

Please cite this paper as follows:

Fabien Cornillier, José Eduardo Pécora Jr, Vincent Charles, A variable depth search branching, Operations Research Letters (2012),
<http://dx.doi.org/10.1016/j.orl.2012.03.003>

A variable depth search branching

Fabien Cornillier^{a,b,*}, José Eduardo Pécora Jr.^c, Vincent Charles^a,

^a*CENTRUM Católica, Graduate School of Business, Pontificia Universidad Católica del Perú, Jr. Alomía Robles 125-129, Los Alamos de Monterrico, Lima, Peru*

^b*Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada*

^c*General and Applied Business Department (DAGA), Universidade Federal do Paraná, Curitiba, Paraná, Brazil*

Abstract

We introduce a *variable depth search branching*, an extension to the local branching for solving Mixed-Integer Programs. Two strategies are assessed, a best improvement strategy and a first improvement strategy. The extensive computational assessment evidences a significant improvement over the local branching for both strategies.

Keywords: Mixed integer programming, local branching, variable depth search, heuristic

1. Introduction

This article proposes a heuristic based on the variable depth search (VDS) algorithm that can be viewed as a sequence of local branchings [1] to improve primal solutions of 0-1 mixed integer linear programming problems.

Kernighan and Lin [2] initially introduced the VDS heuristic to solve the graph partitioning problem. The VDS heuristic was later applied to the traveling salesman problem by Christofides and Eilon [3], and then by Lin and Kernighan [4]. It has also been successfully used in many other problems including the generalized assignment problem [5, 6], the single-vehicle dial-a-ride problem with time windows [7], the job shop scheduling [8], the multiprocessor flow shop scheduling problem [9], the unconstrained binary quadratic programming problem [10], and frequency allocation problems in telecommunications [11, 12]. Unfortunately, efficient algorithms based on the VDS are not always straightforward to implement, particularly when no time-effective local search algorithm is known, when feasibility checks have to be performed, or when there is no simple way to compute the differences in objective value. It has been shown by Fischetti and Lodi [1] that a generic mixed integer programming solver used as a black-box could be of interest in these cases.

The remainder of this article is organized as follows. In Section 2 the VDS heuristic is briefly presented. The VDS branching sequence is introduced in Section 3, and a complete VDS branching heuristic framework is proposed in Section 4. The results of the numerical tests undertaken to assess the performance of the proposed algorithm are presented in Section 5.

2. The VDS heuristic

Applied to the traveling salesman problem, the k -opt local search algorithm consists in deleting k edges of a tour to find the best way to reconnect it, namely the k -optimal solution [13, 14]. Observing that this algorithm can be rapidly impracticable with increasing values of k , Kernighan and Lin [2] proposed a VDS heuristic using a sequence of k -opt exchanges with practicable values $k < K$ to approximate a larger K -opt exchange. Considering a combinatorial programming problem with binary variables, the VDS can be summarized as follows: from an initial feasible solution, a sequence of moves is attempted, each move consisting of complementing the value of a maximum of k binary variables selected among those which have never been complemented from the beginning of the sequence. Indeed, within a same sequence, each time a binary variable is complemented it is set aside and cannot be considered again until the termination of the sequence. A sequence is usually terminated either when the cumulative objective improvement becomes negative [2, 4] or when a given number of subsequent non-improvement moves have been performed. This procedure is repeatedly applied from the best feasible solution obtained in the last sequence.

The following section describes the VDS branching sequence with a simple illustration. This branching sequence is integrated into the complete VDS branching heuristic framework in Section 4.

3. The VDS branching sequence

Consider a 0-1 mixed integer linear program:

$$(P) \quad \min z(x) = c^T x$$

subject to

$$Ax \geq b,$$

*Corresponding author

Email addresses: fcornillier@pucp.pe (Fabien Cornillier), pecora@ufpr.br (José Eduardo Pécora Jr.), vcharles@pucp.pe (Vincent Charles)

$$\begin{aligned}
x_i &\in \{0, 1\}, & \forall i \in \mathcal{B} \neq \emptyset, \\
x_i &\geq 0, x_i \in \mathbb{N}, & \forall i \in \mathcal{G}, \\
x_i &\geq 0, & \forall i \in \mathcal{C},
\end{aligned}$$

where \mathcal{B} is a non-empty index set of binary variables, and \mathcal{G} and \mathcal{C} are possibly empty index set of integer variables and continuous variables, respectively.

If \bar{S} is the binary support of an incumbent solution \bar{x} , i.e., $\bar{S} = \{i \in \mathcal{B} : \bar{x}_i = 1\}$, and R is a subset of the index set of binary variables, the k -opt neighborhood restricted to the subset R of a solution \bar{x} is defined as the set of all the feasible solutions of (P) satisfying the following additional constraint:

$$\Delta^R(x, \bar{x}) \leq k,$$

where

$$\Delta^R(x, \bar{x}) = \sum_{i \in R \cap \bar{S}} (1 - x_i) + \sum_{i \in R \setminus \bar{S}} x_i,$$

is the Hamming distance restricted to the subset $R \subseteq \mathcal{B}$ between a solution x and the incumbent solution \bar{x} .

