Yujor xx (yyyy), zzz–zzz
DOI 10.2298/YJOR140417027L

Variable and Single Neighbourhood Diving for MIP Feasibility

Jasmina Lazić
Brunel University, West London UB8 3PH, UK
Mathematical Institute, Serbian Academy of Sciences and Arts,
Kneza Mihaila 36, 11000 Belgrade, Serbia
Jasmina.Lazic@brunel.ac.uk

Raca Todosijević
LAMIH - Université de Valenciennes,
ISTV 2 Le Mont Houy, 59313 Valenciennes Cedex 9, France
Mathematical Institute, Serbian Academy of Sciences and Arts,
Kneza Mihaila 36, 11000 Belgrade, Serbia
raca.todosijevic@gmail.com

Saïd Hanafi
LAMIH - Université de Valenciennes,
ISTV 2 Le Mont Houy, 59313 Valenciennes Cedex 9, France
CNRS, FRE 3304, 59313 Valenciennes Cedex 9, France
Université Lille Nord de France, 59000 Lille, France
said.hanafi@univ-valenciennes.fr

Nenad Mladenović
Mathematical Institute, Serbian Academy of Sciences and Arts,
Kneza Mihaila 36, 11000 Belgrade, Serbia
nenad@mi.sanu.ac.rs

Received: 17. April 2014. / Accepted: 24. September 2014.

Abstract. In this paper, we propose two new diving heuristics for finding a feasible solution for a mixed integer programming problem, called *variable neighbourhood (VN) diving* and *single neighbourhood (SN) diving*, respectively. They perform systematic hard variable fixing (i.e. diving) by exploiting the information obtained from a series of LP relaxations in order to generate a sequence of subproblems. Pseudo cuts are added during the search process to avoid revisiting the same search space areas. VN diving is based on the variable neighbourhood decomposition search framework. Conversely, SN diving explores only a single neighbourhood in each iteration: if a feasible solution is not

found, then the next reference solution is chosen using the feasibility pump principle and the search history. Moreover, we prove that the two proposed algorithms converge in a finite number of iterations (i.e. either return a feasible solution of the input problem, or prove its infeasibility). We show that our proposed algorithms significantly outperform the CPLEX 12.4 MIP solver and the recent variants of feasibility pump regarding the solution quality.

Keywords: Mixed Integer Programming, Constructive Heuristics, Feasibility Pump, CPLEX

MSC: 90B06, 90C05, 90C08

1. Introduction

The mixed integer programming (MIP) problem can be formulated as follows:

$$(P) \qquad \min\{c^T x \mid x \in X\}, \tag{1}$$

where $X = \{x \in \mathbb{R}^n \mid Ax \leq b, x_j \in \{0,1\} \text{ for } j \in \mathcal{B}, x_j \in \mathbb{Z}^+ \text{ for } j \in \mathcal{G}, l_j \leq x_j \leq u_j \text{ for } j \in \mathcal{C} \cup \mathcal{G}\}$ ($\mathcal{B}, \mathcal{G}, \mathcal{C}$ respectively constitute the index sets for the binary (0-1), integer (non-binary) and continuous variables) is the feasible set, $c^T x$ is the objective function, and $x \in X$ are the feasible solutions. In the special case when $\mathcal{G} = \emptyset$, the resulting MIP problem is called the 0-1 MIP problem (0-1 MIP). The LP-relaxation of problem P, denoted as LP(P), is obtained from the original formulation by relaxing the integer requirements on x:

$$\text{LP}(P) \qquad \min\{c^T x \mid x \in \overline{X}\}, \tag{2}$$

where $\overline{X} = \{x \in \mathbb{R}^n \mid Ax \leq b, l_j \leq x_j \leq u_j \text{ for } j \in \mathcal{G} \cup \mathcal{C}, x_j \in [0,1] \text{ for } j \in \mathcal{B}\}$.

Many real-world problems can be modelled as MIP problems [5, 6]. However, a number of special cases of MIP problem are proven to be NP-hard [11] and cannot be solved to optimality within acceptable time/space with existing exact methods. This is why various heuristic methods have been designed in attempt to find good near-optimal solutions of hard MIP problems. Most of them start from a given feasible solution and try to improve it. Still, finding a feasible solution of 0-1 MIP is proven to be NP-complete [28] and for a number of instances finding a feasible solution remains hard in practice. This calls for the development of efficient constructive heuristics which can attain feasible solutions in short time. Over the last decade, a number of heuristics that address the problem of MIP feasibility have been proposed. The feasibility Pump (FP) heuristic was proposed for the special case of pure 0-1 MIP problem in [8]. It generates a sequence of linear programming problems, whose objective function represents the infeasibility measure of the initial MIP problem. The solution of each subproblem is used to define the objective function of the next subproblem, so that the infeasibility measure is reduced in each iteration [8]. This approach was extended in [3] for the case of general MIP problems. The FP heuristic is quite efficient in terms of computational time, but usually provides poor-quality solutions. In [1], objective

FP was proposed with the aim to improve the quality of the feasible solutions obtained. However, the computational time was increased on average, compared to the basic version of FP. Another approach, proposed in [10], applies the Local Branching (LB) heuristic [9] to near-feasible solutions obtained from FP in order to locate feasible solutions. LB is applied to a modified problem in which the original objective function is replaced by an infeasibility measure taking into account a weighted combination of the degree of violation of the single linear constraints. This heuristic provides feasible solutions very fast, but those solutions are again usually of poor quality since the original objective function is completely discarded.

The concept of variable fixing in order to find solutions to MIP problems was conceived in the late 1970s and early 1980s, when the first methods of this type were proposed [2, 26]. Subproblems are iteratively generated by fixing a certain number of variables in the original problem according to the solution of the linear programming relaxation of the original problem. This approach is also referred to as a *core approach*, since the subproblems so obtained are sometimes called *core problems* [2, 25]. The terms *hard variable fixing* or *diving*, which are used throughout this paper, are also present in the literature (see, for example, [7]). The critical issue in this type of methods is the way in which the variables to be fixed are chosen. Depending on the selection strategy and the way of manipulating the obtained subproblems, different MIP solution methods are obtained. The basic strategy was initially proposed in [2], for solving the multidimensional knapsack problem. A number of its successful extensions were proposed over the years. For example, a greedy strategy for determining the core is developed in [23], whereas in [25] the core is defined according to a chosen efficiency function. Another iterative scheme, again for the 0-1 multidimensional knapsack problem, was developed in [27]. This scheme, which is based on a dynamic fixation of the variables, uses the search history to build up feasible solutions and to select variables for a permanent/temporary fixation. Variable neighbourhood search was combined with a very large scale neighbourhood search approach to select variables for fixing (*binding sets*) for the general assignment problem [20, 22]. This approach was further extended for 0-1 mixed integer programming in general [21].

With the expansion of general-purpose MIP solvers over the last decade, different hybridisations of MIP heuristics with commercial solvers are becoming increasingly popular. A number of efficient heuristics that perform some kind of variable fixing at each node of the Branch and Bound tree in the CPLEX MIP solver have been developed. Relaxation induced neighbourhood search (RINS) [7] fixes the values of the variables, which are the same in the current continuous (i.e. LP) relaxation and in the incumbent integral solution. Besides considering the values of variables in the current LP relaxation solution, Distance induced neighbourhood search [12] performs a more sophisticated fixation taking into account the solution of the LP relaxation in the root of the Branch-and-Bound tree and the counts of occurrences of different values. Relaxation enforced neighbourhood search [4] is an extension of RINS, which additionally performs a large-scale neighbourhood search over the set of general integer variables by an intelligent rebounding according to the current LP relaxation solution. In [19], variable fixation is performed

in a variable neighbourhood decomposition search manner [15].

In this paper we propose two new diving heuristics for MIP feasibility, which exploit the information obtained from a series of LP relaxations. Since the variables to be fixed depend on the LP relaxation values, this approach may also be called *relaxation guided diving*. Relaxation guided variable neighbourhood search was proposed in [24], but for defining the order of neighbourhoods within VNS (where neighbourhoods are defined by soft variable fixing) rather than selecting the variables to be hard-fixed. The first heuristic, called *variable neighbourhood diving* is based on the variable neighbourhood decomposition search principle [15]. A similar approach was proposed in [19] for optimising 0-1 MIP problems starting from a given initial MIP feasible solution. In this paper we propose a modification of the algorithm from [19] for constructing feasible solutions of 0-1 MIP problems. We exploit the fact that the CPLEX MIP solver can be used not only for finding near-optimal solutions but also as a black-box for finding a first feasible solution for a given 0-1 MIP problem. We also extend this approach for general MIP problems, so that fixation is performed on general integer variables as well. The second heuristic, called *single neighbourhood diving* explores only a single neighbourhood in each iteration. However, the size of the neighbourhood is updated dynamically according to the solution status of the subproblem in a previous iteration. The incumbent solution is updated in a feasibility pump manner, whereas revisiting the same point in the search process is prohibited by keeping the list of all visited reference solutions. This list is implemented as a set of constraints in a new (dummy) MIP problem. We show that our proposed algorithms significantly outperform the CPLEX 12.4 MIP solver and the recent variants of the feasibility pump heuristic, both regarding the solution quality and the computational time.

This paper is organised as follows. In Section 2, we present the necessary notation and a brief overview of the existing approaches related to our work. A detailed description of the two new diving heuristics for MIP feasibility is provided in Section 3. In Section 4, we analyse the performance of the proposed methods as compared to the commercial IBM ILOG CPLEX 12.4 MIP solver and the basic and objective variant of the FP heuristic [1, 8]. At last, in Section 5, we give some final remarks and conclusions.

2. Preliminaries

2.1. Notation

Given an arbitrary integer solution x^0 of problem (1) and an arbitrary subset $J \subseteq \mathcal{B} \cup \mathcal{G}$ of integer variables, the problem *reduced* from the original problem P and associated with x^0 and J can be defined as:

$$P(x^0, J) \qquad \min\{c^T x \mid x \in X, x_j = x_j^0 \text{ for } j \in J\} \tag{3}$$

If C is a set of constraints, we will denote with $(P \mid C)$ the problem obtained by adding all constraints in C to the problem P.

