Noname manuscript No.
(will be inserted by the editor)

Guided Dive for the Spatial Branch-and-Bound

D. Gerard · M. Köppe · Q. Louveaux

September 12, 2016

Abstract We study the spatial Brand-and-Bound algorithm for the global optimization of nonlinear problems. In particular we are interested in a method to find quickly good feasible solutions. Most spatial Branch-and-Bound-based solvers use a non-global solver at a few nodes to try to find better incumbents. We show that it is possible to improve the branching rules and the node priority by exploiting the solutions from the non-global solver. We also propose several smart adaptive strategies to choose when to run the non-global solver. We show that despite the time spent in solving more NLP problems in the nodes, the new strategies enable the algorithm to find the first good incumbents faster and to prove the global optimality faster. Numerous easy, medium size as well as hard NLP instances from the Coconut library are benchmarked. All experiments are run using the open source solver Couenne.

Keywords spatial branch-and-bound, local search, heuristic, guided dive, branching, Couenne

1 Introduction

A wide range of engineering and scientific applications may be written as a nonlinear optimization problem, e.g. electric [29], water [24] and gas [35] distribution networks, contigency analysis in electric networks [13], nuclear reactor reloading

D. Gerard
Department of Electrical Engineering and Computer Science, University of Liège, Liège 4000, Belgium
E-mail: damien.gerard@ulg.ac.be

M. Köppe
Department of Mathematics, University of California, Davis, CA 95616, U.S.A.
E-mail: mkoeppe@math.ucdavis.edu

Q. Louveaux
Department of Electrical Engineering and Computer Science, University of Liège, Liège 4000, Belgium
E-mail: q.louveaux@ulg.ac.be

patterns [36], reinforced concrete beams optimization [10], lens design in optics [22]. Many applications can also be found in chemical engineering [6], such as chemical processes control [7], life cycle optimization for sustainable design [20] and production planning in multi-plant facilities [23]. Other applications include channel interference [11] and optimal resource management [39] in wireless networks, optimal path for vehicles [1, 21], traffic modeling [18], conflict resolution involving multiple aircraft [37] and flight clearance [15].

Definition 1 Smooth nonlinear programming (NLP) problems are conveniently expressed as

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & c(x) \leq 0 & \\ & L_i \leq x_i \leq U_i \qquad i \in N & \end{array} \qquad (\text{NLP}(L,U))$$

where $N = \{1, \ldots, n\}$, $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions.

Any (NLP(L,U)) problem can be solved by a NLP solver. The NLP solvers are categorized depending on the methods implemented: sequential quadratic programming [8], interior point [32], penalty [14], augmented lagrangian [34], trust-region [9], filter [17]... Those strategies guarantee the global convergence to a local optimum, i.e. the convergence to a point satisfying the KKT conditions from any starting point.

If (NLP(L,U)) is nonconvex, the local optimum towards which a NLP solver converges may not be the global optimum. The local optimum objective value may also be far from the global optimum. Making sure to find the global optimum of a problem is computationally demanding.

The best-known method for solving nonconvex NLP problems to global optimality is the *spatial Branch-and-Bound* (sBB) [38, 40], implemented for instance in BARON [41], Couenne [3], ANTIGONE [27], LindoGlobal [26], SCIP [5], α-BB [2] and COCONUT [33]. The sBB explores the whole feasible set of the problem. In all sBB implementations, the solvers make use of the best feasible solution known so far to *prune* some regions which cannot contain any better solution. Hence, finding good feasible solutions quickly is of paramount importance to prune as much space as possible. This paper focuses on new variants of the sBB algorithm to find good feasible solutions quicker. All the ideas are implemented in Couenne, a well-known open-source global optimization software.

The urge behind obtaining good feasible solutions quickly finds its root in Mixed-Integer Programming (MIP). Some MIP models remain very hard to solve to optimality. In such cases, several heuristics are known to sometimes provide good feasible solutions quickly. Variable Neighborhood Search (VNS) [28] seeks a feasible solution in a neighborhood and subsequently adds a perturbation to move to another neighborhood. Local Branching [16] explores the neighborhood of a feasible solution by increasing or decreasing the value of a few integer variables while excluding the previously found incumbent with a so called 'no-good cut'. Relaxation Induced Neighborhood Search (RINS) [12] is based on the intuition that there is a link between the incumbent and the solution of the continuous relaxation. Many variables take the same values between the two. Therefore, RINS fixes

them and solves a sub-MIP on the remaining variables to hopefully find a solution which achieves both integrality and a good objective value. Relaxation Enforced Neighborhood Search (RENS) [4] explores the neighborhood of the solution of the continuous relaxation by rounding all integer variables. [12] also introduces the guided dive, which uses the current incumbent solution to decide to dive in the left or right child node first.

The heuristics in MIP gave birth to several heuristics for nonconvex Mixed Integer Nonlinear Programming (MINLP). [25] uses VNS for the global search phase and a black-box convex MINLP solver for the local search. [31] argues that a local search performed with a convex MINLP solver is significantly slower than it was in MIP. Therefore, [31] proposes to avoid the convex MINLP solver and consider a sequence of limited-size MILPs and NLPs instead. The NLP estimates the descent direction of the original objective function and the MILP solves a convexification with a local branching constraint. [30] also performs a sequence of NLPs and MIPs, but with a rounding of the integer variables. However, research still lacks insights on how to adapt these heuristics for nonconvex NLP without any integer variables, besides using a black-box NLP solver.

We base our work on the aforementioned works, in particular the guided dive of [12], and aim at extending it to nonconvex NLP.

The sBB recursively partitions the feasible set into subsets such that every solution to (NLP(L,U)) is feasible in one of the subsets.

Each subset is associated with a node uniquely defined by a specific set of bounds (l,u) such that $L_i \leq l_i \leq u_i \leq U_i$ for each $i \in N$, and corresponds to the NLP

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & c(x) \leq 0 \\ & l_i \leq x_i \leq u_i \qquad i \in N \end{aligned} \tag{nodeNLP(l,u))}$$

The feasible set of (nodeNLP(l,u)) is

$$\Omega(l,u) = \{x \in \mathbb{R}^n : c(x) \leq 0; l_i \leq x_i \leq u_i; i = 1,\dots,n\} \tag{1}$$

Definition 2 At a node (nodeNLP(l,u)), consider a variable $x_b, b \in N$ on which we *branch*, and let $\{x_b^{(1)},\dots,x_b^{(K+1)}\}$ be a sequence of *branching values* such that $x_b^{(1)} = l_b$, $x_b^{(K+1)} = u_b$ and $x_b^{(k)} \leq x_b^{(k+1)}$ for $k = 1,\dots,K$. The *multiway branching* defines the K partitions of $\Omega(l,u)$, also referred to as *child nodes*, as

$$\Omega^{(k)} = \Omega(l,u) \cap \{x \in \mathbb{R}^n : x_b^{(k)} \leq x_b \leq x_b^{(k+1)}\} \quad k = 1,\dots,K \tag{2}$$

$\Omega^{(k)}$ is the feasible set of nodeNLP$_k(l,u)$.

In practice, all the newly created nodes are stored in a queue Q.

