portuguese.

[14] M.R. Garey and D.S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H. Freeman and Co., New York, 1979.

[15] B.L. Golden and W. R. Stewart. Empirical analysis of heuristics. In E.L. Lawler, J. K. Lenstra, A.H.G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*, chapter 7, pages 207–250. John Wiley and Sons, Chichester, NY, Brisbane, Toronto, 1985.

[16] M. Guignard. Lagrangean relaxation. In M.A. Lòpez-Cerdá and I. García-Jurado, editors, *Top*, volume 11, pages 151–200. Springer Berlin/ Heidelberg, 2003.

[17] G. Kliewer and L. Timajev. Relax-and-cut for capacitated network design. In *Proceedings of 13^{th} Annual European Symposium on Algorithms (ESA)*, volume 3669, pages 47–58, Mallorca, Spain, October 2005.

[18] A. Lucena. Steiner problem in graphs: Lagrangean relaxation and cutting-planes. In *COAL Bulletin*, volume 21. Mathematical Programming Society, 1992.

[19] A. Lucena. Non delayed relax-and-cut algorithms. *Annals of Operations Research*, 140(1):375–410, 2005.

[20] C. Martinhon, A. Lucena, and N. Maculan. Stronger k-tree relaxations for the vehicle routing problem. *European Journal on Operational Research*, 158:56–71, 2004.

[21] R.Z. Ríos-Mercado and J. F. Bard. An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. *Journal of Heuristics*, 5:53–70, 1999.

A The complexity of the separation of CD inequalities

Let LRP be the Lagrangian subproblem resulting from the relaxation of the set of constraints (2).

Proposition A.1 *Let $G = (V, E)$ be the input graph for a VSP instance and $\bar{u}$ an optimal value solution to LRP. Further, let $S \subseteq V$ be the set of vertices $i \in V$ such as $\bar{u}_{i1} + \bar{u}_{i2} = 1$ and $G[S]$ the subgraph induced by S in G. Then, there exists $W \subset V$ whose* CD *inequality is violated by $\bar{u}$ if and only if S is a dominator and $G[S]$ is connected.*

Proof: ($\Rightarrow$) *Suppose that there exists a* CD *inequality violated by $\bar{u}$. Let W be the connected dominator associated to this inequality. Then, $\bar{u}_{i1} + \bar{u}_{i2} = 1$ must hold for every $i \in W$. Thus, $u(W) = |W|$ and, by construction of S, every vertex in W is also in S. Hence, $W \subseteq S$ and S is a dominator too.*

Let us suppose that S is disconnected. Indeed, given that $W \subseteq S$ and W is connected, there is $v \in S \backslash W$ with no adjacent vertex in W. Hence, we arrive to a contradiction because W is a dominator.

($\Leftarrow$) *Do $W = S$* $\square$

Corollary A.1 *The separation of* CD *inequalities over $G[S]$ has polynomial time complexity when the constraints $u_{i1} + u_{i2} \leq 1, \forall\ i \in V$ are kept in LRP.*

Notice that, if constraints (1) are relaxed, an optimal solution $\bar{u}$ of LRP may assign the same vertex i in two distinct sets. Figure 16 depicts a counterexample for Proposition A.1 in this case. In this picture if i is gray then $\bar{u}_{i1} = \bar{u}_{i2} = 1$, otherwise, $\bar{u}_{i1} = \bar{u}_{i2} = 0$. Observe that, together, gray and black vertices form a connected dominator whose corresponding CD inequality is violated although $G[S]$ is disconnected.

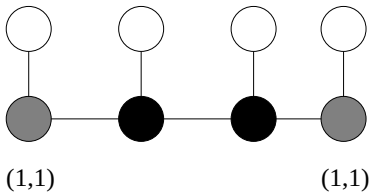

Figure 16: Counterexample for Proposition A.1 when constraints (1) are relaxed.

Appendix B presents a description of a polynomial time dynamic programming algorithm that solves the separation problem of CD inequalities in polynomial time when the constraints $u_{i1} + u_{i2} \leq 1, \forall i \in V$ are satisfied.