Let x and y be two arbitrary integer solutions of the problem P. The distance between x and y is then defined as

$$\Delta(x,y) = \sum_{j \in \mathcal{B} \cup \mathcal{G}} |x_j - y_j| \tag{4}$$

If $J \subseteq \mathcal{B} \cup \mathcal{G}$, the partial distance between x and y, relative to J, is defined as $\Delta(J,x,y) = \sum_{j \in J} \mid x_j - y_j \mid$ (obviously, $\Delta(\mathcal{B} \cup \mathcal{G}, x, y) = \Delta(x,y)$). The linearisation of the distance function $\Delta(x,y)$, as defined in (4), requires the introduction of additional variables. More precisely, for any integer feasible vector y, function $\Delta(x,y)$ can be linearised as follows [9]:

$$\Delta(x,y) = \sum_{j \in \mathcal{B} \cup \mathcal{G}: y_j = l_j} (x_j - l_j) + \sum_{j \in \mathcal{B} \cup \mathcal{G}: y_j = u_j} (u_j - x_j) + \sum_{j \in \mathcal{G}: l_j < y_j < u_j} d_j, \tag{5}$$

where $l_j = 0$ and $u_j = 1$ for $j \in \mathcal{B}$ and new variables $d_j = |x_j - y_j|$ need to satisfy the following constraints :

$$d_j \geq x_j - y_j \;\text{ and }\; d_j \geq y_j - x_j \;\text{ for all } j \in \{i \in \mathcal{G} \mid l_i < y_i < u_i\}. \tag{6}$$

In the special case of 0-1 MIP problems, the distance function between any two binary vectors x and y can be expressed as:

$$\delta(x,y) = \sum_{j \in \mathcal{B}} x_j(1 - y_j) + y_j(1 - x_j). \tag{7}$$

Furthermore, if x is a given binary vector, then formula (7) can be used to compute the distance from x to any vector $\overline{x} \in \mathbb{R}^n$:

$$\delta(x,\overline{x}) = \sum_{j \in \mathcal{B}} x_j(1 - \overline{x}_j) + \overline{x}_j(1 - x_j).$$

As in the case of general MIP problems, the partial distance between x and $\overline{x}$, relative to $J \subseteq \mathcal{B}$, is defined as $\delta(J,x,\overline{x}) = \sum_{j \in J} x_j(1 - \overline{x}_j) + \overline{x}_j(1 - x_j)$. Note that the distance function δ, as defined in (7), can also be used for general MIP problems, by taking into account that $\delta(x,y) = \Delta(\mathcal{B},x,y)$ for any two solution vectors x and y of a general MIP problem (1).

The LP-relaxation of the modified problem, obtained from a MIP problem P, as defined in (1), by replacing the original objective function $c^T x$ with $\delta(\tilde{x},x)$, for a given integer vector $\tilde{x} \in \{0,1\}^{|\mathcal{B}|} \times \mathbb{Z}_+^{|\mathcal{G}|} \times \mathbb{R}_+^{|\mathcal{C}|}$, can be expressed as:

$$\mathrm{LP}(P,\tilde{x}) \qquad \min\{\delta(\tilde{x},x) \mid x \in \overline{X}\} \tag{8}$$

Similarly, the notation $\mathrm{MIP}(P,\widetilde{x})$ will be used to denote a modified problem, obtained from P by replacing the original objective function with $\delta(x,\widetilde{x})$:

$$\mathrm{MIP}(P,\tilde{x}) \qquad \min\{\delta(\tilde{x},x) \mid x \in X\}. \tag{9}$$

We will also define the *rounding* $[x]$ of any vector x, as vector $[x] = ([x]_j)$, with:

$$[x]_j = \begin{cases} \lfloor x_j + 0.5 \rfloor, & j \in \mathcal{B} \cup \mathcal{G} \\ x_j, & j \in \mathcal{C}. \end{cases} \tag{10}$$

The neighbourhood structures $\{\mathcal{N}_k \mid 1 \leq k_{min} \leq k \leq k_{max} \leq |\mathcal{B}| + |\mathcal{G}|\}$ can be defined knowing the distance $\delta(x, y)$ between any two solutions $x, y \in X$. The set of all solutions in the kth neighbourhood of $x \in X$ is defined as

$$\mathcal{N}_k(x) = \{y \in X \mid \delta(x, y) \; = \; k\}. \tag{11}$$

2.2. Related Work

We here present a brief survey of the methods closely related to the research reported in this paper. We provide short descriptions of the feasibility pump heuristic [8, 1, 3] and variable neighbourhood decomposition search for 0-1 MIP problems [19].

Feasibility Pump. Feasibility Pump (FP), introduced in [8], is a fast and simple heuristic for finding a feasible solution to 0-1 MIP. Starting from an optimal solution of the LP-relaxation, the FP heuristic generates two sequences of solutions $\overline{x}$ and $\tilde{x}$, which satisfy LP-feasibility and integrality feasibility, respectively. The two sequences of solutions are obtained as follows: at each iteration, a new binary solution $\tilde{x}$ is obtained from the fractional $\overline{x}$ by simply rounding its integer-constrained components to the nearest integer, i.e. $\tilde{x} = [\overline{x}]$, while a new fractional solution $\overline{x}$ is defined as an optimal solution of $\mathrm{LP}(P, \tilde{x})$. To avoid cycling, some random perturbations of the current solution $\tilde{x}$ are performed. In the original implementation, the neighbourhood $\mathcal{N}_k(\tilde{x})$, $k \in [T/2, 3T/2]$ of the current solution $\tilde{x}$ is chosen at random (where T is an input parameter), and $\tilde{x}$ is replaced with $x' \in \mathcal{N}_k(\tilde{x})$, such that $\delta(x', \overline{x}) = \max_{y \in \mathcal{N}_k(\tilde{x})} \delta(y, \overline{x})$. The whole process is iterated until a feasible solution is detected, or some of stopping criteria are fulfilled. The stopping criteria usually contain a running time limit and/or the total number of iterations. The pseudo-code of the basic FP is given in Figure 1.

The basic feasibility pump employs the distance function (7) which is defined only on the set of binary variables. The general feasibility pump, proposed in [3], employs the distance function (4) in which the general integer variables also contribute to the distance. According to the computational results reported in [3, 8], the feasibility pump is usually quite effective with respect to the computational time needed to provide the first feasible solution. However, the solution provided is often of a poor-quality in terms of the objective value. The reason is that the original objective function is completely discarded after solving the LP relaxation of the original problem in order to construct the starting point for the search. In an attempt to provide good-quality initial solutions, a modification of the basic FP scheme, the so called *objective feasibility pump* was proposed in [1]. The idea of objective FP is to include the original objective function as a part of the objective function of the problem considered at a certain pumping cycle of FP. At each

```
Procedure FP(P)
 1   Set x̄ = LPSolve(P); Set proceed = true;
 2   while (proceed) do
 3      if (x̄ is integer) then return x̄;
 4      Set x̃ = [x̄];
 5      if (cycle detected) then
 6         Select k ∈ {1, 2, ..., |B| + |G|} at random;
 7         Select x' ∈ N_k(x̃);
 8         Set x̃ = x';
 9      endif
10      x̄ = LPSolve(LP(P, x̃));
11      Update proceed;
12   endwhile
```

Figure 1: The basic feasibility pump.

pumping cycle, the actual objective function is computed as a linear combination of the feasibility measure and the original objective function:

$$\Delta_\alpha(x, \widetilde{x}) = (1-\alpha)\Delta(x, \widetilde{x}) + \alpha \frac{\sqrt{|\mathcal{B} \cup \mathcal{G}|}}{||c||} c^{\mathrm{T}} x, \ \alpha \in [0, 1], \tag{12}$$

where $||\cdot||$ denotes the Euclidean norm. Results reported in [1] indicate that this approach usually yields considerably higher-quality solutions than the basic FP. However, it generally requires much longer computational time.

Variable Neighbourhood Pump (VNP). The feasibility pump approach from [8] and variable neighbourhood branching (VNB) from [16] were successfully combined to provide a method for finding good quality solutions within a relatively short computational time (see [13, 17]).

The VNP heuristic starts from an optimal solution $\overline{x}$ of the LP-relaxation of the initial 0-1 MIP problem. It first performs one iteration of the FP pumping cycle to the rounded vector $[\overline{x}]$ in order to obtain a near-feasible vector $\tilde{x}$. A deterministic search procedure VNB$(P, \tilde{x}, k_{min}, k_{step}, k_{max})$ based on variable neighbourhood branching [16], and adjusted for 0-1 MIP feasibility as in [13, 17], is then applied to $\tilde{x}$ in an attempt to locate a feasible solution of the original problem. Procedure VNB applies variable neighbourhood descent [14] to an initial reference solution $\tilde{x}$, starting from the minimum neighbourhood size k_{min}, with the neighbourhood increase step k_{step}, until the maximum neighbourhood size k_{max} is reached. The variable neighbourhood pump algorithm is based on the observation that $\tilde{x}$ is usually near-feasible, and it is very likely that feasible solution vectors can be found in small neighbourhoods of $\tilde{x}$. In addition, if VNB fails to detect a feasible solution due to the time or neighbourhood size limitations, a pseudo-cut is added to the current subproblem in order to change the linear relaxation solution, and the

process is iterated. If no feasible solution has been found, the algorithm reports failure and returns the last integer (infeasible) solution. The VNP pseudo-code for 0-1 MIP feasibility is given in Figure 2.

```
Procedure VNP(P)
 1   Set proceed1 = true;
 2   while (proceed1) do
 3      Set x̄ = LPSolve(P); Set x̃ = [x̄]; Set proceed2 = true;
 4      while (proceed2) do
 5         if (x̄ is integer) then return x̄;
 6          x̄ = LPSolve(LP(P, x̃));
 7          if (x̃ ≠ [x̄]) then x̃ = [x̄];
 8         else Set proceed2 = false;
 9         endif
10      endwhile
11      k_min = ⌊δ(x̃, x̄)⌋; k_max = ⌊(|B| − k_min)/2⌋; k_step = (k_max − k_min)/5;
12      x' = VNB(P, x̃, k_min, k_step, k_max);
13      if (x'=x̃) then //VNB failed to find the feasible solution.
14          P = (P | δ(x, x̄) ≥ k_min); Update proceed1;
15      else return  x';
16      endif
17   endwhile
18   Output message: "No feasible solution found."; return x̃;
```

Figure 2: The variable neighbourhood pump heuristic pseudo-code.