Definition 3 Let Q be a set of nodes $\{\text{nodeNLP}_j(l,u)\}$ to solve for $j = 1,\dots,|Q|$, among which a sBB implementation must choose the next node to proceed. A *node selection strategy* assigns a priority $p(j)$ to each node in Q. The next node to select is nodeNLP$_j(l,u)$ with $j \in \arg\max_j p(j)$.

Most sBB implementations also factorize the original problem (nodeNLP(l,u)) into a series of equivalent constraints for which a convex relaxation can be derived [19].

One iteration of the sBB algorithm involves selecting a problem (nodeNLP(l,u)) $\in Q$ with the highest priority p (Definition 3). Its convexification is solved, which yields the point $x^{(l,u)}$. If $x^{(l,u)} \notin \Omega(l,u)$, the node is partitioned according to Definition 2. (nodeNLP(l,u)) may also be solved with an NLP solver to obtain a local optimum $h^{(l,u)}$.

Upon partitioning a set into two subsets or more, one may use for instance the *best bound* node comparison strategy to decide which branch is to be explored first. However, so far, there is no NLP-specific rule to guess which branch to explore first to find feasible solutions with a good objective value quickly. This paper aims at introducing new such rules and provides extensive tests to assess how effective they are on real-world and academic problems. In particular, Couenne runs IPOPT (an NLP solver) on (nodeNLP(l,u)) at a few upper levels of the sBB tree to try to find better incumbents. We show that it is possible to improve the branching rules by exploiting the solution of (nodeNLP(l,u)) provided by the NLP solver.

The remainder of this paper is organized as follows. Section 2 introduces the guided dive for MIP and proposes an extension for nonconvex NLP. Section 3 incorporates the guided dive and its specificities into a multiway branching scheme. Finally, Section 4 provides extensive tests and conclusions for all the features introduced. Section 5 concludes and gives directions for further research.

2 Guided dive

First introduced by [12] for MIP, the guided dive consists in a small modification of the tree traversal strategy. In the branch-and-bound branching procedure, the first decision is the selection of a variable to branch on, and the second is the choice of which child node to explore first. Let x^* be the best incumbent known so far, and x_b be the selected branching variable at node (nodeNLP(l,u)). The guided dive explores first the child κ of (nodeNLP(l,u)) such that $x_b^* \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$, according to Definition 2. The guided dive and the depth-first tree traversal, by making the Branch-and-Bound dive towards the best incumbent and then backtrack the tree, are together somewhat similar to local branching heuristics which explicitly explore the neighborhood of x^*.

It is important to point out that although $x_b^* \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$, the best incumbent x^* may not be feasible in (nodeNLP(l,u)), or in any of its child nodes nodeNLP$_k(l,u)$, $k = 1, \ldots, K$. Unlike local branching heuristics, the guided dive also makes use of the incumbent even if it is not located in the neighborhood of x^*, provided $x_b^* \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$ only. The incumbent could therefore be far from $\Omega(l,u)$. [12] notes that despite its simplicity, this approach is quite effective at improving the solutions found in MIP. We extend the guided dive to the sBB for nonconvex NLP problems.

In Section 4, we first show computationally that as for MIP problems, the guided dive leads to a major improvement for nonconvex NLP problems solved to global optimality. The first limitation of the original guided dive is that there

are many nodes which cannot be guided if $x_b^* \notin [l_b, u_b]$. A blind child selection must therefore be performed. In practice, the solvers use a random or a fixed child selection. Couenne belongs to the latter by diving in the leftmost child first.

In the sBB, an NLP solver is run at several nodes at the upper levels of the tree, each potentially providing one NLP feasible solution. Finding feasible solutions may be hard. In this paper, we consider problems for which at least a few feasible solutions are found throughout the sBB. Let $F^{(l,u)}$ be the set of all known feasible points when node (nodeNLP(l,u)) is processed. We improve the guided dive by using the points in $F^{(l,u)}$, which includes

1. former incumbents before a new better incumbent was found;
2. newly computed feasible points which are not better than the incumbent.

At node (nodeNLP(l,u)), with the branching variable x_b, let $H^{(l,u)} = \{x \in F^{(l,u)} : l_b \leq x_b \leq u_b\}$ be the set of known feasible points for which the value of x_b lies within the bounds $[l_b, u_b]$ of (nodeNLP(l,u)). Our improved guided dive chooses the feasible point $h^{(l,u)} \in H^{(l,u)}$ with the best objective value to direct the dive at node (nodeNLP(l,u)).

In some cases, the feasible point $h^{(l,u)}$ used by one node (nodeNLP(l,u)) can be reused in some of its child nodes, thereby avoiding to solve some new NLP problems. Denote by κ the child node of (nodeNLP(l,u)) such that $h_b^{(l,u)} \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$, according to Definition 2. There are two cases to consider. Case 1 considers $h^{(l,u)} \in \Omega(l,u)$ whereas case 2 considers $h^{(l,u)} \notin \Omega(l,u)$.

Case 1. $h^{(l,u)} \in \Omega(l,u)$ and $h_b^{(l,u)} \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$ imply that $h^{(l,u)} \in \Omega^{(\kappa)}$, the feasible set of nodeNLP$_\kappa(l,u)$. $h^{(l,u)}$ can be reused by nodeNLP$_\kappa(l,u)$, for any branching variable selected when processing nodeNLP$_\kappa(l,u)$. For the other child nodes, $h^{(l,u)}$ cannot be reused if the branching variable is x_b and sometimes cannot be reused for the other branching variables $x_g, g \in N\backslash\{b\}$ if $h_g^{(l,u)}$ is excluded from the tightened bounds $[l_g, u_g]$ by the bound tightening procedures.

Case 2. $h^{(l,u)} \notin \Omega(l,u)$ and $h_b^{(l,u)} \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$ imply that $h^{(l,u)}$ can only be reused by nodeNLP$_\kappa(l,u)$ provided its branching variable is also x_b. For the other branching variables of nodeNLP$_\kappa(l,u)$ as well as for all the branching variables of the other child nodes, there is no garantee that $h^{(l,u)}$ can be reused.

In most sBB implementations, at the v upper levels of the tree, the NLP problem is always solved by the NLP solver, starting from the solution of the convexification. The aim is to find new incumbents early. Below the v level, i.e. for all the lower levels of the tree, no NLP problem is solved.

At the beginning of the tree, the major source of feasible points comes from running a local NLP solver at several nodes, the other one being solutions of the convexification provided they are nonlinear feasible. However, solving local NLP problems has a non-negligible running time. We further improve the algorithm with the main goal of solving a NLP problem when it is most useful.

In our implementation, we replace the v threshold with the v_1 and v_2 thresholds instead ($v_1 \leq v_2$). Above the v_1 level, the NLP problem is always solved because the partitioning and the bounds tightening techniques at the upper levels of the tree are likely to enable the NLP solver to find new incumbents. Between the v_1

and v_2 levels, the NLP problem is solved if $H^{(l,u)} = \emptyset$. It is more useful to solve a NLP problem if $H^{(l,u)} = \emptyset$, because the solution obtained will direct the dive instead of having to perform a blind diving. Below the v_2 level, no NLP problem is solved. However, if $H^{(l,u)} \neq \emptyset$ for a node (nodeNLP(l,u)) below the v_2 level, then one feasible point $h^{(l,u)} \in H^{(l,u)}$ is used for the child selection; otherwise the default blind diving is performed.