Given a feasible solution x^0 , the local branching consists in an attempt to find an improving sequence of incumbent solutions x^1, x^2, \dots, x^n , where $x^h, h \in \{1, 2, \dots, n\}$, is an optimal solution of (P) within the k -opt neighborhood of the solution x^{h-1} , and outside of the k -opt neighborhoods of the solutions x^0, x^1, \dots, x^{h-2} , i.e., x^h is an optimal solution of (P) within the following set of new linear constraints:

$$\begin{aligned}
\Delta(x, x^0) &> k \\
\Delta(x, x^1) &> k \\
&\vdots \\
\Delta(x, x^{h-2}) &> k \\
\Delta(x, x^{h-1}) &\leq k.
\end{aligned}$$

It is worth noting that $\Delta(x, x^h)$ represents the absolute Hamming distance between solutions x and x^h , contrary to $\Delta^R(x, x^h)$ which represents the Hamming distance restricted to R . While only the absolute Hamming distance is used in the local branching, the VDS branching makes use of the restricted Hamming distance to limit to k the number of complemented binary variables in R , and to prevent any complementation of the remaining binary variables. At each step of the sequence the Hamming distance is restricted to the subset $\mathcal{B} \setminus T$, where T is the index set of all the binary variables that have been complemented between the beginning of the sequence and the current improved solution. In the VDS branching, x^h is an optimal solution of (P) within the following set of linear constraints:

$$\Delta^{\mathcal{B}}(x, x^0) > k \quad (1)$$

$$\Delta^{\mathcal{B} \setminus T^1}(x, x^1) + M\Delta^{T^1}(x, x^1) > k \quad (2)$$

\vdots

$$\Delta^{\mathcal{B} \setminus T^{h-2}}(x, x^{h-2}) + M\Delta^{T^{h-2}}(x, x^{h-2}) > k \quad (3)$$

$$\Delta^{\mathcal{B} \setminus T^{h-1}}(x, x^{h-1}) + M\Delta^{T^{h-1}}(x, x^{h-1}) \leq k, \quad (4)$$

where M is an sufficiently large positive number. Constraints (1) to (3) prevent any solution within the already searched k -opt neighborhoods of the solutions x^0, x^1, \dots, x^{h-2} , while the local branching constraint (4) restricts the search to the k -opt neighborhood of the last incumbent solution x^{h-1} . The first part of constraints (2) to (4) limits to k the number of binary variables in $\mathcal{B} \setminus T^{h-1}$ to be complemented, and the last part prevents any complementation of the already complemented binary variables.

Whenever an improved k -optimal solution is found, the incumbent solution is updated, the last local branching constraint is reversed, a new local branching constraint based on the incumbent solution is added, and the set T is updated.

Illustration with $k = 2$

Figure 1 illustrates the case of a pure binary program with nine binary variables. The sequence starts with a feasible incumbent solution x^0 and an empty index set T^0 of complemented binary variables. A first improvement is obtained by solving the program within the 2-opt neighborhood of x^0 . The improved solution x^1 is found by complementing the first binary variable, giving a new index set $T^1 = \{1\}$ (boxed variables in Figure 1). A second improvement is obtained by solving the program within the 2-opt neighborhood of x^1 , but outside of the 2-opt neighborhood of x^0 , i.e., without considering the variables corresponding to the index set T^1 . The improved solution x^2 is found by complementing the fifth binary variable, giving a new set $T^2 = \{1, 5\}$ (in Figure 1, new complemented variables are boxed, and already complemented variables are shadowed). Finally, a third improved solution x^3 is found by solving the program within the 2-opt neighborhood of x^2 without considering the binary variables corresponding to $T^2 = \{1, 5\}$. The new set $T^3 = \{1, 5, 6, 7\}$ is then created.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
$x^0 = \{$	0	0	1	0	0	1	0	0	0	$\}$
$x^1 = \{$	1	0	1	0	0	1	0	0	0	$\}$
$x^2 = \{$	1	0	1	0	1	1	0	0	0	$\}$
$x^3 = \{$	1	0	1	0	1	0	1	0	0	$\}$

Figure 1: Illustration of the elementary complementation procedure

In this example, finding x^1 consists in solving (P) within the 2-opt neighborhood of x^0 , i.e., by adding the local branching constraint $\Delta^{\mathcal{B} \setminus T^0}(x, x^0) + M\Delta^{T^0}(x, x^0) \leq 2$ to (P). Because T is initially empty this constraint could be written as $\Delta^{\mathcal{B}}(x, x^0) \leq 2$.

To find the solution x^2 , we start from the incumbent solution x^1 to solve (P) while reversing the last local branching constraint and adding a new one corresponding to the 2-opt restricted neighborhood of x^1 :

$$\Delta^{\mathcal{B}}(x, x^0) > 2$$

$$\Delta^{\mathcal{B} \setminus T^1}(x, x^1) + M\Delta^{T^1}(x, x^1) \leq 2.$$

In the same manner, the solution x^3 is found by solving (P) with the following set of additional constraints:

$$\Delta^{\mathcal{B}}(x, x^0) > 2$$

$$\Delta^{\mathcal{B} \setminus T^1}(x, x^1) + M\Delta^{T^1}(x, x^1) > 2$$

$$\Delta^{\mathcal{B} \setminus T^2}(x, x^2) + M\Delta^{T^2}(x, x^2) \leq 2.$$

4. The VDS branching framework

The proposed branching framework is based on the VDS branching sequence presented in the previous section. Without any imposed time limit or upper cutoff value to solve a subproblem, the improved solutions are always k -optimal. In practice however, a time limit and an upper cutoff value are generally imposed and the solver may end with one of four different status: optimum, proven infeasible, feasible, or no feasible. These four states are described in the pseudo-code of Figure 2, in a similar presentation as the local branching pseudo-code presented in [1] to allow an easier comparison. In Figure 2, the deviations from the local branching algorithm are shadowed and numbered.

```

function VariableDepthSearchBranching( $k, totalTimeLimit, nodeTimeLimit, dvMax, x^*$ )
begin
   $rhs := bestUB := UB := TL := +\infty; x^* := \text{undefined}$ 
   $opt := first := \text{true}; dv := 0; diversify := \text{false}; T := \emptyset$ 
  repeat
1    if ( $rhs < +\infty$ ) then Add the local branching constraint  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) \leq rhs$ 
       $TL := \min(TL, totalTimeLimit - elapsedTime)$ 
       $stat := MIPSolve(TL, UB, first, \bar{x})$ 
       $TL := nodeTimeLimit$ 
      if ( $stat = optSolFound$ ) then
        if ( $c^T \bar{x} < bestUB$ ) then  $bestUB := c^T \bar{x}; x^* := \bar{x}$ 
        if ( $rhs \geq +\infty$ ) then return  $opt$ 
2      Reverse the last local branching constraint into  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) > rhs$ 
3       $T := T \cup \{i \in B : \bar{x}_i = \tilde{x}_i\}$ 
       $diversify := first := \text{false}; \bar{x} := \tilde{x}; UB := c^T \bar{x}; rhs := k$ 
      if ( $stat = provenInfeasible$ ) then
        if ( $rhs \geq +\infty$ ) then return  $opt$ 
4      Reverse the last local branching constraint into  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) > rhs$ 
        if ( $T \neq \emptyset$ ) then
5           $rhs := k; T := \emptyset$ 
        else
          if ( $diversify$ ) then  $UB := TL := +\infty; dv := dv + 1; first := \text{true}$ 
           $rhs := rhs + \lceil \frac{k}{2} \rceil; diversify := \text{true}$ 
      if ( $stat = feasibleSolFound$ ) then
        if ( $rhs < +\infty$ ) then
6          if ( $first$ ) then
            Delete the last local branching constraint  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) \leq rhs$ 
          else
7            Replace last local branching constraint by  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) \geq 1$ 
          REFINE( $\bar{x}$ )
          if ( $c^T \bar{x} < bestUB$ ) then  $bestUB := c^T \bar{x}; x^* := \bar{x}$ 
8           $T := T \cup \{i \in B : \bar{x}_i = \tilde{x}_i\}$ 
           $first := diversify := \text{false}; \bar{x} := \tilde{x}; UB := c^T \bar{x}; rhs := k$ 
      if ( $stat = noFeasibleSolFound$ ) then
        if ( $T \neq \emptyset$ ) then
9           $rhs := k; T := \emptyset$ 
        else
          if ( $diversify$ ) then
            Replace the last local branching constraint by  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) \geq 1$ 
             $UB := TL := +\infty; dv := dv + 1; rhs := rhs + \lceil \frac{k}{2} \rceil; first := \text{true}$ 
          else
10          Delete the last constraint  $\Delta^{\mathcal{B} \setminus T}(x, \bar{x}) + M\Delta^T(x, \bar{x}) \leq rhs$ 
             $rhs := rhs - \lceil \frac{k}{2} \rceil$ 
             $diversify := \text{true}$ 
11
  until ( $elapsedTime > totalLimitTime$ ) or ( $dv > dvMax$ )
   $TL := totalTimeLimit - elapsedTime; first := \text{false}$ 
   $stat := MIPSolve(TL, bestUB, first, x^*)$ 
   $opt := (stat = optSolFound) \text{ or } (stat = provenInfeasible)$ 
  return  $opt$ 

```

Figure 2: The VDS branching function

The VDS branching function uses four input parameters. The user needs to define the size k of the k -opt neighborhood, the overall time limit $totalTimeLimit$, a time limit $nodeTimeLimit$

to explore each k -opt neighborhood, and a maximal number $dvMax$ of strong diversifications (defined in the next paragraph). The function consists in repetitively solving a subproblem of (P) including a new local branching constraint at each iteration (line 1 of Figure 2), except in the first iteration used to find a first integer solution ($first := \text{true}$), until either the total time limit or the maximum number of diversifications is reached. It returns one of the above said four possible optimization status, and x^* the best integer solution found.

To solve each subproblem, the function MIPSolve is called with a time limit TL , and an upper cutoff value UB to discard any solution with a greater objective value. The parameter $first$ is set to true when no integer solution has been found or in case of a strong diversification which consists in finding an integer solution while relaxing the upper cutoff value UB . In these cases, the function MIPSolve returns the first integer solution found, otherwise it returns the best integer solutions found during the prescribed node time limit TL . The four possible optimization status are as follows:

Proven optimum solution. As an improved solution is proven to be k -optimal, the last local branching constraint is reversed (line 2 of Figure 2), the index of the last complemented binary variables are added to the set T (line 3 of Figure 2), and the upper cutoff value UB and possibly $bestUB$, the best found objective value, are updated.