Variable Neighbourhood Decomposition Search for 0-1 MIP problems. Variable neighbourhood decomposition search (VNDS) is a two-level variable neighbourhood search (VNS) scheme for solving optimisation problems, based upon the decomposition of the problem [15]. Recently, a new variant of VNDS for solving 0-1 MIP problems, called VNDS-MIP, was proposed in [19]. This method combines a linear programming (LP) solver, a MIP solver, and variable neighbourhood branching [16] in order to efficiently solve a given 0-1 MIP problem. At the beginning of the algorithm, the LP-relaxation LP(P) of the original problem P is solved in order to obtain an optimal solution $\overline{x}$, and an initial integer feasible solution x is generated. Then, the search for an improvement of the incumbent objective value is performed solving a series of subproblems $P(x, J_k)$, where subset $J_k \subseteq \mathcal{B} \cup \mathcal{G}$ corresponds to the indices of variables with k smallest $|\overline{x}_j - x^*_j|$ values, and x^* is the current incumbent solution. If the improvement occurs, VNB is performed over the whole search space and the process is iterated. The pseudo-code of the VNDS-MIP method can be found in [19].

3. New Diving Heuristics for MIP Feasibility

The new diving heuristics presented in this section are based on the systematic hard variable fixing (diving) process, according to the information obtained from the linear relaxation solution of the problem. They rely on the observation that a general-purpose MIP solver can be used not only for finding (near) optimal solutions of a given input problem, but also for finding the initial feasible solution. For the sake of simplicity, in Subsections 3.1 and 3.2 we will first present both algorithms for the special case of 0-1 MIP problems. Then, in Subsection 3.3, we explain how the presented algorithms can be adapted for solving general MIP problems.

3.1. Variable Neighbourhood Diving

The variable neighbourhood (VN) diving algorithm begins by obtaining the LP-relaxation solution $\overline{x}$ of the original problem P and generating an initial integer (not necessarily feasible) solution $\tilde{x} = [\overline{x}]$ by rounding the LP-solution $\overline{x}$. If the optimal solution $\overline{x}$ is integer feasible for P, we stop and return $\overline{x}$. At each iteration of the VN diving procedure, we compute the distances $\delta_j = \mid \tilde{x}_j - \overline{x}_j \mid$ from the current integer solution values $(\tilde{x}_j)_{j \in \mathcal{B}}$ to the corresponding LP-relaxation solution values $(\overline{x}_j)_{j \in \mathcal{B}}$ and index the variables $\tilde{x}_j, j \in \mathcal{B}$ so that $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_{|\mathcal{B}|}$). Then, we successively solve the subproblems $P(\tilde{x}, \{1, \ldots, k\})$ obtained from the original problem P, where the first k variables are fixed to their values in the current incumbent solution $\tilde{x}$. If a feasible solution is found by solving $P(\tilde{x}, \{1, \ldots, k\})$, it is returned as a feasible solution of the original problem P. Otherwise, a pseudo-cut $\delta(\{1, \ldots, k\}, \tilde{x}, x) \geq 1$ is added in order to avoid exploring the search space of $P(\tilde{x}, \{1, \ldots, k\})$ again, and the next subproblem is examined. If no feasible solution is detected after solving all subproblems $P(\tilde{x}, \{1, \ldots, k\})$, $k_{min} \leq k \leq k_{max}$, $k_{min} = k_{step}$, $k_{max} = |\mathcal{B}| - k_{step}$, the linear relaxation of the current problem P, which includes all the pseudo-cuts added during the search process, is solved and the process is iterated. If no feasible solution has been found due to the fulfilment of the stopping criteria, the algorithm reports failure and returns the last (infeasible) integer solution.

The pseudo-code of the proposed VN diving heuristic is given in Figure 3. The input parameters for the VN diving algorithm are the input MIP problem P and the parameter d, which controls the change of neighbourhood size during the search process. In all pseudo-codes, a statement of the form $y =$ `FindFirstFeasible`(P, t) denotes a call to a generic MIP solver, an attempt to find a first feasible solution of an input problem P within a given time limit t. If a feasible solution is found, it is assigned to the variable y, otherwise y retains its previous value.

Since the VN diving procedure examines only a finite number of subproblems, it is easy to prove the following proposition.

`VN-Diving`(P, d)
1 Set *proceed*1 = `true`, *proceed*2 = `true`; Set *timeLimit* for subproblems;
3 **while** (*proceed*1) **do**
4 $\overline{x} =$ `LPSolve`(P); $\tilde{x} = [\overline{x}]$;
5 **if** $(\overline{x} = \tilde{x})$ **then return** $\tilde{x}$;
6 $\delta_j =\mid \tilde{x}_j - \overline{x}_j \mid$; index x_j so that $\delta_j \leq \delta_{j+1}$, $j = 1, \ldots, |\mathcal{B}| - 1$;
7 Set $n_d =\mid \{j \in \mathcal{B} \mid \delta_j \neq 0\} \mid$, $k_{step} = [n_d/d]$, $k = |\mathcal{B}| - k_{step}$;
8 **while** (*proceed*2 **and** $k \geq 0$) **do**
9 $J_k = \{1, \ldots, k\}$; $x' =$ `FindFirstFeasible`$(P(\tilde{x}, J_k), timeLimit)$;
10 **if** $(P(\tilde{x}, J_k)$ *is proven infeasible*$)$ **then**
11 $P = (P \mid \delta(J_k, \tilde{x}, x) \geq 1)$;
12 **if** (x' is feasible) **then return** x';
13 **if** $(k - k_{step} > |\mathcal{B}| - n_d)$ **then** $k_{step} = \max\{[k/2], 1\}$;
14 Set $k = k - k_{step}$;
15 Update *proceed*2;
16 **endwhile**
17 Update *proceed*1;
18 **endwhile**
19 Output message: "No feasible solution found"; **return** $\tilde{x}$;

Figure 3: Variable neighbourhood diving for 0-1 MIP feasibility.

Proposition 1. *If the* `timeLimit` *parameter is set to infinity, the variable neighbourhood diving algorithm finishes in a finite number of iterations and either returns a feasible solution of the input problem, or proves the infeasibility of the input problem.*

Note however that, in the worst case, the last subproblem examined by VN diving is the original input problem. Therefore, the result of Proposition 1 does not have any theoretical significance.

3.2. Single Neighbourhood Diving

In the case of variable neighbourhood diving, a set of subproblems $P(\tilde{x}, J_k)$, for different values of k, is examined in each iteration until a feasible solution is found. In the single neighbourhood diving procedure, we only examine one subproblem $P(\tilde{x}, J_k)$ in each iteration (a single neighbourhood, see Figure 4). However, because only a single neighbourhood is examined, additional diversification mechanisms are required. This diversification is provided through keeping the list of constraints which ensures that the same reference integer solution $\widetilde{x}$ cannot occur more than once (i.e. in more than one iteration) in the solution process. An additional MIP problem Q is introduced to store these constraints. In the beginning of the algorithm, Q is initialised as an empty problem (see line 4 in Figure 4). Then, in each iteration, if the current reference solution $\widetilde{x}$ is not feasible (see line 8 in Figure 4), constraint $\delta(\widetilde{x}, x) \geq \lceil \delta(\widetilde{x}, \overline{x}) \rceil$ is added to Q (line 9). This guarantees that future reference solutions can not be the same as the current one, since the next reference solution is obtained by solving the problem MIP$(Q, [\overline{x}])$ (see line 17), which

contains all constraints from Q, (see definition (9)). The variables to be fixed in the current subproblem are chosen among those which have the same value as in the linear relaxation solution of the modified problem $\mathrm{LP}(P, \widetilde{x})$, where $\widetilde{x}$ is the current reference integer solution (see lines 7 and 10). The number of variables to be fixed is controlled by the parameter α (line 10). After initialisation (line 5), the value of α is updated in each iteration, depending on the solution status returned from the MIP solver. If the current subproblem is proven infeasible, the value of α is increased in order to reduce the number of fixed variables in the next iteration (see line 16), and thus provide better diversification. Otherwise, if the time limit allowed for subproblem is exceeded without reaching a feasible solution or proving the subproblem infeasibility, the value of α is decreased. Decreasing the value of α, increases the number of fixed variables in the next iteration (see line 17), and thus reduces the size of the next subproblem. In the feasibility pump, the next reference integer solution is obtained by simply rounding the linear relaxation solution $\overline{x}$ of the modified problem $\mathrm{LP}(P, \widetilde{x})$. However, if $[\overline{x}]$ is equal to some of the previous reference solutions, the solution process is caught in a cycle. In order to avoid this type of cycling, we determine the next reference solution as the one which is at the minimum distance from $[\overline{x}]$ (with respect to binary variables) and satisfies all constraints from the current subproblem Q (see line 19). This way we guarantee the convergence of the variable neighbourhood diving algorithm, as stated in the following proposition.

Proposition 2. *If the* `timeLimit` *parameter is set to infinity, the single neighbourhood diving algorithm finishes in a finite number of iterations and either returns a feasible solution of the input problem, or proves the infeasibility of the input problem.*

Proof. Let $\widetilde{x}^i$ be the reference solution at the beginning of the ith iteration, obtained by solving the MIP problem $\mathrm{MIP}(Q_i, [\overline{x}])$ and let $j \geq i+1$. The problem Q_j contains all constraints from Q_{i+1}. If the algorithm has reached the jth iteration, it means that in the ith iteration feasible solution was not found and cut $\delta(\widetilde{x}^i, x) \geq \lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$ (line 9 in Figure 4) was added to Q_{i+1}. Hence, the problem $\mathrm{MIP}(Q_j, [\overline{x}])$ contains $\delta(\widetilde{x}^i, x) \geq \lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$. Furthermore, because $\lceil \delta(\widetilde{x}^i, \overline{x}) \rceil > 0$ (otherwise, $\widetilde{x}^i$ would be feasible and the algorithm would stop in the ith iteration), this implies that $\widetilde{x}^i(\mathcal{B}) \neq \widetilde{x}^j(\mathcal{B})$. Since this reasoning holds for any two iterations $j > i \geq 0$, the total number of iterations of the single neighbourhood diving algorithm is limited by the number of possible sub vectors $\widetilde{x}^i(\mathcal{B})$, which is $2^{|\mathcal{B}|}$. Therefore, the single neighbourhood diving algorithm finishes in a finite number of iterations.

The single neighbourhood diving algorithm can only return a solution vector as a result if either $\lceil \delta(\widetilde{x}^i, \overline{x}) \rceil = 0$, therefore $\widetilde{x}^i$ being feasible for P, or if a feasible solution of the reduced problem $P(\widetilde{x}^i, J_k)$ is found. Since a feasible solution of $P(\widetilde{x}^i, J_k)$ is also feasible for P, this means that any solution vector returned by single neighbourhood diving algorithm must be feasible for P.