3 Multiway branching

Couenne implements the branching procedure of Definition 2 for $K = 2$. By default, for continuous variables, the branching value is $\bar{x}_b = \alpha x_b^{(l,u)} + (1-\alpha)x_b^M$, where $x^{(l,u)}$ is the solution of the convex relaxation of (nodeNLP(l,u)) and $x_b^M = \frac{l_b+u_b}{2}$. This branching-point selection is refered to as *mid-point*. The branching is performed by adding the inequality $x_b \leq \bar{x}_b$ for the left child nodeNLP$_1(l,u)$ and $x_b \geq \bar{x}_b$ for the right child nodeNLP$_2(l,u)$. There are several other branching-point selection rules described in [3]. All of them aim at improving the convexification of the subsequent child nodes. However, none of them adapt their branching value to try to dive as fast as possible towards regions with higher odds of finding better feasible solutions. The previous section argued that diving in the child node containing the incumbent has a positive impact. This section explores modifications of the branching values to try to achieve even better performance. For the remainder of this section, we rewrite $h^{(l,u)}$ as h.

Figure 1 and Figure 2 both illustrate a disadvantageous 2-way branching. In both figures, the left child is the vertically hatched region on the left and the right child is the horizontally hatched region on the right. In Figure 1, the branching value is $\bar{x}_b$. With the mid-point selection $\bar{x}_b = \alpha x_b^{(l,u)} + (1-\alpha)x_b^M$ and the default value $\alpha = 0.25$, the relative distance $\frac{|x_b^M - \bar{x}_b|}{u_b - l_b}$ is at most $\frac{\alpha}{2} = 0.125$, placing $\bar{x}_b$ relatively close to the center $x_b^M = \frac{l_b+u_b}{2}$. However, the solution h_b of (nodeNLP(l,u)) may lie anywhere in the range $[l_b, u_b]$, regularly resulting in situations like the one depicted in Figure 1, where h_b is close to u_b. According to its definition, the guided dive chooses the right child first because it contains h_b. However, the span $[\bar{x}_b, u_b]$ is fairly wide, which prevents a fast dive towards the close neighborhood of h.

Figure 2 illustrates another partitioning where another branching value delimits a relatively short span containing h_b. Diving in the right child first enables the guided dive to explore the neighborhood of h quickly. However, dismissing $\bar{x}_b$ as the branching point has the major drawback of not focusing on improving the convexification of the child nodes anymore, which was the purpose of $\bar{x}_b$. Hence the convexification of the vertically hatched area is barely improved. While the partitioning of Figure 2 may help find new incumbents faster, it impedes on the global convergence of the sBB. We propose to take these considerations into account by keeping the branching value $\bar{x}_b$ and further partitioning with a second branching value, as defined in Definition 4 and depicted in Figure 3 (we can assume $\bar{x}_b \leq h_b$ with no loss of generality).

Definition 4 The *guided 3-way partitioning* defines a set of 2 branching values $\{\bar{x}_b,\ h_b - \Delta_1\}$ (resp. $\{\bar{x}_b, h_b + \Delta_2\}$) which delimit the 3 parts according to Definition 2. The guided 3-way partitioning creates the second branching point either on the left ($h_b - \Delta_1$) or on the right ($h_b + \Delta_2$) of h_b, depending on the location of

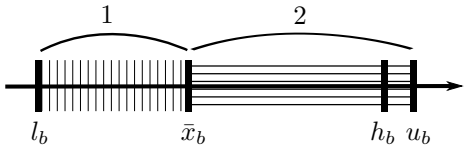

Fig. 1: Branching value $\bar{x}_b$

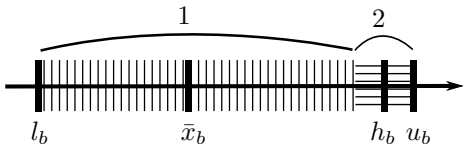

Fig. 2: Branching value close to h_b

$\bar{x}_b$ and h_b. Δ_1 and Δ_2 are margins which are a fraction of $u_b - l_b$. The branching value $h_b - \Delta_1$ is possible if $\bar{x}_b \leq h_b - \Delta_1$ and the branching value $h_b + \Delta_2$ is possible if $h_b + \Delta_2 \leq u_b$. In Definition 3, the 3 parts are each given a priority p such that $p(3) > p(2) > p(1)$ (resp. $p(2) > p(3) > p(1)$ with $\{\bar{x}_b, h_b + \Delta_2\}$), i.e. the 3$^{\text{rd}}$ child is to be solved before the 2$^{\text{nd}}$ child, which is to be solved before the 1$^{\text{st}}$ child.

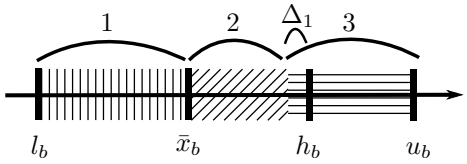

Fig. 3: Guided 3-way partitioning with branching values $\{\bar{x}_b,\ h_b - \Delta_1\}$

Definition 5 extends the guided 3-way by allowing one additional branching point on the other side of h_b. Illustration in Figure 4.

Definition 5 The *guided 4-way partitioning* defines a set of 3 branching values $\{\bar{x}_b,\ h_b - \Delta_1,\ h_b + \Delta_2\}$ which delimit the 4 parts according to Definition 2. Δ_1 and Δ_2 are margins which are a fraction of $u_b - l_b$. In Definition 3, they are each given a priority p such that $p(3) > p(2) > p(4) > p(1)$

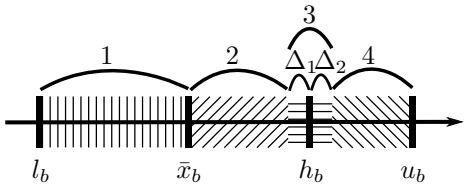

Fig. 4: Guided 4-way partitioning

Finally, we define the *guided 2-way partitioning* in Definition 6.

Definition 6 The *guided 2-way partitioning* defines $\{\bar{x}_b\}$ as the sole branching value which delimits the 2 partitions according to Definition 2. In Definition 3, they are each given a priority p such that $p(1) > p(2)$ if $h_b \in [l_b, \bar{x}_b]$, and $p(2) > p(1)$ otherwise.

Creating more nodes in a sBB results in a noticeable computing overhead, more than in MIP. For each sBB node, several bound tightening algorithms are performed, a convexification is computed and solved, and an NLP solver is possibly run. This suggests that the 3-way and a fortiori the 4-way partition should be scarcely used. At each node (nodeNLP(l,u)), the 2-way, the 3-way or the 4-way is chosen wisely to avoid creating too many small nodes. From the 4-way partitioning (Definition 5), if $h_b - \Delta_1 - \bar{x}_b < \delta$ for some value $\delta \geq 0$ (i.e. part 2 is too small), parts 2 and 3 are merged, thereby resulting in a 3-way partitioning. If $u_b - h_b - \Delta_2 < \delta$ (i.e. part 4 is too small), parts 3 and 4 are merged. The priority $p(k)$ is set such that $p(3) > p(2)$ (if not merged) $> p(4)$ (if not merged) $> p(1)$.