Corollary A.2 *Let S be the set of vertices such that $\bar{u}_{i1} + \bar{u}_{i2} \geq 1, i \in V$, and $G[S]$ be the subgraph induced by S in G. Then, if S is a dominator and $G[S]$ is connected, the* CD *inequality associated to S is violated by $\bar{u}$.*

This result shows that our separation routine, described in Section 4.1.1 can only be viewed as an heuristic to separate CD inequalities. In order to analyse the complexity of the separation problem, let us rewrite the CD inequality associated to a connected dominator W of G as follows:

$$u(W) \leq |W| - 1 \ \equiv \ u(W) < |W| \ \equiv \ \sum_{i \in W} (u_{i1} + u_{i2}) < |W|.$$

Therefore, if $\bar{u}$ violates the CD inequality to W, we must have $\sum_{i \in W}(u_{i1} + u_{i2}) \geq |W|$.

Now, consider the following minimum weighted connected dominator problem of a graph G:

INSTANCE: Undirected graph $G = (V, E)$ and weight $w_i \in \mathbb{Z}$ for all vertex $i \in V$. PROBLEM *mWCD (Minimum Weighted Connected Dominator)*: Find a connected dominator W whose weight, given by $\sum_{i \in W} w_i$, is minimum.

Taking $w_i = \bar{u}_{i1} + \bar{u}_{i2}$ for all $i \in V$, we have that the separation problem of CD inequalities can be solved through the optimization problem just described. If the optimal value is equal to or larger than $|W|$, a CD inequality violated is found. Otherwise, all the CD inequalities are satisfied by $\bar{u}$.

From now on, we focus on the decision version of *mWCD* where the additional integer value k is given as input and the problem is to decide if there exists a connected dominator weighting at most k in G. Besides, restricting the possible values of w_i to the subset $\{0, 1, 2\}$, we obtain exactly the case corresponding to the separation of CD inequalities.

Consider the decision problem below.

INSTANCE: Undirected graph $G = (V, E)$, $\|V\| = n$ and a positive integer k. PROBLEM *MLST (Maximum Leaf Spanning Tree)*: Find a spanning tree T for G such as at least k' vertices of T have degree one, i.e., are leaves ?

Fact A.1 *MLST is $\mathcal{NP}$-complete (see [ND2] in [14]).*

Proposition A.2 *mWCD is $\mathcal{NP}$-complete.* **Proof:** *It is easy to see that mWCD is in $\mathcal{NP}$. Now, we show that MLST α_p mWCD.*

Take $k = n - k'$ in mWCD and set $w_i = 1$ for all $i \in V$. So, in mWCD, we look for a connected dominator with no more than $n - k'$ vertices. Notice that, by removing the leaves from any spanning tree T, the remaining vertices form a set that is a connected dominator of G. Hence, if T is a solution to MLST, at least k' vertices are leaves in T and the remaining vertices form a connected dominator of size at most $n - k' = k$. On the other hand, if W is a solution to mWCD, than, there is a spanning tree T' in $G[W]$. Since W is a dominator, any of the $n - |W|$ vertices in $V \setminus W$ is linked to W through at least one edge. By adding to T' one of these edges for each of the $n - |W|$ vertices , we obtain a spanning tree for G with at least $n - |W| \geq n - k = k'$ leaves. This completes the proof. □

This proof shows that, in general, *mWCD* is *$\mathcal{NP}$-complete*, suggesting that the separation problem is equally hard. However, we saw previously that the separation problem may be solved in polynomial time when the corresponding *mWCD* has binary vertex weights and $k = |W|$. Also, the latter proof does not permit us to conclude that *mWCD* remains hard to solve when the vertex weights are in $\{0, 1, 2\}$ and k is equal to $|W|$. This is precisely the case of the separation problem relative to the solution $\bar{u}$ of the Lagrangian relaxation when the inequalities $\bar{u}_{i1} + \bar{u}_{i2} \leq 1$, $i \in V$, are dualized.