Finally, we will prove that any feasible solution of P has to be feasible for Q_i, for any iteration $i \geq 0$. Moreover, we will prove that any feasible solution of P has

```
Procedure SN-Diving(P)
 1  Set x̄ = LPSolve(P);
 2  Set i = 0; Set x̃^0 = [x̄];
 3  if (x̄ = x̃^0) then return x̃^0;
 4  Set Q_0 = ∅;
 5  Set proceed = true; Set timeLimit for subproblems; Set value of α;
 6  while (proceed) do
 7      x̄ = LPSolve(LP(P, x̃^i));
 8      if (⌈δ(x̃^i, x̄)⌉ = 0) then return x̃^i;
 9      Q_{i+1} = (Q_i | δ(x̃^i, x) ≥ ⌈δ(x̃^i, x̄)⌉);
10      k =| {j ∈ B : x̃^i_j = x̄_j} | /α; J_k = {1, ..., k};
11      x' = FindFirstFeasible(P(x̃^i, J_k), timeLimit);
12      if (feasible solution found) then
13          return x';
14      if (P(x̃^i, J_k) is proven infeasible) then
15          Q_{i+1} = (Q_{i+1} | δ(J_k, x̃^i, x) ≥ 1); P = (P | δ(J_k, x̃^i, x) ≥ 1);
16          α = 3α/2;
17      else if (time limit for subproblem exceeded)
18          α = max(1, α/2);
19      x̃^{i+1} = FindFirstFeasible(MIP(Q_{i+1}, [x̄]), timeLimit);
20      if (MIP(Q_{i+1}, [x̄]) is proven infeasible) then
21          Output message: "Problem P is proven infeasible"; return;
22      i = i + 1;
23  endwhile
```

Figure 4: Single neighbourhood diving for 0-1 MIP feasibility.

to satisfy all constraints in Q_i, for any iteration $i \geq 0$. Since Q_0 does not contain any constraints, this statement is obviously true for $i = 0$. Let us assume that the statement is true for some $i \geq 0$, i.e. that for some $i \geq 0$ every feasible solution of P satisfies all constraints in Q_i. The problem Q_{i+1} is obtained from Q_i by adding constraints $\delta(\widetilde{x}^i, x) \geq \lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$ and $\delta(J_k, \widetilde{x}^i, x) \geq 1$. According to the definition of $\lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$, there cannot be any feasible solution of P satisfying the constraint $\delta(\widetilde{x}^i, x) < \lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$. In other words, all feasible solutions of P must satisfy the constraint $\delta(\widetilde{x}^i, x) \geq \lceil \delta(\widetilde{x}^i, \overline{x}) \rceil$. Furthermore, if the constraint $\delta(J_k, \widetilde{x}^i, x) \geq 1$) is added to Q_{i+1}, this means that the problem $P(\widetilde{x}^i, J_k) = (P \mid \delta(J_k, \widetilde{x}^i, x) = 0)$ is proven infeasible, and therefore no feasible solution of P can satisfy the constraint $\delta(J_k, \widetilde{x}^i, x) = 0$. Therefore, any feasible solution of P satisfies the constraints added to Q_i in order to obtain Q_{i+1} and hence any feasible solution of P satisfies all constraints in Q_{i+1}. This proves that any feasible solution of P satisfies all constraints in Q_i, for any $i \geq 0$. In other words, any feasible solution of P is feasible for Q_i, for any $i \geq 0$. Since $\text{MIP}(Q_{i+1}, [\overline{x}])$ has the same set of constraints as Q_i, this means that any feasible solution of P is feasible for $\text{MIP}(Q_i, [\overline{x}])$. As a consequence, if $\text{MIP}(Q_i, [\overline{x}])$ is proven infeasible for some $i \geq 0$, this implies that

the original problem P is infeasible. ■

3.3. Extension to a General MIP Case

Obviously, fixing a certain number of variables can be performed for general MIP problems, as well as for 0-1 MIP problems. We here explain how the previously presented algorithms can be adapted and employed for solving the general MIP problems. In the case of VN diving, we compute the distances $\Delta_j = \mid \tilde{x}_j - \overline{x}_j \mid$, $j \in \mathcal{B} \cup \mathcal{G}$, for all integer variables (not just the binaries). Then, we successively solve subproblems $P(\tilde{x}, J_k)$, $J_k = \{1, \ldots, k\}$, $k = \mid \{j \in \mathcal{B} \cup \mathcal{G} : \widetilde{x}_j = \overline{x}_j\} \mid$, where $\widetilde{x}$ is the current reference integer solution and $\overline{x}$ is the solution of the LP relaxation of the original problem LP(P). If a feasible solution is found by solving $P(\tilde{x}, J_k)$, for some k, $0 \leq k \leq |\mathcal{B} \cup \mathcal{G}|$, it is returned as a feasible solution of the original problem P. In the VN diving variant for 0-1 MIP problems, a pseudo-cut is added to P if a subproblem $P(\tilde{x}, J_k)$ is proven infeasible. In the case of general MIP problems however, generating an appropriate pseudo-cut would require operating with extended problems, which contain significantly more variables and constraints than the original problem P. More precisely, the input problem would have to contain additional variables d_j, $j \in \mathcal{G}$, and additional constraints (see definition (5)):

$$u_j - d_j \leq x_j \leq d_j + l_j \quad \text{for all } j \in \{i \in \mathcal{G} \mid l_i < y_i < u_i\}.$$

Consequently, all subproblems derived from the so extended input problem would have to contain these additional variables and constraints. In order to save the memory consumption and computational time for solving subproblems, we therefore decide not to add any pseudo-cuts in the VN diving variant for general MIP problems, although that implies possible repetitions in the search space exploration. This means that we only perform decomposition with respect to the LP relaxation solution of the initial problem. In this aspect, VN diving for general MIP problems is similar to the VNDS algorithm for 0-1 MIP problems from [19].

In order to avoid memory and time consumption when dealing with large problems, the implementation of the SN diving algorithm for general MIP problems is the same as for 0-1 MIP problems. In other words, all distance values are computed with respect to the distance function δ (which takes into account only binary variables), and general integer variables are handled by the generic MIP solver itself.

4. Computational Results

In this section we present the computational results for single and variable neighbourhood diving algorithms. We compare our proposed methods with the following existing methods CPLEX MIP solver without feasibility pump (`CPLEX` for short), the standard feasibility pump heuristic (`standard FP`), the objective feasibility pump (`Objective FP`) and the variable neighbourhood pump (`VNP`). Since the feasibility pump is already included as a primal heuristic in the employed version of the CPLEX MIP solver, we use the appropriate parameter settings to

control the use of FP and to chose the version of FP. All results reported are obtained on a computer with a 4.5GHz Intel Core i7-2700K Quad-Core processor and 32GB RAM, using the general purpose MIP solver IBM ILOG CPLEX 12.4. Both algorithms were implemented in C++ and compiled within Microsoft Visual Studio 2010. For comparison purposes, we consider 83 0-1 MIP instances [8]) previously used for testing the performance of the basic FP (see Table 1 and 34 general MIP instances previously used in [3] (see Table 2). In Tables 1and2, columns denoted by n represent the total number of variables, whereas columns denoted by $|\mathcal{B}|$ and m show the number of binary variables and the number of constraints, respectively. Additionally, the column denoted by $|\mathcal{G}|$ in Table 2 provides the number of general integer variables for a given instance.

In both proposed diving heuristics, the CPLEX MIP solver is used as a black-box for solving subproblems to feasibility. For this special purpose, the parameter `CPX_PARAM_MIP_EMPHASIS` is set to `FEASIBILITY`, the parameter `CPX_PARAM_INTSOLLIM` is set to 1 and the parameter `CPX_PARAM_FPHEUR` was set to -1. All other parameters are set to their default values, unless otherwise specified. Results for the CPLEX MIP solver without FP were obtained by setting the parameter `CPX_PARAM_FPHEUR` to -1. The feasibility pump heuristics are tested through the calls to the CPLEX MIP solver with the settings `CPX_PARAM_FPHEUR`=1 for standard FP and `CPX_PARAM_FPHEUR`=2 for objective FP. All tested methods (CPLEX MIP without FP, standard FP, objective FP and both proposed diving heuristics) were allowed 100 seconds of total running time on 0-1 MIP test instances, while on General MIP instances maximum running time, for all methods, was set to 150 seconds. In addition, the time limit for solving subproblems within variable neighbourhood diving and single neighbourhood diving was set to 10 seconds for all instances.

The value of the neighbourhood change control parameter d in the VN diving algorithm (see Figure 3) is set to 10, meaning that, in each iteration of VN diving, $10+\lfloor 1+\log_2 |\overline{x}_j \in \{0,1\} : j \in \mathcal{B}| \rfloor$ subproblems (i.e. neighbourhoods) are explored, where $\overline{x}$ is the LP relaxation solution of the current problem. The neighbourhood size control parameter α in the SN diving algorithm (see Figure 4) is set to 2.5, meaning that $\frac{1}{2.5} \times 100 = 40$ percent of the variables with integer values in $\overline{x}$ are initially fixed to those values to obtain the first subproblem. Those values of d and α are based on brief experimental analysis.