4 Computational results

Our features are implemented in Couenne 0.4 and benchmarked on an Intel Core i7 at 3.47GHz. Several different versions of the diving strategies were tested on NLP instances from the Coconut library. Among the whole Coconut test set (around 600 problems), some problems were discarded:

- 258 problems solved almost instantaneously;
- 12 problems claimed by IPOPT to be infeasible because it makes Couenne assume that the problem is infeasible and stop;
- 26 problems for which numerical issues were encountered by the original Couenne or our modified versions;
- 78 problems with no solution found in the sBB within the two hours time limit, besides the local solution found by IPOPT.

This leaves us with 170 usable problems for testing, further split into

- 117 easy or medium size problems;
- 53 hard problems.

A problem is considered hard if at least the original Couenne or one of our modified versions does not converge to global optimality within the two hours time limit. Moreover, besides the local solution found by IPOPT, at least one feasible solution with a better objective value must be found in the sBB by the original Couenne or one of our modified versions. If no better solution can be found in the sBB, the branching strategies could not be compared.

Our measurements are:

1. total running time (resp. total number of processed nodes) until proof of global optimality;
2. time (resp. number of processed nodes) to close 60% of the gap between the first feasible solution f_0 found at root node and the global optimum f^*.

We compare the original Couenne (C) with our modified versions (M). The time gain between the two algorithms is $\frac{t_M - t_C}{\max(t_M, t_C)}$, where t is either the total running time or the time to close 60% of the gap. If the time gain is negative, the modified version is faster than Couenne. Otherwise Couenne runs faster than our modified verison. To compare the total number of nodes or the number of nodes processed to close 60% of the gap, the formula $\frac{n_M - n_C}{\max(n_M, n_C)}$ is used, where n is the number of nodes.

4.1 Guided dive

Table 1 shows the average gains over the easy and medium size problems for the different diving strategies. Test 1 (first row in Table 1) shows the gain if Couenne dives first in the child node $\text{nodeNLP}_\kappa(l, u)$ *containing* the best incumbent x^*, i.e. $x^* \in \Omega(l, u)$, which is similar to a local branching heuristic. If $x^* \notin \Omega(l, u)$, the default diving of Couenne is performed. Finding feasible solutions helps closing the 60% gap quicker. However, there is little impact on the total running time, besides a few branches closed thanks to the incumbent. Generating tighter convexifications has a much larger impact on the total running time. We refer the reader to [3].

Test 2 is the guided dive as originally defined in [12] for MIP, which dives in the child node $\text{nodeNLP}_\kappa(l, u)$ such that $x_b^* \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$, even if $x^* \notin \Omega^{(\kappa)}$. As for MIP problems, the guided dive also leads to a significant gain for NLP problems, with respect to diving in the child node containing x^*. To show how important the guided dive is, the inverted guided dive in test 3 first dives in the child node $\text{nodeNLP}_\tau(l, u)$ such that $x_b^* \notin [x_b^{(\tau)}, x_b^{(\tau+1)}]$, the opposite of test 2. The slightly positive gain values for the inverted guided dive mean that it is slightly worse than the default diving of Couenne, which dives in the leftmost child first.

In test 4, the feasible point considered for the diving is either computed at the current node or inherited from a parent node, not necessarily the best incumbent known so far. The gain is not as good as the guided dive in test 2. This points out that it is more advantageous to use the best incumbent instead of several other feasible points with a weaker objective value, although the latter allow the sBB to use the guided dive more often.

In test 5, we use the best incumbent if possible as in test 2; otherwise we use another feasible point inherited from a parent node as in test 4. If $x_b^* \notin [l_b, u_b]$, the best incumbent cannot be used and a blind dive must be performed. To avoid this as much as possible, we use another known feasible solution from a parent or the solution from a local NLP solver at the current node, if solved. This improves the guided dive based only on the best incumbent.

In test 6, from $H^{(l,u)} = \{x \in F^{(l,u)} : l_b \leq x_b \leq u_b\}$, the set of known feasible points for which the value of x_b lies within the bounds $[l_b, u_b]$, we consider the feasible point $h^{(l,u)} \in H^{(l,u)}$ with the best objective value, no matter its location in the variable space. This further slightly improves the performance. Since $h^{(l,u)}$ may be spatially far from $\Omega(l, u)$, it is reasonable to try promote a solution $\hat{h}^{(l,u)} \in F^{(l,u)}$ closer to $\Omega(l, u)$, albeit with a weaker objective value. A weighting between the value of the objective function and the distance has been carried out and benchmarked. However, all the weighting values tested yield a deterioration of the gain. It seems that the feasible solution with the best objective value should always be used first, no matter how far it is from $\Omega(l, u)$.

Test 7 uses the two depth thresholds v_1 and v_2. The values used are $v_1 = \lceil 0.57e \rceil$ and $v_2 = \lceil 0.91e \rceil$, where e is the number of variables of the extended reformulation [19]. Between the v_1 and v_2 levels, the NLP problem is not solved if $H^{(l,u)} \neq \emptyset$. The sBB tree is expected to barely be affected by not computing those solutions. Indeed, the total number of nodes and the number of nodes to close 60% of the gap are almost unchanged. However, computing fewer local NLP problems saves time. The time gain depicted in Table 1 is rather limited because we also used a parameter preventing Couenne from solving more than 10 local NLP problems in the whole tree, so the algorithm of test 7 only avoids a few calls to IPOPT. This parameter, among others, was used to improve the original Couenne. If one were to increase this parameter, the improvement from test 6 to test 7 would be larger, but the gains of all the other tests, from tests 1 to test 6, would slightly drop.

#	Dive condition	Number of nodes		Time (s)	
		60% gap	Total	60% gap	Total
1	In node *containing* the best incumbent x^*	-9.33%	-3.27%	-8.16%	-0.74%
2	In node κ such that $x_b^* \in [x_b^{(\kappa)}, x_b^{(\kappa+1)}]$	-20.29%	-0.35%	-15.55%	1.08%
3	Inverted guided dive	3.47%	1.86%	1.75%	5.65%
4	With parent heuristic	-13.86%	-1.95%	-12.45%	-0.12%
5	With x^* or parent heuristic	-25.30%	-2.23%	-18.55%	1.74%
6	With feasible point with best obj. value	-26.88%	-5.81%	-18.89%	-2.45%
7	With v_1 and v_2	-27.70%	-5.31%	-25.29%	-8.53%

Table 1: Diving strategies

Table 2 shows how often on average the first feasible solutions (ordered by objective value, as in test 6) are used to direct the dive at the nodes of the sBB. The current incumbent x^* is used if $x_b^* \in [l_b, u_b]$. If not, the 2^{nd} best known feasible solution x' is used if $x'_b \in [l_b, u_b]$. If not, the 3^{rd} best known feasible solution x'' is used if $x''_b \in [l_b, u_b]$ etc. The most remarkable result is the very scarce use of the 2^{nd} and next feasible solutions. The feasible solutions are spatially closed to one another. If $x_b^* \notin [l_b, u_b]$, the next feasible solutions are unlikely to have their x_b value in the range $[l_b, u_b]$. Still, using all the known feasible solutions results in the noticeable improvement from test 2 to test 6.