Denote by *mCD* the special case of *mWCD* where $w_i = 1$ for all $i \in V$. According to the proof of Proposition A.2, *mCD* is *$\mathcal{NP}$-complete*. Furthermore, observe that, when G has a connected dominator of size $p \leq k$, G also has a connected dominator of size l, where $p \leq l \leq n$. Hence, the

problem of determining if there is a connected dominator of size k, named $mCD^{=}$ below, is also *$\mathcal{NP}$-complete.*

INSTANCE: Undirected graph $G = (V, E)$ and a positive integer k. PROBLEM *$mCD^{=}$ (Minimum Connected Dominator)*: Find a connected dominator $W \subseteq V$ in G of size k?

Now, consider the separation of `CD` inequalities over a graph G with weights on the vertices restricted to $\{0, 1, 2\}$ and $k = |W|$. This problem can be solved by the optimization problem associated to the following decision problem:

INSTANCE: Undirected graph $G = (V, E)$, $\|V\| = n$, weight $w_i \in \{0, 1, 2\}$ for all vertices $i \in V$ and a positive integer $t \leq n$. PROBLEM *SEP*: Find a connected dominator W in G satisfying $\sum_{i \in W} w_i = t \geq |W|$?

Fact A.2 *$mCD^{=}$ is $\mathcal{NP}$-complete.*

Proposition A.3 *SEP is $\mathcal{NP}$-complete.*

Proof: *It is not difficult to see that SEP belongs to $\mathcal{NP}$. We now show that $mCD^{=} \leq_P$ SEP, which proves that $mCD^{=}$ is also $\mathcal{NP}$-hard.*

Figure 17 illustrates a polynomial time transformation of an arbitrary instance $I(mCD^{=})$ of $mCD^{=}$ into an instance $I(SEP)$ of SEP. The figure shows details concerning the graph vertices, edges and weights besides the expression that defines the input parameter t.
(I): *$mCD^{=}$ YES $\Rightarrow$ SEP YES. Let $W \subseteq V$ be a connected dominator of G such that $|W| = k$. Without loss in generality, we can suppose that $W = \{v_1, v_2, \ldots, v_k\}$ (black vertices in Figure 17). Further, let $W' \subset V'$ be given by $W' = W \cup A \cup \{b_1, c_1\} \cup \{c_{k+1}, \ldots, c_n\} \cup \{d\}$ and denote $p(W') = \sum_{i \in W'} w_i$. Thus, based on the transformation of $I(mCD^{=})$ into $I(SEP)$ we can conclude that:*

Fact A.3 *Since $G[W]$ is a connected graph, $G'[W']$ is also a connected graph.*

Fact A.4 *W' is a dominator because: (i) all the vertices of V are covered by W; (ii) d is in W'; (iii) every vertex $i \in A \cup \{c_{k+1}, \ldots, c_n\}$ is in W'; (iv) b_1 and c_1 are both in W'; (v) $\{b_2, \ldots, b_k\}$ are covered by W while $\{b_{k+1}, \ldots, b_n\}$ are covered by $\{c_{k+1}, \ldots, c_n\}$; (vi) $\{c_2, \ldots, c_k\}$ are covered by d and (vii) $\{e\}$ is covered by a_{n-k+2}.*

Fact A.5 *$t \geq |W'|$ is satisfied since:*

$$\begin{aligned} p(W') &= \overbrace{k}^{W} + \overbrace{2(n-k+2)}^{A} + \overbrace{0}^{\{b_1\}} + \overbrace{0}^{\{c_1\}} + \overbrace{0}^{\{c_{k+1},\ldots,c_n\}} + \overbrace{1}^{\{d\}} = 2n-k+5, \\ |W'| &= \underbrace{k}_{W} + \underbrace{(n-k+2)}_{A} + \underbrace{1}_{\{b_1\}} + \underbrace{1}_{\{c_1\}} + \underbrace{n-k}_{\{c_{k+1},\ldots,c_n\}} + \underbrace{1}_{\{d\}} = 2n-k+5, \end{aligned}$$

and, therefore,

$$t = p(w') = \sum_{i \in W'} w_i = |W'| = 2n - k + 5.$$

$I(mCD^{=})$: $G = (V, E)$
$I(SEP)$: $G' = (V', E')$

$A = \{a_1, a_2, \ldots, a_{n-k+2}\}$
$B = \{b_1, b_2, \ldots, b_n\}$
$C = \{c_1, c_2, \ldots, c_n\}$
$V = \{v_1, v_2, \ldots, v_n\}$