No.	Instance name	n	$\lvert\mathcal{B}\rvert$	m	No.	Instance name	n	$\lvert\mathcal{B}\rvert$	m
1	10teams	2025	1800	230	43	bg512142	792	240	1307
2	aflow30a	842	421	479	44	dg012142	2080	640	6310
3	aflow40b	2728	1364	1442	45	blp-ar98	16021	15806	1128
4	air04	8904	8904	823	46	blp-ic97	9845	9753	923
5	air05	7195	7195	426	47	blp-ic98	13640	13550	717
6	cap6000	6000	6000	2176	48	blp-ir98	6097	6031	486
7	dano3mip	13873	552	3202	49	CMS750_4	11697	7196	16381
8	danoint	521	56	664	50	berlin_5_8_0	1083	794	1532
9	ds	67732	67732	656	51	railway_8_1_0	1796	1177	2527
10	fast0507	63009	63009	507	52	glass4	322	302	396
11	fiber	1298	1254	363	53	net12	14115	1603	14021
12	fixnet6	878	378	478	54	nsrand_ipx	6621	6620	735
13	harp2	2993	2993	112	55	tr12-30	1080	360	750
14	liu	1156	1089	2178	56	van	12481	192	27331
15	markshare1	62	50	6	57	biella1	7328	6110	1203
16	markshare2	74	60	7	58	NSR8K	38356	32040	6284
17	mas74	151	150	13	59	rail507	63019	63009	509
18	mas76	151	150	12	60	rail2536c	15293	15284	2539
19	misc07	260	259	212	61	rail2586c	13226	13215	2589
20	mkc	5325	5323	3411	62	rail4284c	21714	21705	4284
21	mod011	10958	96	4480	63	rail4872c	24656	24645	4875
22	modglob	422	98	291	64	A1C1S1	3648	192	3312
23	momentum1	5174	2349	42680	65	A2C1S1	3648	192	3312
24	nw04	87482	87482	36	66	B1C1S1	3872	288	3904
25	opt1217	769	768	64	67	B2C1S1	3872	288	3904
26	p2756	2756	2756	755	68	sp97ar	14101	14101	1761
27	pk1	86	55	45	69	sp97ic	12497	12497	1033
28	pp08a	240	64	136	70	sp98ar	15085	15085	1435
29	pp08aCUTS	240	64	246	71	sp98ic	10894	10894	825
30	protfold	1835	1835	2112	72	usAbbrv.8.25_70	2312	1681	3291
31	qiu	840	48	1192	73	manpower1	10565	10564	25199
32	rd-rplusc-21	622	457	125899	74	manpower2	10009	10008	23881
33	set1ch	712	240	492	75	manpower3	10009	10008	23915
34	seymour	1372	1372	4944	76	manpower3a	10009	10008	23865
35	swath	6805	6724	884	77	manpower4	10009	10008	23914
36	t1717	73885	73885	551	78	manpower4a	10009	10008	23866
37	vpm2	378	168	234	79	ljb2	771	681	1482
38	dc1c	10039	8380	1649	80	ljb7	4163	3920	8133
39	dc1l	37297	35638	1653	81	ljb9	4721	4460	9231
40	dolom1	11612	9720	1803	82	ljb10	5496	5196	10742
41	siena1	13741	11775	2220	83	ljb12	4913	4633	9596
42	trento1	7687	6415	1265					

Table 1: Benchmark instances for 0-1 MIP.

No.	**Instance name**	n	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{G}\rvert$	m
1	arki001	1388	415	123	1048
2	atlanta-ip	48738	46667	106	21732
3	gesa2	1224	240	168	1392
4	gesa2-o	1224	384	336	1248
5	ic97_potential	728	450	73	1046
6	ic97_tension	703	176	4	319
7	icir97_potential	2112	1235	422	3314
8	icir97_tension	2494	262	573	1203
9	manna81	3321	18	3303	6480
10	momentum2	3732	1808	1	24237
11	momentum3	13532	6598	1	56822
12	msc98-ip	21143	20237	53	15850
13	mzzv11	10240	9989	251	9499
14	mzzv42z	11717	11482	235	10460
15	neos7	1556	434	20	1994
16	neos8	23228	23224	4	46324
17	neos10	23489	23484	5	46793
18	neos16	377	336	41	1018
19	noswot	128	75	25	182
20	rococoB10-011000	4456	4320	136	1667
21	rococoB10-011001	4456	4320	136	1677
22	rococoB11-010000	12376	12210	166	3792
23	rococoB11-110001	12431	12265	166	8148
24	rococoB12-111111	9109	8910	199	8978
25	rococoC10-001000	3117	2993	124	1293
26	rococoC10-100001	5864	5740	124	7596
27	rococoC11-010100	12321	12155	166	4010
28	rococoC11-011100	6491	6325	166	2367
29	rococoC12-100000	17299	17112	187	21550
30	rococoC12-111100	8619	8432	187	10842
31	rout	556	300	15	291
32	timtab1	397	64	107	171
33	timtab2	675	113	181	294
34	roll3000	1166	246	492	2295

Table 2: Benchmark instances for general MIP.

Model	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving	Model	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
1	952.00	952.00	964.00	**948.00**	994.00	956.00	43	915355705.00	915355705.00	915355705.00	**120738665.00**	**120738665.00**	**120738665.00**
2	**1244.00**	**1244.00**	**1244.00**	1609.00	1342.00	1375.00	44	1202855539.00	1202855539.00	1202855539.00	**153406945.50**	**153406945.50**	**153406945.50**
3	1502.00	4703.00	2361.00	1663.00	**1370.00**	1973.00	45	**6910.11**	**6910.11**	**6910.11**	7684.11	7082.24	9434.80
4	57528.00	57528.00	57528.00	59145.00	**56635.00**	59834.00	46	**4656.92**	**4656.92**	**4656.92**	4745.99	4965.41	4934.99
5	29888.00	29888.00	29888.00	31988.00	31988.00	**29077.00**	47	**5042.91**	**5042.91**	**5042.91**	5493.79	10372.78	13698.32
6	-106024.00	-106024.00	-106024.00	-2445700.00	**-2451175.00**	-1318020.00	48	**2433.13**	3101.46	3294.20	2616.77	2845.44	3974.77
7	**727.22**	**727.22**	**727.22**	768.38	768.38	5000.00	49	993.00	993.00	993.00	**258.00**	358.00	335.00
8	**66.50**	79.00	**66.50**	83.00	77.00	69.50	50	100.00	100.00	100.00	70.00	73.00	**67.00**
9	5418.56	5418.56	5418.56	5418.56	**1573.84**	4004.71	51	500.00	500.00	500.00	416.00	413.00	**408.00**
10	733.00	733.00	733.00	**181.00**	**181.00**	585.00	52	3691696333.33	3691696333.33	3691696333.33	5200046900.00	**2850022250.00**	3066694866.67
11	451769.10	**436408.71**	752876.96	613482.32	1400807.72	608394.16	53	**214.00**	337.00	**214.00**	296.00	255.00	295.00
12	**4505.00**	**4505.00**	**4505.00**	**4505.00**	8621.00	6044.00	54	261440.00	261440.00	261440.00	261760.00	**195200.00**	225920.00
13	-44633925.00	-44633925.00	-44633925.00	-42637348.00	**-72614611.00**	-54044012.00	55	140249.00	285747.00	154248.00	197146.00	**134912.00**	137933.00
14	6450.00	6450.00	6450.00	**5762.00**	**5762.00**	**5762.00**	56	64.00	64.00	64.00	43.61	**4.82**	64.00
15	7286.00	7286.00	7286.00	**230.00**	**230.00**	**230.00**	57	45112270.55	45112270.55	45112270.55	**13439555.63**	**13439555.63**	116460764.01
16	10512.00	10512.00	10512.00	**338.00**	**338.00**	**338.00**	58	5007115060.37	5007115060.37	5007115060.37	6336253949.24	6439612939.29	**2634420708.79**
17	157344.61	157344.61	157344.61	**14372.87**	**14372.87**	123739.83	59	451.00	451.00	451.00	**181.00**	**181.00**	398.00
18	157344.61	157344.61	157344.61	**43774.26**	**43774.26**	119400.61	60	1698.00	1698.00	1698.00	**711.00**	**711.00**	1889.00
19	3370.00	4365.00	3545.00	3620.00	**3330.00**	3375.00	61	2395.00	2395.00	2395.00	**1001.00**	**1001.00**	2348.00
20	0.00	0.00	0.00	0.00	**-261.75**	-92.21	62	2701.00	2701.00	2701.00	**1126.00**	**1126.00**	2805.00
21	0.00	0.00	0.00	0.00	0.00	0.00	63	3761.00	3761.00	3761.00	**1614.00**	**1614.00**	3400.00
22	36180511.32	36180511.32	36180511.32	**35147088.88**	**35147088.88**	**35147088.88**	64	19005.23	19005.23	15838.91	13081.23	**12259.60**	18962.51
23	**314555.10**	**314555.10**	**314555.10**	372338.73	372351.67	430397.54	65	20865.33	20865.33	20865.33	**11934.26**	12151.49	20625.65
24	**17004.00**	**17004.00**	**17004.00**	19792.00	19792.00	24228.00	66	69933.52	69933.52	69933.52	28798.94	**28034.72**	69933.52
25	0.00	0.00	0.00	**-16.00**	**-16.00**	0.00	67	70575.52	70575.52	70575.52	30885.28	**27474.62**	70375.52
26	3705.00	3705.00	3705.00	5728.00	**3131.00**	3711.00	68	11286473874.42	11286473874.42	11286473874.42	11286519578.42	**1149811855.24**	6241827883.02
27	731.00	731.00	731.00	**18.00**	**18.00**	**18.00**	69	1464309330.88	1464309330.88	1464309330.88	**747411597.12**	**747411597.12**	1312533290.88
28	27080.00	27080.00	27080.00	**9140.00**	10760.00	25160.00	70	2374928235.04	2374928235.04	2374928235.04	2349465005.92	**893040556.48**	4456259661.60
29	13690.00	13690.00	13690.00	**8250.00**	10940.00	11820.00	71	1695655079.52	1695655079.52	1695655079.52	**535501367.36**	658218799.04	1367805334.56
30	-13.00	-13.00	-13.00	-16.00	**-17.00**	-13.00	72	200.00	200.00	200.00	144.00	**131.00**	**131.00**
31	1805.18	1805.18	1805.18	385.91	**-1.21**	1577.11	73	**6.00**	9.00	**6.00**	**6.00**	7.00	**6.00**
32	186117.18	184040.51	184732.09	-	177132.88	**165894.84**	74	6.00	6.00	6.00	6.00	6.00	6.00
33	479735.75	479735.75	479735.75	389031.00	**77487.00**	235084.50	75	6.00	6.00	6.00	6.00	6.00	6.00
34	666.00	666.00	666.00	**473.00**	**473.00**	632.00	76	6.00	6.00	6.00	6.00	6.00	6.00
35	713.21	713.21	713.21	713.21	**512.09**	1955.96	77	**6.00**	**6.00**	**6.00**	**6.00**	7.00	**6.00**
36	258934.00	258934.00	258934.00	322075.00	320556.00	**218167.00**	78	6.00	6.00	6.00	6.00	6.00	6.00
37	17.00	17.00	17.00	15.75	19.75	**15.50**	79	**0.88**	**0.88**	**0.88**	7.24	7.24	7.24
38	1628753907.38	1628753907.38	1628753907.38	**15919302.95**	16271096.12	1169188836.35	80	**0.87**	**0.87**	**0.87**	8.61	8.61	8.61
39	16644728415.79	16644728415.79	16644728415.79	**26768702.57**	27048192.51	13567591658.00	81	**1.42**	**1.42**	**1.42**	9.48	9.48	9.48
40	2386640556.42	2386640556.42	2386640556.42	**199985211.16**	**199985211.16**	1866404347.13	82	**1.63**	**1.63**	**1.63**	7.31	7.31	7.31
41	560995341.87	560995341.87	560995341.87	**121151565.75**	**121151565.75**	327279193.17	83	**1.91**	**1.91**	**1.91**	6.20	6.20	6.20
42	5749161137.01	5749161137.01	5749161137.01	**429883219.07**	**429883219.07**	4438712711.00	*Wins*						

Table 3: Objective values for 0-1 MIP instances.