Proven infeasible. The last local branching is reversed (line 4 of Figure 2). If the algorithm is at the beginning of a VDS branching sequence, a soft or strong diversification is applied according to the flag $diversify$, otherwise T is emptied and the right-hand side rhs is reset to k (line 5 of Figure 2). A diversification (soft or strong) consists in increasing the value of rhs by $\lceil \frac{k}{2} \rceil$, while the upper cutoff value UB is released in a strong diversification ($UB := +\infty$).

Feasible solution. The index of the last complemented binary variables are added to the set T (line 8 of Figure 2). Because the improved solution found is not proven to be k -optimal, the last local branching constraint cannot be reversed, but is replaced by a *tabu* constraint used to cut off the current incumbent solution (line 7 of Figure 2). This *tabu* constraint can only be added when the current incumbent solution is proven to be optimal when fixing its binary variables (line 7 of Figure 2). The function REFINE is used to prove the optimality of the binary solution and consists in solving the current subproblem subject to the constraint $\Delta(x, \bar{x}) \leq 0$. If the optimality of the procedure is not critical, one can deactivate this function which could need excessive CPU time. The last local branching constraint cannot be replaced by a *tabu* constraint in this case.

No feasible solution. If T is not empty (i.e., the algorithm is not at the beginning of a VDS branching sequence), it is emptied, and rhs is reset to k , otherwise an intensification or a strong diversification has to be performed according to the value of the flag $diversify$. The intensification consists in deleting the last local branching constraint and decreasing the value of rhs

by $\lceil \frac{k}{2} \rceil$ to reduce the size of the k -neighborhood. In a strong diversification, the last local branching constraint is replaced by a *tabu* constraint to escape from the current solution, the value of rhs is increased by $\lceil \frac{k}{2} \rceil$, and the upper cutoff value UB is released.

Contrary to the local branching constraint, the VDS local branching constraint does not only depends on the incumbent solution, but also on the index set T of the complemented binary variables (lines 1, 2, 4, 6, 7, 10 and 11 of Figure 2). Each time an improved solution is found, however it is optimal or not, the indices of the new complemented binary variables are added to T and the VDS branching sequence continues (lines 3 and 8 of Figure 2). Otherwise, the current VDS branching sequence ends, and a new sequence is attempted after emptying T and resetting rhs (lines 5 and 9 of Figure 2).

In the same manner as the local branching, the VDS branching can be applied whenever an integer solution is found, including a solution found by the VDS branching itself.

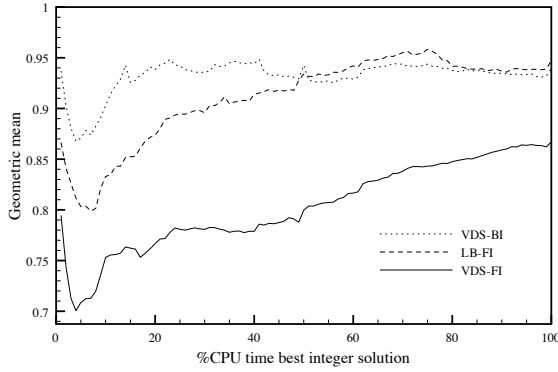


Figure 3: Geometric mean over the 111 instances of the test bed

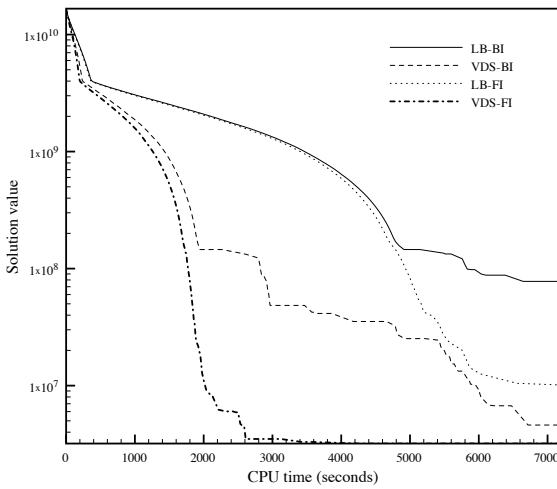


Figure 4: Solving MIP instance dc11

5. Computational results

Two approaches of the local branching and the VDS branching have been compared. In the first approach, called *best improvement* strategy (BI), the function `MIPsolve` returns the best integer solution found within the time limit TL . In the second approach, called *first improvement* strategy (FI), the function `MIPsolve` returns the first improving integer solution. In Fischetti and Lodi [1] only the *best improvement* strategy is used. In this paper, the two approaches have been combined with the local branching, namely LB_{BI} and LB_{FI} , and with the VDS branching, namely VDS_{BI} and VDS_{FI} .

These algorithms were coded in C++ and used the Concert Library of IBM Ilog CPLEX 12.2. They were run on 2.2GHz Dual Core AMD Opteron 275 processor computers with the CentOS 5.3 operating system, and 8Gb RAM. Results were compared on 111 instances of the *challenge* MIPLIB 2010 library [15]. From this library, the instances were retained in the test bed only when a first integer solution has been found in a maximum of two CPU hours time limit, and when a minimum of 5 incumbent integer solutions have been generated during a second period of two CPU hours. All the assessed algorithms started from a same first integer solution, with $k = 20$, $dvMax = +\infty$, and a node time limit of 600 CPU seconds.