$V' = A \cup B \cup C \cup V \cup \{d\} \cup \{e\}$
$|V'| = 4n - k + 4$

$w_{a_j} = 2,\ \forall j = 1, \ldots, n-k+2$
$w_{b_i} = w_{c_i} = 0,\ w_{v_i} = 1,\ \forall i = 1, \ldots, n$
$w_d = 1$
$w_e \in \{0, 1, 2\}$

$t = 2n - k + 5$

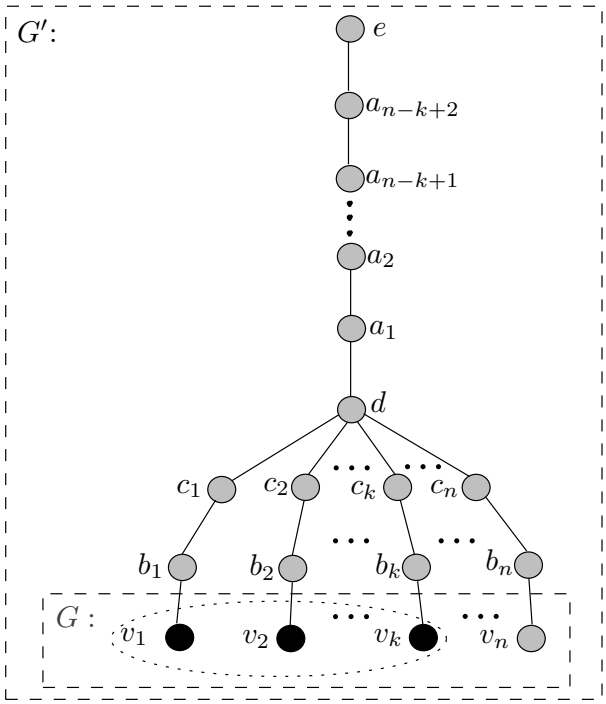

Figure 17: Transformation of an instance of $mCD^{=}$ into an instance of SEP.

We conclude that if $I(mCD^{=})$ is a YES instance for $mCD^{=}$, $I(SEP)$ is a YES instance for SEP.

(II): *SEP YES $\Rightarrow$ $mCD^{=}$ YES. Let us suppose that W' is a connected dominator of G' satisfying $p(W') = t = 2n - k + 5 \geq |W'|$. First notice that the vertices in $\{d, a_1, \ldots, a_{n-k+2}\}$ belong to any connected dominator of G'. We want to show that $W \subseteq V$ is a connected dominator of size k. To this end, suppose that $W = W' \cap V = \{V_1, V_2, \ldots, V_p\}$ where every V_i is a connected component of G.*

Observe that $p(W') = |W| + 2|A| + |d| = 2n - k + 5 = k + 2(n - k + 2) + 1$. So, W must have exactly k vertices and $|W'|$ cannot contain more than $n - k + 2$ vertices with zero weight without violating $p(W') \geq |W'|$. Without loss in generality, assume that $V \backslash W = \{v_{k+1}, \ldots, v_n\}$. Vertices $\{b_{k+1}, \ldots, b_n\}$ cannot be in W' since, otherwise, we would not have W' connected and satisfying $p(W') \geq |W'|$ simultaneously. Thus, the $n - k$ vertices of $\{c_{k+1}, \ldots, c_n\}$ are in W'. Now, every component of G induced by vertices in W requires a pair of vertices $\{b_i, c_i\}$ as connector to the other vertices of W'. Indeed, we need at least $2p$ more vertices (2 for each connected component of G) in W', beyond the $2n - k + 3$ already identified, to guarantee that W' remains connected. But, since the size of W' is limited to $2n - k + 5$, we are left only with two more vertices to put into W'. Hence, W has a single connected component ($p = 1$), which must cover all the uncovered vertices of V. Consequently, W is a connected dominator of G with k vertices. □

B A Dynamic Programming to CD inequalities

The dynamic programming algorithm described in Figure 18 solves the separation problem of CD inequalities under the circumstances announced in corollary A.1.