Model	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving	Model	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
1	0.96	1.23	1.20	0.61	0.52	**0.40**	43	**0.01**	**0.01**	**0.01**	0.08	0.05	0.05
2	0.09	0.04	**0.02**	0.05	0.23	0.05	44	**0.01**	**0.01**	**0.01**	0.32	0.18	0.18
3	0.24	**0.07**	0.13	0.55	3.21	0.24	45	2.11	2.38	2.33	2.59	10.78	**1.10**
4	**0.48**	0.49	0.53	2.21	0.98	0.56	46	0.20	**0.19**	**0.19**	0.74	5.68	0.23
5	**0.14**	**0.14**	**0.14**	0.42	0.27	0.35	47	0.42	0.39	0.40	1.03	0.44	**0.19**
6	**0.06**	**0.06**	**0.06**	**0.06**	0.30	0.07	48	0.56	0.18	0.20	0.64	0.57	**0.12**
7	59.58	60.21	60.00	5.31	**3.53**	13.11	49	0.22	**0.20**	**0.20**	31.46	17.4	5.23
8	0.36	0.22	0.68	0.25	**0.05**	0.10	50	**0.01**	**0.01**	**0.01**	0.44	0.64	0.25
9	0.42	**0.41**	0.42	13.40	78.78	364.69	51	**0.02**	0.03	**0.02**	3.75	1.34	0.28
10	**0.26**	**0.26**	**0.26**	1.45	0.92	0.84	52	**0.01**	**0.01**	**0.01**	0.04	0.13	**0.01**
11	0.02	**0.01**	0.02	0.03	**0.01**	0.02	53	29.52	1.88	30.18	6.37	2.31	1.70
12	**0.01**	**0.01**	**0.01**	0.02	0.02	**0.01**	54	0.14	**0.11**	**0.11**	0.43	0.38	0.12
13	**0.01**	**0.01**	**0.01**	0.04	0.22	**0.01**	55	0.86	**0.04**	**0.04**	18.50	10.03	0.35
14	**0.01**	**0.01**	**0.01**	0.03	0.03	0.02	56	48.06	48.33	47.96	100.19	12.92	**8.03**
15	0.00	0.00	0.00	0.00	0.00	0.00	57	**0.04**	**0.04**	**0.04**	0.88	0.59	0.50
16	0.00	0.00	0.00	0.00	0.00	0.00	58	13.77	13.75	**13.73**	46.77	34.79	29.49
17	0.00	0.00	0.00	0.00	0.00	0.00	59	**0.32**	**0.32**	**0.32**	2.10	1.50	1.08
18	0.00	0.00	0.00	0.00	0.00	0.00	60	**0.10**	**0.10**	**0.10**	4.18	2.47	2.37
19	0.14	0.05	0.07	0.09	0.04	**0.01**	61	0.20	**0.19**	0.20	3.21	1.94	1.83
20	**0.04**	**0.04**	**0.04**	0.13	0.08	**0.04**	62	0.33	**0.32**	0.35	14.99	8.71	8.08
21	**0.04**	**0.04**	**0.04**	0.05	**0.04**	**0.04**	63	0.41	**0.40**	0.41	8.36	5.42	4.28
22	**0.00**	**0.00**	**0.00**	0.01	**0.00**	**0.00**	64	**0.13**	0.51	**0.13**	15.81	10.37	0.26
23	0.77	**0.75**	**0.75**	15.01	24.82	12.67	65	**0.06**	**0.06**	**0.06**	24.17	10.09	0.09
24	1.05	1.23	1.30	0.60	0.42	**0.41**	66	**0.06**	**0.06**	0.07	19.26	10.11	0.10
25	**0.00**	0.01	0.01	0.01	**0.00**	0.01	67	**0.07**	**0.07**	**0.07**	16.99	10.13	0.09
26	**0.04**	**0.04**	**0.04**	0.07	0.34	0.17	68	**0.36**	**0.36**	**0.36**	3.19	1.55	0.90
27	0.00	0.00	0.00	0.00	0.00	0.00	69	0.22	**0.21**	**0.21**	0.87	0.55	0.54
28	**0.00**	**0.00**	**0.00**	0.03	0.01	**0.00**	70	0.35	**0.34**	**0.34**	2.75	1.07	0.99
29	**0.01**	**0.01**	**0.01**	0.05	**0.01**	**0.01**	71	0.25	**0.24**	0.25	1.59	0.76	0.51
30	16.04	16.33	16.36	4.54	10.69	**2.86**	72	**0.03**	**0.03**	**0.03**	13.49	11.24	0.98
31	**0.04**	**0.04**	**0.04**	1.34	0.99	0.06	73	1.23	**0.92**	0.93	5.10	2.86	2.25
32	29.61	37.02	44.55	100.06	26.47	**4.89**	74	**5.15**	5.64	5.56	10.74	6.68	7.59
33	**0.01**	**0.01**	**0.01**	0.04	**0.01**	**0.01**	75	**3.94**	4.77	4.79	9.60	7.46	4.57
34	**0.04**	**0.04**	**0.04**	1.19	0.59	0.67	76	**4.60**	5.37	5.48	14.03	9.05	5.12
35	**0.06**	**0.06**	**0.06**	0.20	20.17	20.36	77	**3.13**	3.70	3.77	12.62	7.64	4.16
36	100.15	100.48	100.61	**8.85**	14.85	10.57	78	**5.09**	5.87	5.92	12.32	9.52	7.08
37	**0.01**	**0.01**	**0.01**	0.02	0.02	**0.01**	79	**0.01**	**0.01**	**0.01**	0.03	0.03	0.03
38	**0.07**	**0.07**	**0.07**	3.25	1.87	1.72	80	0.15	**0.14**	**0.14**	0.65	0.58	0.57
39	0.26	**0.25**	**0.25**	5.98	2.50	2.32	81	0.19	0.19	**0.17**	0.85	0.72	0.70
40	**0.10**	**0.10**	**0.10**	3.16	2.11	1.87	82	0.23	0.24	**0.22**	1.33	1.13	1.12
41	2.32	**2.31**	2.28	7.75	5.48	5.71	83	**0.18**	**0.18**	**0.18**	0.96	0.86	0.81
42	**0.04**	**0.04**	**0.04**	2.55	1.57	1.66	*Avg.*	4.05	3.85	4.29	7.14	5.09	6.63

Table 4: Execution time values (in seconds) for 0-1 MIP instances.

	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
Solution quality						
Instances solved	83	83	83	82	83	83
Avg. gap from LP relaxation obj. w.r.t. all instances (%)	49665.28	49666.96	49649.94	-	6620.55	17890.24
Avg. gap from LP relaxation obj. w.r.t. 82 instances solved by VNP (%)	48002.46	48029.48	48003.82	4683.57	4542.36	16086.52
Number of wins	18	15	17	32	44	17
Computational time						
Average w.r.t. all instances(sec)	4.05	3.85	4.29	7.14	5.09	6.63
Average w.r.t. 82 instances solved by VNP	3.74	3.45	3.79	6.01	4.83	6.65
Number of wins	42	50	48	2	8	19

Table 5: Summarised results for 0-1 MIP instances.

The results obtained by all 6 solvers, for the first 83 benchmark 0-1 MIP instances, which was first used in [8], are presented in Tables 3 and 4. Table 3 provides the objective values obtained by all 6 methods and Table 4 provides the corresponding execution time. The summarized results for this benchmark, including the variable neighbourhood pump heuristic [13, 17], are presented in Table 5. In the solution quality block of Table 5, we provide the number of instances solved by each of the 6 methods, the average percentage gap from the LP relaxation objective value regarding all 83 instances, the average percentage gap from the LP relaxation objective value regarding the instances solved by VNP, and the number of times that each of the methods managed to obtain the best objective value among the others (including ties). For each method, a percentage gap for a particular instance was computed according to the formula $\frac{f-f_{LP}}{|f_{LP}|} \times 100$, where f is the objective function value for the observed instance obtained by that method, and f_{LP} is the objective function value of the LP relaxation of the observed instance. The exceptions are instances `markshare1`, `markshare2` and `mod011`, for which the gap value was computed as $(f - f_{LP}) \times 100$, since the LP relaxation objective value is equal to 0 for all three instances. In the computational time block of the Table 5, we provide the average computational time over all instances in the benchmark for each of the 6 methods compared, the average computational time over all instances solved by VNP, as well as the number of times that each of the methods managed to obtain a solution in shortest time.