CPLEX parameters. The CPLEX MIP emphasize parameter has always been setup to feasibility and optimality, the default option (BALANCED), except when no integer solution has been found, or whenever the algorithm starts a strong diversification, in which case the emphasize has been setup to feasibility over optimality (FEASIBILITY). The CPLEX parallel threads parameter has been set to proceed sequentially and deterministically within a single thread. Remaining parameters have been set to their default value.

Table 1 provides a comparison of VDS_{BI} , LB_{FI} and VDS_{FI} , the assessed algorithms, with LB_{BI} , the benchmark algorithm. Denoting by t_{first} the instant when a first integer solution has been found, and by t_{best} the instant when the best solution has been found either by the assessed algorithm or by the reference algorithm, the results indicate for each assessed algorithm the performance (*perf*) as the ratio, at t_{best} , of the best integer solution value found with the assessed algorithm to the best solution value found with the benchmark algorithm LB_{BI} . For each instance of the test bed, Table 1 also provides the number $\#rows$ of constraints, the total number of variables $\#cols$, the numbers $|G|$ and $|B|$ of integer and binary variables, and the number $\#NZ$ of non-zero variables.

Figure 3 shows the behavior of the geometric mean of each of the three assessed algorithms VDS_{BI} , LB_{FI} , and VDS_{FI} , over the 111 instances, compared to the local branching LB_{BI} , between t_{first} (0%) and t_{best} (100%). This figure shows, on average, consistent improvement over the time for VDS_{BI} , LB_{FI} , and VDS_{FI} , when compared to the LB_{BI} algorithm, with a better improvement shortly after the beginning of the computation. In addition, we can see that the best results are obtained with the VDS branching combined with the *first improvement* strategy.

Figure 4 illustrates the behaviors of the discussed algorithms with the two approaches applied to the dc11 instance. It can be observed that VDS branching with *best improvement* and *first improvement* strategies performs better than its counterparts. It is evidenced from both Figures 3 and 4 that among the two approaches the *first improvement* strategy contributes more towards the performance of VDS branching and local branching.

5.1. VDS branching versus local branching

Table 2 provides the statistics of the performance of VDS_{BI}, LB_{FI} and VDS_{FI} obtained at their respective t_{best} . VDS_{BI} compared to LB_{BI} shows a geometric mean of 0.9393, and a standard deviation of the natural logarithm of 0.3503, with a 95% confidence interval of [0.8800 – 1.0025]. Moreover, with a geometric mean of 0.8153 corresponding to a 18.47% improvement over the 48 reported improvements, compared to a geometric mean of 1.0470 corresponding to a 4.70% deterioration over the 62 reported deteriorations, it can be observed that the magnitude of the improvements is on average 3.93 times greater than the magnitude of the deteriorations.

Table 2: Performance statistics

	VDS _{BI}	LB _{FI}	VDS _{FI}
Geometric mean $perf$	0.9393	0.9386	0.8671
Standard deviation $\ln(perf)$	0.3503	0.3607	0.5695
CI 95% $perf$	[0.8800 – 1.0025]	[0.8871 – 1.0145]	[0.7799 – 0.9640]

VDS_{FI} and LB_{FI} have been compared, and Table 2 shows a significant improvement of the geometric means with 0.8671 for the VDS branching, compared to 0.9386 for the local branching under *first improvement* strategy. A paired sample t -test has been applied on the natural logarithm of their performances, which rejects the null hypothesis that there is no significant difference between the log performance of VDS_{FI} and LB_{FI} at a 95% significance level with a p -value of 0.005. It is clear from the calculated t -value of 2.90 that the VDS branching performs better than the local branching under *first improvement* strategy, with a mean difference of 0.0899.

5.2. Best improvement versus first improvement

In Table 2, LB_{FI} compared to LB_{BI} shows a geometric mean of 0.9386 and a standard deviation of the natural logarithm of 0.3607, with a 95% confidence interval of [0.8871 – 1.0145]. Applied to the VDS branching, the results show a significant improvement of the geometric mean with 0.8671 for the *first improvement* strategy when compared to 0.9393 for the *best improvement* strategy.

A paired sample t -test has been carried out on the natural logarithm of their performances which rejects the null hypothesis that there is no significant difference at a 95% level of confidence, with a p -value of 0.031. The calculated t -value of 2.183 shows that the *first improvement* strategy performs better than the *best improvement* strategy, with a mean difference of 0.0800. Moreover, with a geometric mean of 0.8154

corresponding to a 18.46% improvement over the 47 reported improvements, compared to a geometric mean of 1.00632 corresponding to a 6.32% deterioration over the 61 reported deteriorations, it can be observed that the magnitude of the improvements is on average 2.93 times greater than the magnitude of the deteriorations.