DP Description (LRP') /* $LRP' \equiv LRP$, where (1) is satisfied */

1. Definitions:

   z: matrix $(n+1) \times (b+1) \times (b+1)$, $n = |V|$, keeping the subproblem solutions.

   $z[k,p,q]$: stores the objective function maximum value considering that vertices $k+1$ to n belong to separator C and that exactly p vertices are in subset A and that q vertices are in subset B.

   c_{k1}, c_{k2}: costs for adding vertex k to A and B, respectively.

2. Initializations: suppose that, for all $j = 1, 2$, $c_{\varphi(i)j} \geq c_{\varphi(i+1)j}$, for all i in V, i.e., $\{\varphi(1), \varphi(2), \ldots, \varphi(n)\}$ corresponds to the sequence of vertices non-increasingly sorted by the costs $c_{.j}$.

   $z[0,0,0] = z[k,0,0] \equiv 0$, $\forall k$.

   $$z[k,p,0] = \begin{cases} 0, & \text{if } p > k \\ \sum_{i=1}^{\min\{p,k\}} c_{\varphi(i)1}, & \text{otherwise.} \end{cases}$$

   $$z[k,0,q] = \begin{cases} 0, & \text{if } q > k \\ \sum_{i=1}^{\min\{q,k\}} c_{\varphi(i)2}, & \text{otherwise.} \end{cases}$$

   $z[k,p,q] \equiv -\infty$, when $p < 0$ or $q < 0$ or $p + q > k$.

3. Recurrence (subproblem solutions):

   $z[k,p,q] \equiv \max\{z[k-1,p,q], c_{k1} + z[k-1,p-1,q], c_{k2} + z[k-1,p,q-1]\}, \forall k,p,q$.

4. Optimal solution value:

   $z^* = \max\{z[|V|,p,q]\}$, $1 \leq p \leq b$ e $1 \leq q \leq b$.

Figure 18: Dynamic Programming description.

C Computational results on unsolved instances

Table 8 shows some results obtained with two of our best versions of relax-and-cut frameworks over the set of 11 instances not solved by any of the implemented algorithms within the time limit of 30 minutes. In order to make possible performance comparisons with possible future algorithms to the VSP the table shows final upper bounds, lower bounds and values corresponding to the number of nodes produced by the B&C enumeration tree.

Table 8: Results for VSP unsolved instances.

Instance				NDHYBRID(CD)			PHYBRID(CD)		
label	n	d	Opt	ub	lb	nodes	ub	lb	nodes
dim.DSJC125.1	125	0.09	90	102.26	89	143809	102.38	88	124345
dim.queen12_12	144	0.25	97	113.69	97	19518	114.12	97	16062
dim.queen11_11	121	0.27	81	91.64	81	37586	91.76	81	32499
dim.queen10_10	100	0.30	67	72.29	67	88727	72.28	67	81052
dim.queen8_12	96	0.30	65	69.56	65	96996	69.60	65	84300
dim.queen9_9	81	0.33	55	56.49	55	231242	56.44	55	202559
dim.DSJC125.5	125	0.50	74	88.91	74	6208	87.65	74	5329
mat.L125.can_161	125	0.16	97	105.37	95	73721	105.37	97	52825
miplib.stein27_r.p	118	0.32	62	82.00	62	18801	83.21	62	22951
miplib.10teams.p	210	0.34	120	188.67	120	1824	190.38	120	1768
miplib.mod010.p	146	0.38	90	97.29	88	16171	96.21	86	10965