Objective values							Running times						
Model	**CPLEX**	**Standard FP**	**Objective FP**	**VNP**	**VN Diving**	**SN Diving**	**Model**	**CPLEX**	**Standard FP**	**Objective FP**	**VNP**	**VN Diving**	**SN Diving**
1	7583912.00	7583912.00	7583912.00	**7583157.35**	7616973.21	7607607.29	1	0.50	0.87	0.84	1.44	10.2	**0.33**
2	106.01	106.01	106.01	-	**97.01**	128.01	2	122.88	123.42	124.53	150.08	45.31	**15.72**
3	83888091.14	83888091.14	83888091.14	166697234.63	**26334562.23**	68817832.42	3	0.02	**0.01**	0.02	0.06	0.04	0.02
4	**25792003.54**	29819721.89	26854088.34	**25792003.54**	26334562.23	25815849.31	4	0.04	0.04	0.05	0.09	0.04	**0.02**
5	4263.00	4206.00	4206.00	4275.00	**4042.00**	4279.00	5	**0.07**	0.42	0.47	20.38	1.52	0.12
6	4084.00	4077.00	4077.00	4086.00	4076.00	**4044.00**	6	0.17	0.52	0.54	2.54	0.73	**0.15**
7	7082.00	6966.00	6966.00	7088.00	**6449.00**	6845.00	7	1.03	0.69	0.65	31.05	10.28	**0.42**
8	6451.00	6470.00	6687.00	**6417.00**	6594.00	6630.00	8	4.67	6.17	**0.35**	22.01	11.18	1.58
9	0.00	0.00	0.00	**-6994.00**	**-6994.00**	**-6994.00**	9	**0.01**	**0.01**	**0.01**	0.11	0.06	0.06
10	**15218.97**	**15218.97**	**15218.97**	-	21216.59	25720.52	10	34.74	35.59	35.38	150.03	136.22	**13.88**
11	525000.00	525000.00	525000.00	525000.00	**325184.11**	496675.41	11	1.27	1.26	**1.25**	54.98	39.58	29.10
12	23307748.01	23307748.01	23307748.01	-	**22579326.01**	26502059.01	12	83.81	84.51	85.01	150.12	14.67	**4.98**
13	0.00	0.00	0.00	0.00	**-19078.00**	0.00	13	**0.71**	0.76	**0.71**	8.47	7.17	3.28
14	0.00	0.00	0.00	0.00	**-20370.00**	-9052.00	14	0.83	**0.81**	**0.81**	10.12	12.72	4.57
15	**723934.00**	5741899.00	853405.60	**723934.00**	4183899.00	**723934.00**	15	0.23	**0.07**	0.09	0.19	0.14	0.13
16	0.00	0.00	0.00	0.00	**-3719.00**	0.00	16	**0.14**	**0.14**	**0.14**	8.17	4.72	4.95
17	2.00	2.00	2.00	2.00	**-952.00**	2.00	17	0.14	0.13	**0.12**	3.00	2.75	1.99
18	447.00	447.00	451.00	**446.00**	450.00	450.00	18	13.11	11.58	11.95	8.95	13.79	**7.7**
19	**-40.00**	**-40.00**	**-40.00**	-34.00	-34.00	-34.00	19	0.01	0.01	0.01	**0.00**	**0.00**	**0.00**
20	55270.00	55270.00	55270.00	**31330.00**	52926.00	34068.00	20	0.17	0.18	0.18	0.85	**0.16**	0.19
21	59350.00	59350.00	59350.00	61176.00	52926.00	**37673.00**	21	0.17	0.18	0.18	0.60	**0.16**	0.41
22	99336.00	99336.00	99336.00	99336.00	96932.00	**96692.00**	22	0.36	0.38	0.39	1.31	**0.35**	0.50
23	99336.00	99336.00	99336.00	99336.00	99771.00	**99152.00**	23	0.70	0.75	0.72	2.17	**0.69**	0.95
24	55765.00	59139.00	56732.00	65252.00	**48062.00**	53453.00	24	50.89	36.91	32.63	11.74	11.03	**4.91**
25	26633.00	26633.00	26633.00	**17843.00**	23730.00	30062.00	25	0.09	**0.08**	**0.08**	0.33	**0.08**	0.09
26	34954.00	87563.00	**28095.00**	37521.00	69850.00	32054.00	26	2.92	0.38	0.88	1.26	**0.15**	0.25
27	146524.00	146524.00	146524.00	146524.00	**143170.00**	146524.00	27	0.47	0.40	0.40	1.27	**0.32**	0.37
28	146524.00	146524.00	146524.00	**83853.00**	128760.00	146524.00	28	0.24	0.24	0.23	0.77	0.22	**0.19**
29	**52189.00**	**52189.00**	**52189.00**	63480.00	117774.00	74202.00	29	2.91	3.19	3.29	3.92	0.74	**0.91**
30	102274.00	82848.00	82848.00	110131.00	115593.00	**47383.00**	30	0.73	1.08	1.08	1.61	**0.48**	0.63
31	2375.25	2375.25	2375.25	2375.25	**1231.38**	1383.98	31	**0.01**	**0.01**	**0.01**	0.05	0.18	0.08
32	1040909.00	986130.00	1114433.00	1115617.00	**846107.00**	1055386.00	32	0.16	0.12	**0.05**	0.69	1.12	0.08
33	1549355.00	1403843.00	1544127.00	1691420.00	**1231580.00**	1492526.00	33	0.72	2.63	2.7	0.82	10.05	**0.18**
34	15850.00	15850.00	15850.00	16259.00	**13095.00**	16049.00	34	0.74	1.05	1.01	1.56	2.58	**0.48**
Wins	5	3	4	9	17	7	*Avg.*	9.58	9.25	9.02	19.14	9.98	2.92

Table 6: Solution quality and running time performance for general MIP instances.

From Tables 3 and 5, we can see that all methods except VNP are able to solve all 83 instances. VNP does not manage to solve just one test instance. Therefore, the comparison of performances of CPLEX MIP solver, standard FP, objective FP, VN diving and SN diving has been done regarding all 83 instances, while the performances of VNP has been evaluated relatively to the performances of the previous five methods regarding 82 instances solved by VNP. It appears that VN diving clearly outperforms all other methods regarding the solution quality. Indeed, it manages to solve all 83 instances from the benchmark and has the smallest average gap (6620.55%) from the LP relaxation objective. In addition, VN diving provides the best objective values among all 6 methods in 44 out of 83 instances That is much more than number of times that VNP (32 times), CPLEX MIP solver(18 times), objective FP(17 times) or standard FP(17 time) succeeds to reach best objective value. The second best among methods able to solve all 83 instances is SN diving with an average gap from the LP relaxation of 17890.24%. It is followed by objective FP (49649.94%), standard FP (49666.96 %) and CPLEX MIP solver (49665.28 %). On the other hand, regarding 82 instances solved by VNP, VNP has much smaller average gap from LP relaxation objective (4683.57%) in comparison with SN diving (16086.52%), CPLEX MIP solver(48002.46%), objective FP (48003.82 %)and standard FP (48029.48 %). However with respect to the average gap from LP relaxation objective, VNP is the second best method. Its average gap is slightly greater than the average gap of VN diving whose gap is 4542.36%.

From Tables 4 and 5, we can observe that the shortest average computational time of 3.85s is reported by standard FP, whereas objective FP and CPLEX MIP solver are only slightly slower with the average computational time of 4.29s and 4.05s, respectively. They are followed by VN diving, whose average computational time is 5.09s, whereas SN diving and VNP are the slowest, with 6.63s and 7.14s average computational time, respectively. Note, that in computation of average computational time of VNP, we include the time of its failed run. Also, note that on one instance (i.e., ds), we allowed to SNdiving more than 100s of computational time and counted that run as successful. However, if we consider the average computational time of all six methods over all instances solved successfully by each of them (82 instances solved by VNP), the ranking of methods is almost unchanged besides that VNP is now faster than SN diving. Regarding the number of wins, the objective FP, the standard FP, and the CPLEX MIP manage to obtain a solution in the shortest time most often, in 50, 48 and 42 cases, respectively. The SN diving and VN diving follow, obtaining a solution in the shortest time in 19, and 8 cases, respectively. The VNP has the worst performance in this respect, since it finds a solution before other methods in just two cases.

The objective function values and the corresponding execution time for the second benchmark of 34 general MIP instances [3] are presented in Table 6. Summarised results for this benchmark are presented in Table 7. For each method, a percentage gap for a particular instance was computed according to the formula

$$\frac{f - f_{LP}}{|f_{LP}|} \times 100,$$

	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
Solution quality						
Instances solved	34	34	34	31	34	34
Avg. gap from LP relaxation obj. w.r.t all instances(%)	403.44	454.18	407.46	-	383.28	381.08
Avg. gap from LP relaxation obj. w.r.t instances solved by VNP(%)	437.31	492.95	441.72	431.42	413.00	406.69
Number of wins	5	3	4	9	17	7
Computational time						
Average w.r.t all instances(sec)	9.58	9.25	9.02	19.14	9.98	2.92
Average w.r.t instances solved by VNP(sec)	2.72	2.29	1.99	6.47	4.62	2.09
Number of wins	5	7	10	1	9	14

Table 7: Summarised results for general MIP instances.

where f is the objective function value for the observed instance obtained by that method, and f_{LP} is the objective function value of the LP relaxation of the observed instance. Note that for this benchmark set, there is no exception to this rule since there is no instance whose LP objectives is equal to 0.

From Tables 6 and 7, we can see that again only the VNP is not able to solve all 34 instances. Therefore, the comparison of performances of CPLEX without FP, standard FP, objective FP, VN diving, and SN diving has been done in the same way as for the previous benchmark set. From Tables 3 and 5, we conclude that VN diving and SN diving have best performances regarding the solution quality. The SN diving heuristic achieves the smallest average gap from the LP objective (381.08%) and obtains the best objective among all 6 methods in 7 cases. The VN diving has a slightly worse average gap of 383.28%, but obtains the best objective among all methods in 17 cases. If we take into account the average computational time of these two methods, we may conclude that SN diving is the best method for the general MIP problem. The third best method appears to be the CPLEX MIP solver without FP, with 403.44% average LP relaxation gap and 5 wins, followed by objective FP with 407.46% average gap and 4 wins. The standard FP heuristics have a significantly higher gap from the LP relaxation (454.18%) and only 3 objective wins, indicating that FP is the worst choice quality-wise for the general MIP benchmark. Moreover, the ranking of CPLEX without FP, standard FP, objective FP, VN diving, and SN diving regarding solution quality on instances solved by VNP is the same. However, on these instances, VNP manifests much better behavior than CPLEX without FP, standard FP, objective FP regarding the average gap from the LP value. Additionally, VNP has 9 objective wins, indicating that VNP is the second best method, after VN diving, regarding the number of wins.

From Tables 6 and 7, we can see that SN diving achieves the impressive average execution time of 2.92s. The next method, according to the average execution time, is the objective FP heuristic which is more than three times slower, with average computational time of 9.02s. It is followed by standard FP with 9.25s average time, the CPLEX MIP solver without FP with 9.58s average time, VN diving(9.98s), and finally the VNP heuristic, which is the slowest method with 19.14s average computational time. Moreover, the ranking of methods remains the same even in case that the average computational times are computed regarding instances solved by VNP. Regarding number of wins, the SN diving manages to obtain a solution in the shortest time in 14 cases. The objective FP, VN diving, standard

FP, and CPLEX MIP solver follow by obtaining a solution in the shortest time in 10, 9, 7, 5 cases, respectively. The VNP has the worst performance, since it manages to find a solution before other methods in just one case.

According to the experimental analysis above, our two proposed diving heuristics generally provide solutions of a better quality than the CPLEX MIP solver and the two FP heuristics, within a similar or shorter computational time. Although the VNP heuristic proves to be highly competitive for the 0-1 MIP benchmark, it shows a rather poor performance for the general MIP benchmark. We may therefore claim that, in overall, VN diving heuristic and SN diving outperform all four state-of-the-art solvers which were used for comparison purposes regarding solution quality. Additionally, we may claim that SN diving significantly outperforms all tested methods regarding average computational time needed to provide a feasible solution for the instances from General MIP benchmark.

4.1. Influence of the time limit on the performances of all six methods

In this section we check the imposed time limit influence on the number of solved instances by each method. The results are given in Table 8 and Figure 5 for 0-1 MIP instances, and Table 6 and Figure 6 for General MIP benchmark instances.