5.3. VDS_{FI} versus LB_{BI}

In Table 2, VDS_{FI} compared to the original local branching shows that VDS_{FI} performs better than the original local branching, with a geometric mean of 0.8671, and a standard deviation of the log performance of 0.5695, with a 95% confidence interval of [0.7799 – 0.9640]. Moreover, with a geometric mean of 0.6773 corresponding to a 32.27% improvement over the 51 reported improvements, compared to a geometric mean of 1.0721 corresponding to a 7.21% deterioration over the 58 reported deteriorations, it can be observed that the magnitude of the improvements is on average 4.48 times greater than the magnitude of the deteriorations.

From the above statistical analysis it can be inferred that both the VDS branching combined with the *best improvement* strategy and the local branching combined with the *first improvement* strategy perform better on average than the local branching combined with the *best improvement* strategy. In addition, it has been shown that the VDS branching performs better than the local branching with both strategies, and that the *first improvement* strategy combined with the VDS branching takes the lead in performance when compared to its counterparts.

Acknowledgements

We express our gratitude to the referee for his valuable comments and suggestions that have significantly improved the quality of the paper. We also gratefully acknowledge the support of the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada, and CENTRUM Católica, Peru, for their computing facilities.

References

- [1] M. Fischetti, A. Lodi, Local branching, *Mathematical Programming* 98 (2003) 23-47.
- [2] B. W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, *The Bell System Technical Journal* 49 (1970) 291-307.
- [3] N. Christofides, S. Eilon, Algorithms for large-scale travelling salesman problems, *Operational Research Quarterly* 23 (1972) 511-518.
- [4] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Operations Research* 21 (1973) 498-516.
- [5] M. M. Amini, M. Racer, A rigorous computational comparison of alternative solution methods for the generalized assignment problem, *Management Science* 40 (1994) 868-890.
- [6] M. Yagiura, T. Yamaguchi, T. Ibaraki, A variable depth search algorithm with branching search for the generalized assignment problem, *Optimization Methods and Software* 10 (1998) 419-441.
- [7] L. J. J. Van der Bruggen, J. K. Lenstra, P. C. Schuur, Variable-depth search for the single-vehicle pickup and delivery problem with time windows, *Transportation Science* 27 (1993) 298-311.
- [8] E. Balas, A. Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling, *Management Science* 44 (1998) 262-275.

- [9] E. G. Negenman, Local search algorithms for the multiprocessor flow shop scheduling problem, *European Journal of Operational Research* 128 (2001) 147-158.
- [10] P. Merz, B. Freisleben, Greedy and local search heuristics for unconstrained binary quadratic programming, *Journal of Heuristics* 8 (2002) 197-213.
- [11] S. Tiourine, C. Hurkens, J. K. Lenstra, Local search algorithms for the radio link frequency assignment problem, *Telecommunications Systems* 13 (2000) 293-314.
- [12] K. Aardal, C. Hurkens, J. K. Lenstra, S. Tiourine, Algorithms for the radio link frequency assignment: the CALMA project, *Operations Research* 50 (2002) 968-980.
- [13] A. Croes, A method for solving traveling salesman problems, *Operations Research* 6 (1958) 791-812.
- [14] S. Lin, Computer solutions of the traveling salesman problem, *Bell System Technical Journal* 44 (1965) 2245-2269.
- [15] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. Bixby, E. Danna, G. Gamrath, A. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. Steffy, K. Wolter, *MIPLIB 2010: Mixed integer programming library version 5*, *Mathematical Programming Computation* 3 (2011) 103-163.

Table 1: Performance comparisons.