	Percentage of use
Current incumbent	70.44%
Current 2^{nd} best solution	1.77%
Current 3^{rd} best solution	1.17%
Current 4^{th} best solution	0.73%
Current 5^{th} best solution	0.50%
Current 6^{th} best solution	0.48%
Current 7^{th} best solution	0.27%
...	
No known feasible solution usable	23.68%

Table 2: Incumbents use on average

Table 6 details the time gain between the original Couenne and our modified versions test 2, test 6 and test 7, for each problem. For convenience, the best value is depicted in bold for each problem.

Table 3 compares the time gains of the 53 hard instances. 'TL' stands for the Time Limit of two hours. Since the global solution f^* was not found within two hours for most problems, the 60% gap is considered to be between the first feasible solution f_0 found at root node and the best solution found among the original Couenne and our test 2 and test 6 variants, instead of f^*. Hence, the time to close 60% of the gap is still related with the speed to find the first good feasible solutions. The three rightmost columns show the gap closed between f_0 and the aforementioned best solution known. The large number of '0.0% TL' occurences for the gap closed by Couenne means that beyond the initial feasible solution f_0, the sBB did not manage to find any better feasible solution within the time limit.

	60% primal gap time (s)					Gap closed (%)		
Problem	Couenne	Test 2	Gain	Test 6	Gain	Couenne	Test 2	Test 6
etamac	TL	2.57	-100%	**2.35**	-100%	0.0% TL	100% TL	100% TL
ex5_2_5	TL	TL	-0.0%	**2.73**	-100%	100% TL	100% TL	100% TL
ex5_3_3	5.78	5.05	-12.7%	**4.88**	-15.7%	100% TL	100% TL	100% TL
ex8_3_14	4.99	3.06	-38.8%	**2.94**	-41.1%	66.8% TL	100% TL	100% TL
ex8_4_2	TL	**4.38**	-99.9%	4.83	-99.9%	0.0% TL	100% TL	100% TL
ex8_4_6	TL	**1.13**	-100%	5.13	-99.9%	0.0% TL	100% TL	97.0% TL
ex8_4_7	TL	**2.87**	-100%	6.11	-99.9%	0.0% TL	100% TL	100% TL
ex8_4_8	TL	8.10	-99.9%	**4.64**	-99.9%	0.0% TL	100% TL	100% TL
ex8_5_6	TL	**19.2**	-99.8%	33.6	-99.6%	0.0% TL	**100%**	**100%**
expquad	TL	**36.8**	-99.6%	36.9	-99.6%	0.0% TL	100% TL	100% TL
hadamals	TL	**47.2**	-99.3%	47.2	-99.3%	0.0% TL	84.6% TL	100% TL
haifam	TL	57.3	-99.2%	**56.1**	-99.2%	0.0% TL	66.6% TL	100% TL
himmelbf	TL	2.57	-100%	**2.57**	-100%	0.0% TL	100% TL	100% TL
hs056	**0.12**	15.1	99.2%	TL	100%	92.4% TL	100% TL	0.0% TL
hs059	0.11	0.11	-3.2%	**0.09**	-19.4%	**100%**	**100%**	**100%**
hs101	TL	**505**	-93.0%	3630	-49.6%	0.0% TL	27.6% TL	100% TL
hs103	TL	**1.41**	-100%	1.43	-100%	0.0% TL	100% TL	100% TL
hs109	TL	41.7	-99.4%	**27.0**	-99.6%	0.0% TL	100% TL	100% TL
hs110	TL	**0.74**	-100%	0.77	-100%	0.0% TL	100% TL	99.9% TL
hs111lnp	TL	**0.49**	-100%	0.98	-100%	0.0% TL	100% TL	100% TL
hs112	3278	0.53	-100%	**0.30**	-100%	**100%**	**100%**	**100%**
hs117	3.12	0.29	-90.6%	**0.18**	-94.3%	100% TL	100% TL	100% TL
hs268	TL	**0.14**	-100%	0.16	-100.0%	0.0% TL	100% TL	100% TL
kowosb	TL	**19.0**	-99.7%	22.3	-99.7%	0.0% TL	**100%**	**100%**
lakes	TL	TL	0.0%	**359**	-95.0%	100% TL	100% TL	100% TL
least	TL	**3.87**	-99.9%	4.35	-99.9%	0.0% TL	100% TL	100% TL
mistake	1.02	**0.33**	-67.3%	0.34	-66.6%	92.7% TL	100% TL	100% TL
optmass	2415	**9.09**	-99.6%	TL	66.5%	0.0% TL	0.0% TL	100% TL
orthrds2	**25.7**	35.2	27.0%	35.6	27.9%	100% TL	100% TL	100% TL
osborneb	TL	145	-98.0%	**130**	-98.2%	0.0% TL	100% TL	100% TL
palmer1a	6.99	**0.34**	-95.2%	0.34	-95.2%	74.4% TL	100% TL	100% TL
palmer1b	0.77	**0.35**	-54.6%	0.37	-51.6%	36.0% TL	100% TL	100% TL
palmer2a	TL	**0.35**	-100%	0.35	-100%	0.0% TL	100% TL	100% TL

	60% primal gap time (s)					Gap closed (%)		
Problem	Couenne	Test 2	Gain	Test 6	Gain	Couenne	Test 2	Test 6
palmer3	**0.49**	193	99.7%	180	99.7%	**100%**	**100%**	**100%**
palmer3a	TL	0.37	-100%	**0.34**	-100%	0.0% TL	100% TL	100% TL
palmer3b	**0.26**	0.37	27.7%	0.37	28.5%	75.3% TL	100% TL	100% TL
palmer4	**0.43**	163	99.7%	142	99.7%	83.2% TL	**100%**	**100%**
palmer4b	0.40	0.35	-13.3%	**0.26**	-35.5%	100% TL	**100%**	**100%**
palmer6a	55.0	**0.31**	-99.4%	0.31	-99.4%	100% TL	96.6% TL	96.6% TL
palmer7a	TL	31.1	-99.6%	**15.0**	-99.8%	0.0% TL	100% TL	74.9% TL
palmer7e	**3.27**	TL	100%	TL	100%	100% TL	0.0% TL	0.0% TL
palmer8a	1.98	1.69	-14.5%	**0.21**	-89.4%	72.0% TL	72.0% TL	86.2% TL
penalty2	6309	16.7	-99.7%	**8.99**	-99.9%	4.9% TL	97.5% TL	97.5% TL
pindyck	TL	**5.32**	-99.9%	5.68	-99.9%	0.0% TL	100% TL	100% TL
qp1	TL	**85.1**	-98.8%	95.2	-98.7%	0.0% TL	100% TL	100% TL
qp2	TL	**71.4**	-99.0%	77.4	-98.9%	0.0% TL	100% TL	100% TL
qp3	TL	**6.24**	-99.9%	6.48	-99.9%	0.0% TL	100% TL	100% TL
s368	**194**	206	6.0%	227	14.4%	100% TL	99.4% TL	99.4% TL
sineali	56.5	**21.8**	-61.4%	123	54.2%	100% TL	72.7% TL	69.3% TL
snake	142	527	73.0%	**68.4**	-51.9%	**100%**	100% TL	**100%**
ssnlbeam	TL	**0.53**	-100.0%	1.04	-100%	0.0% TL	100% TL	100.0% TL
swopf	TL	6.73	-99.9%	**5.38**	-99.9%	0.0% TL	100% TL	86.9% TL
weeds	4.10	**2.45**	-40.1%	2.67	-34.8%	100% TL	88.5% TL	88.5% TL
Mean			-59.30%		-61.14%			