Time limit (s)	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
1	67	67	67	40	43	54
5	74	74	73	58	58	70
10	76	77	76	66	67	77
20	78	79	78	78	78	80
30	80	79	78	79	81	82
40	80	80	79	80	82	82
50	81	81	81	81	82	82
60	82	81	82	81	82	82
70	82	82	82	81	82	82
80	82	82	82	81	83	82
90	82	82	82	81	83	82
100	83	83	83	82	83	82

Table 8: Number of solved instances by 6 methods as a function of time limit - 0-1 MIP

It appears that CPLEX MIP solver, standard FP, and objective FP perform better if the time limit is less than 10s. However, increasing the time limit, the number of solved instances by the other methods grows dramatically. Consequently, when the time limit is set to 20 seconds, SN diving becomes the method with the most solved instances, keeping the first place until time limit is extended to 80 seconds, when VN diving becomes the best method able to solve all instances.

From Table 9 and Figure 6, we conclude that CPLEX MIP solver, standard FP, objective FP, and SN diving are able to find a feasible solution within 1 second. The CPLEX MIP solver and objective FP manage to solve 24 instances out of 34 within 1 second, while standard FP and SN diving succeed to get 23

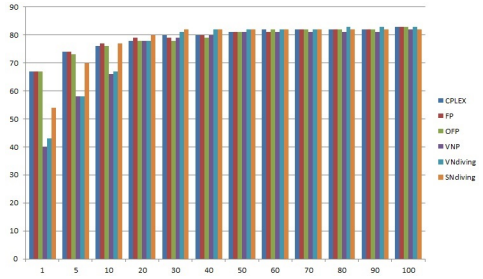

Figure 5: Number of solved instances by 6 methods as a function of time limit - 0-1 MIP

Time limit (s)	CPLEX	Standard FP	Objective FP	VNP	VN Diving	SN Diving
1	24	23	24	12	17	23
5	29	28	29	22	22	30
10	29	29	29	25	23	31
20	30	30	30	27	31	33
30	30	30	30	29	31	34
40	31	32	32	30	32	34
50	31	32	32	30	33	34
60	32	32	32	31	33	34
70	32	32	32	31	33	34
80	32	32	32	31	33	34
90	33	33	33	31	33	34
100	33	33	33	31	33	34
110	33	33	33	31	33	34
120	33	33	33	31	33	34
130	34	34	34	31	33	34
140	34	34	34	31	34	34
150	34	34	34	31	34	34

Table 9: Number of solved instances by 6 methods as a function of time limit - General MIP

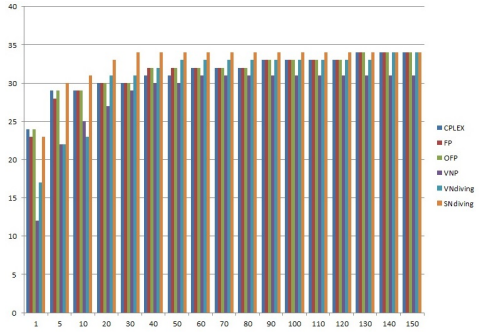

Figure 6: Number of solved instances by 6 methods as a function of time limit - General MIP

out of 34 instances in less than 1 second. Furthermore, it appears that SN diving outperforms all other methods if the time limits is greater than 1s. Moreover, SN diving solves all instances when the time limit is adjusted to 30 seconds; that is the smallest time limit that one method needs to solve all instances. Taking into account our previous observations, one can conclude that SN diving is the best heuristic for finding initial feasible solution for general MIP instances.

5. Conclusion

In this paper we propose two new heuristics for finding initial feasible solutions of mixed integer programs (MIPs). The proposed heuristics, called *variable neighbourhood diving* (*VN diving*) and *single neighbourhood diving* (*SN diving*), perform systematic hard variable fixing (i.e. diving) in order to generate smaller subproblems whose feasible solution (if one exists) is also feasible for the original problem. In VN diving, this fixation is performed according to the rules of variable neighbourhood decomposition search (VNDS) [15]. This means that a number of subproblems (neighbourhoods) generated in a VNDS manner are explored in each iteration. Also, pseudo-cuts are added during the search process in order to prevent exploration of already visited search space areas. However, a feasible solution is usually obtained in the first iteration. In SN diving, only one neighbourhood is explored in each iteration. However, we introduce a new mechanism to avoid the already visited solutions. It consists of memorising a set of constraints in a new MIP problem, which is then solved instead of the original problem in order to obtain the new reference solution. Our experiments show that this mechanism generally provides much better diversification than the addition of pseudo-cuts alone. Moreover, we have proved that the SN diving algorithm converges to a feasible solution, if one exists, or proves the infeasibility in a finite number of iterations. Both methods use the generic CPLEX MIP solver as a black-box for tackling the subproblems generated during the search.

The proposed heuristics are tested on two established sets of benchmark instances, proven to be difficult: the set first contains 83 0-1 MIP instances [8], and the second contains 34 general MIP instances [3]. We compare our heuristics with the IBM ILOG CPLEX 12.4 MIP solver, the two variants of the feasibility pump (FP) heuristic (standard FP and objective FP), and the variable neighbourhood pump (VNP) heuristic [13, 17, 18]. According to an extensive experimental analysis, both VN and SN diving clearly outperform the CPLEX MIP solver and the two FP heuristics regarding the solution quality, within a similar or shorter computational time. Additionally, on the instances from General MIP benchmark, SN diving performs better than any other method tested in this paper, regarding not only solution quality but also the time needed to find a feasible solution. The results reported in this paper are also competitive with those obtained by the recent variable neighbourhood pump heuristic and its extensions [13, 17, 18]. Besides improving the basic variable neighbourhood pump, our future work may consist of designing a multi-objective VNS heuristic, which would tackle both infeasibility and original objective quality during the search process.

Acknowledgements

The present research work has been supported by International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research, and the French National Center for Scientific Research. The presented research has also been supported by the project No. 174010 "Large scale optimization models and methods with applications", funded by the Serbian Ministry of Science. We also thank to anonymous reviewers for their valuable comments and suggestions. All contributions are gratefully acknowledged.

References

[1] Achterberg, T., and Berthold, T. Improving the feasibility pump. *Discrete Optimization 4* (2007), 77–86.
[2] Balas, E., and Zemel, E. An algorithm for large zero-one knapsack problems. *Operations Research* (1980), 1130–1154.
[3] Bertacco, L., Fischetti, M., and Lodi, A. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization 4* (2007), 63–76.
[4] Berthold, T. RENS – relaxation enforced neighborhood search. Technical report, ZIB-07-28. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2008.
[5] Brucker, P., Burke, E., and Groenemeyer, S. A mixed integer programming model for the cyclic job-shop problem with transportation. *Discrete Applied Mathematics 160*, 13 (2012), 1924–1935.
[6] Collet, G., Andonov, R., Yanev, N., and Gibrat, J. Local protein threading by Mixed Integer Programming. *Discrete Applied Mathematics 159*, 16 (2011), 1707–1716.
[7] Danna, E., Rothberg, E., and Le Pape, C. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming 102*, 1 (2005), 71–90.
[8] Fischetti, M., Glover, F., and Lodi, A. The feasibility pump. *Mathematical Programming 104* (2005), 91–104.
[9] Fischetti, M., and Lodi, A. Local branching. *Mathematical Programming 98*, 2 (2003), 23–47.
[10] Fischetti, M., and Lodi, A. Repairing mip infeasibility through local branching. *Computers & OR 35* (2008), 1436–1445.
[11] Garey, M., and Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman San Francisco, 1979.
[12] Ghosh, S. DINS, a MIP improvement heuristic. In *Proceedings of the Integer Programming and Combinatorial Optimization*, M. Fischetti and D. P. Williamson, Eds., vol. 4513 of *Lecture Notes in Computer Science*. Springer, 2007, pp. 310–323.
[13] Hanafi, S., Lazić, J., and Mladenović, N. Variable Neighbourhood Pump Heuristic for 0-1 Mixed Integer Programming Feasibility. *Electronic Notes in Discrete Mathematics 36* (2010), 759–766.
[14] Hansen, P., and Mladenović, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research 130*, 3 (2001), 449–467.
[15] Hansen, P., Mladenović, N., and Perez-Britos, D. Variable Neighborhood Decomposition Search. *Journal of Heuristics 7*, 4 (2001), 335–350.
[16] Hansen, P., Mladenović, N., and Urošević, D. Variable neighborhood search and local branching. *Computers & OR 33*, 10 (2006), 3034–3045.
[17] Lazić, J. *New Variants of Variable Neighbourhood Search for Mixed Integer Programming and Clustering*. PhD thesis, Brunel University, West London, UK, 2010.
[18] Lazić, J., Hanafi, S., and Mladenović, N. Different Variants of Variable Neighbourhood Pump for 0-1 MIP Feasibility, 2010. (In preparation).

[19] Lazić, J., Hanafi, S., Mladenović, N., and Urošević, D. Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers & OR 37*, 6 (2010), 1055–1067.
[20] Mitrović-Minić, S., and Punnen, A. Very large-scale variable neighborhood search for the generalized assignment problem. *accepted for publication in Journal of Interdisciplinary Mathematics* (2008).
[21] Mitrović-Minić, S., and Punnen, A. Variable Intensity Local Search. In *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, V. Maniezzo, T. Stützle, and S. Voß, Eds. Springer, 2009, pp. 245–252.
[22] Mitrović-Minić, S., and Punnen, A. Very large-scale variable neighborhood search for the multi-resource generalized assignment problem. *Discrete Optimization 6*, 4 (2009), 370–377.
[23] Pisinger, D. An expanding-core algorithm for the exact 0-1 knapsack problem. *European Journal of Operational Research 87*, 1 (1995), 175–187.
[24] Puchinger, J., and Raidl, G. Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. *Journal of Heuristics 14*, 5 (2008), 457–472.
[25] Puchinger, J., Raidl, G., and Pferschy, U. The core concept for the multidimensional knapsack problem. *Lecture Notes in Computer Science 3906* (2006), 195–208.
[26] Soyster, A., Lev, B., and Slivka, W. Zero-One Programming with Many Variables and Few Constraints. *European Journal of Operational Research 2*, 3 (1978), 195–201.
[27] Wilbaut, C., Salhi, S., and Hanafi, S. An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research 199*, 2 (2009), 339–348.
[28] Wolsey, L., and Nemhauser, G. Integer and Combinatorial Optimization, 1999.