Instance	#rows	#cols	G	B	#NZ	VDS _{BI}		LB _{FI}		VDS _{FI}	
						t_{best}	$perf$	t_{best}	$perf$	t_{best}	$perf$
50v-10	233	2014	183	1464	2745	7191	1.4811	3392	0.9105	7191	1.6079
a1c1s1	3312	3649	0	192	10178	3931	1.0076	3931	1.0073	3931	1.0355
atlanta-ip	21732	48739	106	46667	257532	592	0.9897	1401	1.0000	566	0.9897
b2c1s1	3904	3873	0	288	11408	4780	0.9463	6122	0.9505	5729	0.9423
bab1	60680	61153	0	61152	854392	966	1.0009	966	0.9994	966	1.0018
bab4	20069	60837	0	60836	541997	6489	1.0000	6625	0.9999	6489	1.0002
bc	1913	1752	0	483	276842	2	0.8750	2	1.0000	2	0.8750
berlin_5_8_0	1532	1084	0	794	4507	15	0.9688	19	1.0323	19	1.0161
bg512142	1307	793	0	240	3953	6669	1.0075	6851	0.9967	6669	1.0270
blp-ar98	1128	16022	0	15806	200601	6687	1.0021	6814	0.9963	3688	0.9926
blp-ic97	923	9846	0	9753	118149	5018	1.0003	5018	1.0058	5018	1.0080
blp-ic98	717	13641	0	13550	191947	2814	0.9981	5593	1.0028	5593	1.0060
co-100	2187	48418	0	48417	1995817	7038	0.8813	5447	0.8838	6597	0.7597
core4872-1529	4875	24657	0	24645	218762	5977	0.9578	6666	0.9970	6270	0.9512
csched007	351	1759	0	1457	6379	7066	1.0171	7066	1.0171	7066	1.1026
csched008	351	1537	0	1284	5687	782	1.0116	782	1.0116	782	1.0058
d10200	947	2001	1267	733	57637	5882	1.0001	3843	0.9994	5882	1.0019
d20200	1502	4001	819	3181	189389	6974	1.0024	6743	0.9975	6974	1.0086
dano3mip	3202	13874	0	552	79655	1734	1.0237	1557	0.9768	1734	1.0237
dcl1c	1649	10040	0	8380	121158	7166	0.9941	6868	1.0016	7153	0.9780
dcl1l	1653	37298	0	35638	448754	6988	0.0592	6555	0.1191	5368	0.0220
dgo12142	6310	2081	0	640	14795	7199	1.2548	7186	0.1160	5125	0.0431
dolom1	1803	11613	0	9720	190413	6986	0.4132	7119	0.6983	3463	0.6432
ds-big	1042	174998	0	174997	4623442	6312	0.2238	6984	1.4619	4167	0.1830
eilA101.2	100	65833	0	65832	959373	4860	0.7868	4765	1.1661	1591	0.4814
eilD76.2	75	30589	0	30588	381749	6144	1.0056	6144	1.0100	6144	1.0056
ex1010-pi	1468	25201	0	25200	102114	5231	1.0078	5231	1.0547	2610	0.9517
g200x740i	940	1481	0	740	2960	6821	1.0091	6821	1.0035	6821	1.0288
germanrr	10779	10814	5286	5288	175547	6446	0.9956	6564	1.0082	6564	1.0040
gmu-35-50	435	1920	0	1914	8643	6324	1.0001	6386	0.9998	6324	1.0006
gmur-75-50	2565	68866	0	68859	571475	7197	0.8814	7153	1.0807	5826	0.7530
gmur-77-40	2554	24339	0	24332	159902	7198	0.9811	5693	0.7898	2259	0.6312
go19	441	442	0	441	1885	17	0.9767	184	0.9882	8	0.9655
ic97_potential	1046	729	73	450	3138	4772	1.0015	3264	0.9950	3297	0.9952
ivu06-big	1177	2277737	0	2277736	23125770	4804	1.0300	4804	1.0314	4804	1.0314
ivu52	2116	157592	0	157591	2179476	7008	1.0619	7008	1.0472	7008	1.0619
lectsched-1-obj	50108	28719	482	28236	310792	3122	1.0108	3153	0.9677	398	0.8878
leo1	593	6732	0	6730	131218	3343	0.9910	5974	1.0310	5974	1.0216
leo2	593	11101	0	11099	219959	7035	1.0059	7035	1.0664	7035	1.0113
liu	2178	1157	0	1089	10626	7129	1.0242	7129	1.0311	7129	1.1055
loopHA13	23758	19357	0	18150	41809	6074	1.0579	6074	1.1571	6074	1.1183
lotsize	1920	2986	0	1195	6565	4736	1.0008	4736	1.0494	4736	1.1700
markshare_5_0	5	46	0	40	203	320	2.5000	320	2.5000	320	4.0000
mc11	1920	3041	0	1520	6080	4653	0.9910	6855	0.9990	6429	1.0040
mc8	1920	3041	0	1520	6080	5919	0.9688	6678	0.9864	4025	0.9537
methanosarcina	14604	7931	0	7930	43812	2104	0.9126	6463	1.0023	989	0.8385
nine-166-10	17024	1661	0	1660	39442	6491	1.0077	7183	0.9995	4865	0.9130
mkc	3411	5326	0	5323	17038	5961	1.0005	5961	1.0032	5961	1.0071
momentum1	42680	5175	0	2349	103198	6467	1.0581	4408	0.9446	6467	1.0583
momentum3	56822	13533	1	6598	949495	6156	1.0125	7136	0.9804	6917	0.9581
n3700	5150	10001	0	5000	20000	3022	1.0029	3022	1.0104	3022	1.0052
n3705	5150	10001	0	5000	20000	7065	1.0081	7065	1.0273	7065	1.0106
n370a	5150	10001	0	5000	20000	4751	1.0063	4751	1.0254	4751	1.0035
nag	5840	2885	35	1350	26499	4774	1.2885	5594	0.9615	4774	1.4135
neos16	1018	378	41	336	2801	6883	1.0022	6883	1.0022	6883	1.0022
nsr8k	6284	38357	0	32040	371608	6980	0.9704	6156	1.1223	3517	0.9196
nu120-pr3	2210	8602	61	8540	25986	5105	1.0486	5399	0.9779	5105	1.0007
nu60-pr9	2220	7351	42	7308	22176	6024	1.0167	6024	1.0004	6024	1.0165
opm2-z9-s14	100680	4357	0	4356	236180	2619	1.0035	2619	1.0025	2619	1.0025
p100x588b	688	1177	0	588	2352	6452	1.0108	6452	1.0050	6452	1.0315
p6b	5852	463	0	462	11704	614	1.0161	614	1.0161	102	0.9683
p80x400b	480	801	0	400	1600	262	1.0017	262	1.0176	262	1.0195
pb-simp-nonunif	1451912	23849	0	23848	4366648	183	0.6067	4170	0.9900	5526	0.9000
pg	125	2701	0	100	5200	4054	1.0009	4054	1.0039	4054	1.0193
probportfolio	302	321	0	300	6620	4513	1.0507	4513	1.0421	4513	1.1180
queens-30	960	901	0	900	93440	2299	0.9744	2023	1.0000	3790	0.9744
r80x800	880	1601	0	800	3200	3884	1.0030	3884	1.0004	3884	1.0135
ramos3	2187	2188	0	2187	32805	6876	0.7356	6715	0.6461	1715	0.5371
rd-rplusc-21	125899	623	0	457	852384	548	1.0035	548	1.0311	548	1.0186
reblock166	17024	1661	0	1660	39442	4836	0.9067	7089	0.9989	1917	0.5438
reblock354	19906	3541	0	3540	52901	7200	0.7884	6611	0.3892	2944	0.2729
reblock90	6270	901	0	900	15407	382	0.9996	1250	1.0002	1587	1.0000
rmatr200-p10	35055	35255	0	200	105362	181	0.8295	184	1.1407	1489	1.0000
rmatr200-p20	29406	29606	0	200	88415	2399	1.0095	3187	0.9988	1723	0.9870
rmatr200-p5	37617	37817	0	200	113048	3949	0.9712	1316	0.9403	6089	0.9981
rmine10	65274	8440	0	8439	162264	6794	0.7554	7149	0.8429	5420	0.4269
rocll-7-11	37215	16102	0	15851	423661	3122	1.0049	4778	0.9980	3122	1.0049
rocll-9-11	47533	20680	0	20361	544031	7090	1.0018	7090	1.0033	7090	1.0029
rococoB10-011000	1667	4457	136	4320	16517	6467	0.9540	6247	1.0711	1331	0.8550
rococoC11-011100	2367	6492	166	6325	30472	6049	1.0241	6049	1.0896	6049	1.0339
rococoC12-111000	10776	8620	187	8432	48920	1302	1.0570	1302	1.0565	2066	0.9842
rvb-sub	225	33766	0	33763	984143	3432	0.9679	4728	1.0055	1023	0.8531
set5	13304	37266	2302	20702	147037	7189	1.0006	7189	1.0341	7189	1.0094
set3-10	3747	4020	0	1424	13747	7198	0.9184	6949	0.7157	6350	0.6505
set3-15	3747	4020	0	1424	13747	7178	0.8791	7121	0.8009	3781	0.3126
set3-20	3747	4020	0	1424	13747	7199	0.9780	7183	0.7342	5851	0.5602
seymour.disj-10	5108	1210	0	1209	64704	3168	1.0035	3168	1.0035	1992	0.9931
seymour	4944	1373	0	1372	33549	500	1.0047	565	0.9976	6563	0.9976
shipsched	45554	13595	0	10549	121571	7125	1.0853	7125	1.1449	7125	1.3306
siena1	2220	13742	0	11775	258915	6869	1.0751	6869	1.8671	6869	1.1053
sing2	28891	31631	0	23377	149712	7135	0.9760	7092	1.0772	7160	0.9172
sing290	50146	54611	0	39921	274556	6774	1.0005	6774	1.0032	6774	1.0092
sp97ar	1761	14102	0	14101	290968	5280	0.9932	6948	1.0037	6328	0.9920
sp97ic	1033	12498	0	12497	316629	6236	1.0018	4912	0.9952	6236	1.0013
sp98ar	1435	15086	0	15085	426148	2611	0.9490	4055	0.9792	2824	0.9563
stockholm	57346	20645	0	962	171076	3459	0.9859	5792	1.0071	5792	1.0071
sts405	27270	406	0	405	81810	2909	1.0029	2909	1.0029	2909	1.0058