Table 3: Results for the hard problems

Although this paper focuses on continuous variables, we also benchmarked mixed-integer nonlinear problem (MINLP) instances from the MINLPlib library to assess the effects of our algorithm on instances with both continuous and integer variables. As for the continuous problems, we let Couenne choose the variable to branch on (whether integer or continuous) and refer the reader to [3]. In the *Continuous* variant, we choose the diving side as in test 6 only if the branching variable is continuous, never modifying the diving side of integer variables. In the *All* variant, we apply test 6 on both continuous and integer variables. Table 5 compares the time gains of the 60 easy or medium instances, whereas Table 5 compares the 21 hard instances. Since all the hard instances hit the time limit of two hours, the right columns of Table 5 show the gap closed instead of the total time gain.

By default, Couenne dives left on an integer variable if $\bar{x}_b - \lfloor \bar{x}_b \rfloor < \lceil \bar{x}_b \rceil - \bar{x}_b$ and dives right otherwise. While the guided dive gives a noticeable improvement if it is performed on continuous variables, it brings no further improvement if also performed on integer variables. However in [12], the guided dive was quite effective in MIP. While the guided dive performs well on integer variables in MIP and on continuous variables in NLP and MINLP, it has little effect if performed on integer variables in MINLP. [3] benchmarked several MINLP instances with most variable selection strategies. The comparison did not show a clear winner and [3] stated that the performance is highly dependent on the instance. We also run tests with the original Couenne and several variable selection strategies (most fractional, random, strong branching, reliability branching, three variants of the

strong branching, violation transfer) and obtained similar results. From our results, the difference in gain (average on the easy and medium MINLP problems) with the 60% primal gap and with the total running time is below 2.28% between the variable selection strategies. We believe that the guided dive is uneffective on the integer variables of MINLP problems for the same reason.

	60% primal gap time (s)					Total time (s)				
Problem	Coue.	Cont.	Gain	All	Gain	Coue.	Cont.	Gain	All	Gain
alan	**0.05**	0.05	4.9%	0.05	7.4%	0.08	**0.07**	-13.1%	0.07	-6.8%
batch	**0.62**	0.70	10.2%	1.10	43.1%	**0.63**	0.70	10.2%	1.10	43.3%
csched1a	0.66	0.48	-28.2%	**0.28**	-57.8%	0.67	0.48	-27.8%	**0.44**	-33.6%
du-opt	2.60	**2.59**	-0.4%	3.20	18.7%	26.4	20.0	-24.0%	**17.2**	-34.8%
du-opt5	3.06	**2.31**	-24.3%	4.92	37.9%	47.9	**28.8**	-39.9%	31.6	-34.1%
elf	1.05	**0.65**	-37.8%	1.01	-3.7%	2.98	2.45	-17.7%	**2.21**	-25.8%
eniplac	359	281	-21.6%	**199**	-44.6%	428	377	-11.9%	**263**	-38.7%
ex1243	2.51	**1.53**	-39.0%	2.08	-16.9%	3.54	**2.14**	-39.6%	2.53	-28.6%
ex1244	2.94	**1.77**	-39.8%	4.46	33.9%	7.58	**4.69**	-38.2%	7.22	-4.8%
ex1252	**2.54**	2.77	8.2%	9.39	73.0%	14.0	**5.09**	-63.7%	13.9	-0.9%
ex1252a	1.14	**0.47**	-58.7%	0.48	-58.2%	9.01	5.02	-44.3%	**4.84**	-46.3%
ex1263a	4.29	**1.83**	-57.3%	2.81	-34.6%	6.55	**3.03**	-53.7%	4.60	-29.7%
ex1264a	0.49	0.31	-37.2%	**0.31**	-37.6%	1.06	**0.65**	-39.1%	0.65	-38.9%
ex1265a	1.10	**0.68**	-37.8%	0.71	-35.4%	6.58	4.75	-27.8%	**4.15**	-36.9%
ex1266a	0.84	**0.57**	-32.4%	0.81	-3.4%	9.63	**6.47**	-32.8%	9.66	0.3%
ex4	14.7	8.58	-41.5%	**2.27**	-84.5%	29.3	18.5	-37.0%	**4.40**	-85.0%
fac2	15.5	12.1	-21.9%	**8.63**	-44.5%	16.5	13.1	-20.7%	**9.21**	-44.3%
fac3	1.37	1.30	-5.3%	**0.89**	-35.1%	1.37	1.30	-5.3%	**0.89**	-35.1%
feedtray2	15.8	**12.2**	-22.7%	12.3	-22.1%	15.8	**12.2**	-22.7%	12.3	-22.1%
fo7_ar2_1	32.6	32.2	-1.1%	**21.6**	-33.6%	2255	1727	-23.4%	**1421**	-37.0%
gastrans	4.65	**2.20**	-52.8%	2.22	-52.3%	7.87	**2.20**	-72.1%	2.22	-71.8%
gear	0.02	0.01	-46.5%	**0.01**	-69.5%	0.02	0.01	-46.5%	**0.01**	-69.5%
gear2	0.12	**0.08**	-29.4%	0.10	-15.6%	0.12	**0.08**	-29.6%	0.10	-15.1%
gear3	0.04	0.02	-44.1%	**0.02**	-52.4%	0.04	0.02	-44.1%	**0.02**	-52.4%
gear4	0.15	**0.07**	-55.3%	0.07	-53.0%	0.81	0.48	-40.0%	**0.47**	-41.9%
ghg_1veh	270	1.54	-99.4%	**1.53**	-99.4%	1095	973	-11.1%	**876**	-20.0%
m3	2.05	1.33	-35.3%	**0.99**	-51.9%	2.06	1.33	-35.3%	**1.16**	-43.7%
m6	2.42	**1.58**	-34.8%	2.28	-5.8%	64.2	43.1	-32.9%	**40.8**	-36.4%
m7	10.1	6.81	-32.2%	**6.78**	-32.6%	839	**587**	-30.0%	593	-29.4%
m7_ar25_1	4.36	2.91	-33.3%	**2.79**	-36.1%	36.5	**25.2**	-30.9%	26.3	-28.0%
m7_ar2_1	22.5	15.5	-31.0%	**10.0**	-55.5%	128	90.3	-29.2%	**88.1**	-30.9%
m7_ar3_1	9.02	**6.11**	-32.3%	6.39	-29.2%	284	306	7.3%	**211**	-25.8%
m7_ar4_1	**29.9**	31.6	5.2%	35.2	14.9%	231	**218**	-5.5%	251	8.1%
m7_ar5_1	86.7	**61.4**	-29.2%	101	13.8%	459	**331**	-27.9%	400	-12.9%
meanvarx	0.71	**0.44**	-37.4%	0.47	-33.7%	1.35	**0.80**	-40.7%	0.91	-32.7%
meanvarxsc	0.67	0.45	-33.2%	**0.28**	-57.9%	1.32	**0.80**	-39.4%	0.82	-37.8%
minlphix	1.49	0.95	-36.5%	**0.92**	-38.3%	2.52	**1.59**	-37.0%	1.59	-36.9%
nous2	16.4	**10.7**	-34.7%	10.7	-34.7%	16.4	**10.7**	-34.7%	10.7	-34.7%
nvs13	0.32	**0.16**	-50.7%	0.17	-48.1%	0.49	**0.26**	-46.9%	0.27	-44.4%
nvs17	**0.43**	0.47	8.2%	0.48	9.9%	3.64	2.23	-38.7%	**2.18**	-40.0%