(continued on next page)

Table 1 (continued)

Instance	#rows	#cols	$ \mathcal{G} $	$ \mathcal{B} $	#NZ	VDS _{BI}		LB _{FI}		VDS _{FI}	
						t_{best}	$perf$	t_{best}	$perf$	t_{best}	$perf$
sts729	88452	730	0	729	265356	1082	1.0061	5442	0.9939	7195	0.9894
swath	884	6806	0	6724	34965	2328	0.9955	2978	1.0111	2978	1.0221
t1717	551	73886	0	73885	325689	5659	0.9154	7109	0.9601	6618	0.9098
t1722	338	36631	0	36630	133096	4194	0.9649	6111	0.9397	2539	0.9589
tanglegram3	192420	97289	0	97288	577260	7120	1.0077	7120	1.0090	7120	1.0081
toll-like	4408	2884	0	2883	13224	1870	0.7020	7159	1.0052	1711	0.7040
tw-myciel4	8146	761	1	759	27961	1354	1.2727	1354	1.0909	1354	1.2727
uc-case11	51438	34135	302	3898	202042	7097	0.9996	4117	1.0011	4117	1.0010
uc-case3	52003	37750	0	11256	273618	7145	0.3882	6750	0.2172	6458	0.2766
uct-subprob	1973	2257	0	379	10147	6676	0.9906	2981	0.9876	5740	0.9812
umts	4465	2948	72	2802	23016	4052	1.0006	4946	0.9919	4052	1.0002
usAbbrv.8.25_70	3291	2313	0	1681	9628	7055	0.9917	6674	1.0083	6674	1.0165
vpphard2	198450	200000	0	199999	648340	7189	0.9277	7079	1.0010	7201	0.9384
wnq-n100-mw99-14	656900	10001	0	10000	1333400	787	1.0035	5045	0.9721	2309	0.9895