	60% primal gap time (s)					Total time (s)				
Problem	Coue.	Cont.	Gain	All	Gain	Coue.	Cont.	Gain	All	Gain
nvs18	0.22	0.14	-38.7%	**0.13**	-39.9%	1.16	0.77	-34.0%	**0.68**	-41.0%
nvs19	0.52	**0.28**	-46.5%	0.29	-44.6%	14.0	**8.43**	-39.8%	8.45	-39.7%
nvs20	**0.52**	0.81	35.6%	0.80	35.0%	1.30	0.92	-29.2%	**0.92**	-29.7%
nvs23	0.79	**0.39**	-50.2%	0.40	-49.5%	34.3	**13.6**	-60.4%	13.7	-60.2%
nvs24	0.95	0.62	-35.0%	**0.62**	-35.4%	86.1	37.8	-56.1%	**37.7**	-56.2%
parallel	30.0	**20.3**	-32.4%	50.8	40.9%	252	**75.0**	-70.2%	105	-58.4%
pump	**1.10**	4.19	73.7%	4.03	72.6%	7.53	5.54	-26.5%	**5.43**	-27.9%
sep1	0.74	0.39	-47.2%	**0.28**	-62.0%	0.75	0.39	-47.6%	**0.31**	-59.0%
spectra2	1.37	0.87	-36.8%	**0.44**	-67.8%	43.3	26.3	-39.2%	**3.41**	-92.1%
spring	0.24	**0.16**	-34.4%	0.25	6.2%	0.45	**0.29**	-35.6%	0.31	-30.3%
st_e31	5.83	3.41	-41.5%	**1.52**	-73.9%	11.6	6.30	-45.6%	**6.24**	-46.2%
st_e40	0.20	**0.19**	-2.2%	0.20	-0.2%	0.22	**0.22**	-0.9%	0.22	-0.8%
st_testgr1	0.05	0.03	-48.1%	**0.03**	-48.3%	0.11	0.07	-38.7%	**0.07**	-39.7%
st_testgr3	0.84	10.2	91.8%	**0.30**	-64.4%	8.13	12.6	35.6%	**2.86**	-64.9%
synthes3	0.28	0.18	-35.1%	**0.18**	-35.4%	0.28	0.18	-35.2%	**0.18**	-35.3%
tln4	**0.18**	5.92	97.0%	5.95	97.0%	301	189	-37.1%	**174**	-42.2%
tloss	1.03	1.00	-2.8%	**0.59**	-42.7%	7.01	6.41	-8.5%	**4.56**	-34.9%
tls2	0.51	**0.31**	-39.5%	0.32	-38.2%	0.51	**0.31**	-39.5%	0.32	-38.2%
tltr	5.20	**1.49**	-71.4%	1.49	-71.3%	6.12	2.31	-62.2%	**2.15**	-64.8%
util	2.04	1.20	-41.3%	**1.14**	-44.1%	6.74	4.04	-40.0%	**2.16**	-68.0%
Mean			-25.40%		-24.86%			-32.48%		-36.09%

Table 4: Results for the easy and medium size MINLP problems

	60% primal gap time (s)					Gap closed (%)		
Problem	Couenne	Cont.	Gain	All	Gain	Couenne	Cont.	All
fo7	3.18	3.12	-1.9%	**2.72**	-14.7%	100% TL	100% TL	100% TL
fo7_2	2.06	**2.03**	-1.3%	2.04	-1.0%	100% TL	100% TL	100% TL
fo7_ar25_1	**29.0**	29.6	2.1%	31.2	7.1%	100% TL	100% TL	100% TL
fo7_ar3_1	4.64	4.41	-4.9%	**4.23**	-8.7%	100% TL	100% TL	100% TL
fo7_ar4_1	9.09	9.20	1.3%	**8.98**	-1.1%	**100%**	100% TL	**100%**
fo7_ar5_1	164	106	-35.7%	**56.2**	-65.8%	80.1% TL	100% TL	80.1% TL
ghg_2veh	TL	15.5	-99.8%	**9.76**	-99.9%	0.0% TL	100% TL	100% TL
no7_ar25_1	10.1	6.83	-32.5%	**6.15**	-39.2%	100% TL	100% TL	100% TL
no7_ar3_1	6.50	6.95	6.5%	**6.13**	-5.6%	100% TL	100% TL	97.8% TL
no7_ar4_1	**65.7**	105	37.6%	76.5	14.2%	100% TL	100% TL	100% TL
no7_ar5_1	18.7	19.1	2.1%	**18.3**	-1.7%	100% TL	93.9% TL	100% TL
o7	7.41	**6.10**	-17.7%	6.51	-12.2%	100% TL	100% TL	100% TL
o7_2	6.62	6.72	1.5%	**6.48**	-2.1%	100% TL	100% TL	100% TL
o7_ar25_1	4.40	**2.53**	-42.3%	2.64	-39.9%	100% TL	100% TL	100% TL
o7_ar2_1	26.2	26.1	-0.4%	**21.6**	-17.8%	100% TL	100% TL	100% TL
o7_ar3_1	**30.3**	30.6	1.0%	30.8	1.6%	100% TL	100% TL	100% TL
o7_ar4_1	**6.98**	10.8	35.2%	10.8	35.6%	100% TL	100% TL	100% TL
o7_ar5_1	6.51	**3.83**	-41.1%	4.04	-38.0%	100% TL	100% TL	100% TL
water3	12.9	7.39	-42.9%	**7.34**	-43.2%	100% TL	91.1% TL	91.1% TL
waters	**30.7**	55.9	45.1%	39.7	22.7%	100% TL	100% TL	100% TL

	60% primal gap time (s)					Gap closed (%)		
Problem	Couenne	Cont.	Gain	All	Gain	Couenne	Cont.	All
watersbp	20.0	**11.8**	-41.0%	22.4	10.6%	100% TL	100% TL	100% TL
Mean			-10.91%		-14.25%			

Table 5: Results for the MINLP hard problems

4.2 Multiway branching

Figure 5 and Figure 6 show the effect of the δ parameter, which corresponds to the minimum width for a new node candidate to be accepted. The gains are an average over the easy and medium size problems. The 4-way partitioning is used if all the 3 new node candidates (parts 2, 3 and 4 in Figure 4) have a minimum width of δ (part 1 in Figure 4 is always kept as is). If not, the 3-way partitioning is considered with the same condition on the minimum width. If the condition is still not matched by the 3-way partitioning, the 2-way partitioning is used.

If $\delta = 0$, all the new node candidates of the 4-way partitioning are always considered wide enough. However, this does not mean that the 4-way partitioning is possible in all configurations. If h_b is close to l_b (resp. u_b), then the margin Δ_1 (resp. Δ_2) may set the branching point $h_b - \Delta_1$ (resp. $h_b + \Delta_2$) outside of the allowed range $[l_b, u_b]$, making the 4-way partitioning sometimes impossible with the margins Δ_1 and Δ_2. If the previous situation occurs or if one node candidate is rejected because its width is below $\delta > 0$, the algorithm falls back on the 3-way partitioning.

Unlike the 4-way partitioning, the 3-way partitioning is always possible because the new branching point can either be set on the left or on the right of h_b, as long as $\Delta_1, \Delta_2 \leq \frac{1-\alpha}{4}(u_b - l_b)$ and the nodes are not merged because of δ. The tightest condition on the maximum value of Δ_1 and Δ_2 occurs if $\bar{x}_b$ has its rightmost value (i.e. the distance between $\bar{x}_b$ and u_b is as small as possible) and $h_b = \frac{\bar{x}_b + u_b}{2}$. The rightmost value of $\bar{x}_b$ is obtained for $x_b = u_b$, which yieds $\bar{x}_b = \frac{1-\alpha}{2} l_b + \frac{1+\alpha}{2} u_b$. If $h_b = \frac{\bar{x}_b + u_b}{2}$, the 3-way partitioning is possible if $\Delta_1, \Delta_2 \leq \frac{u_b - \bar{x}_b}{2} = \frac{1-\alpha}{4}(u_b - l_b)$. Although the 3-way partitioning is possible at all nodes with this condition, the 2-way partitioning must be used if one node is not wide enough because of δ, if no known incumbents have their variable value x_b within $[l_b, u_b]$ or if the algorithm chooses not to use any incumbent to direct the diving at a node.

Figure 5 and Figure 6 show the node and the time gain, against the node acceptance width δ. The two plots confirm the insight that the multiway branching should be scarcely used. Creating too many nodes because of a low δ results in more small nodes and more computing time. As δ is increased, the multiway is used less and starts reducing the number of nodes explored to achieve 60% of the primal gap, while not significantly increasing the total number of nodes.

If part 2 is possible (Figure 4), it is considered too small if $h_b - \Delta_1 - \bar{x}_b < \delta$. The largest range $[\bar{x}_b, h_b - \Delta_1]$ is obtained with $x_b = l_b$ and $h_b = u_b$. The condition becomes $\frac{1+\alpha}{2}(u_b - l_b) - \Delta_1 < \delta$. If part 4 is possible (Figure 4), it is considered too small if $\frac{1+\alpha}{2}(u_b - l_b) - \Delta_2 < \delta$ with a similar reasoning. Therefore, if $\delta > \frac{1+\alpha}{2} - \min(\Delta_1, \Delta_2)$, the parts 2 and 4 are always considered too small and the algorithm performs a 2-way partitioning. Figure 7 plots the average repartition of the branching types over all the instances. With $\alpha = \frac{1}{4}, \Delta_1 = 0.07(u_b - l_b), \Delta_2 =$

$0.13(u_b - l_b)$, beyond $\delta = \frac{1+\alpha}{2} - \min(\Delta_1, \Delta_2) = 0.555(u_b - l_b)$, the 2-way partitioning (with either the guided dive or not) is always used. Slightly below this threshold, the 3-way and 4-way partitionings are also unlikely to be used because x_b and h_b would need to be very close to l_b and u_b. The key point is to use the 3-way branching scarcely. Moreover, the 4-way branching is too prohibitive and should not be used at all.

One parameter limits the depth at which IPOPT can be called, to prevent spending too much time in the NLP solver. However, by creating 4 or 3 nodes instead of only 2, IPOPT is called at more nodes overall. The sBB algorithm is therefore more likely to find new incumbents. As a consequence, the number of nodes gain is improved at the cost of wasting time. This bias would prevent a fair comparison between the 2-way and multiway branching procedures. We add a maximum number of times that IPOPT may be called. Depending on whether the 2-way, 3-way or 4-way partitionings are used at the nodes, they may lead the branch-and-bound to different scenarios, but the overall time spent in IPOPT will be roughly the same. This helps comparing the partitionings more fairly.

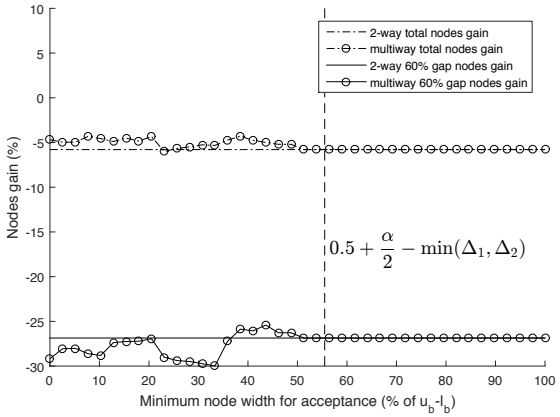

Fig. 5: Node gain for 60% primal gap and convergence with $(\Delta_1, \Delta_2) = (0.07, 0.13)$

5 Conclusion

This paper has presented an adaptation of the guided dive for solving NLP problems to global optimality in a spatial Brand-and-Bound algorithm. We first motivated the original idea of using the incumbent to direct the dive even if it does not belong to the current node. We showed which feasible solution to use if several different feasible solutions are able to direct a dive. The heuristics described were benchmarked in Couenne with a wide range of easy, medium size as well as hard problems from the Coconut library. An insight of the results of the guided dive for easy, medium and hard MINLP problems from the MINLPlib library was presented.

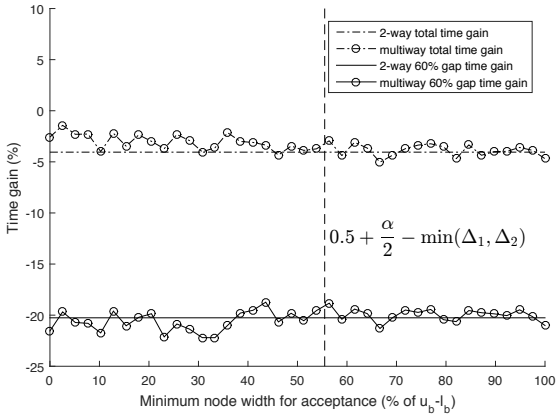

Fig. 6: Time gain for 60% primal gap and convergence with $(\Delta_1, \Delta_2) = (0.07, 0.13)$

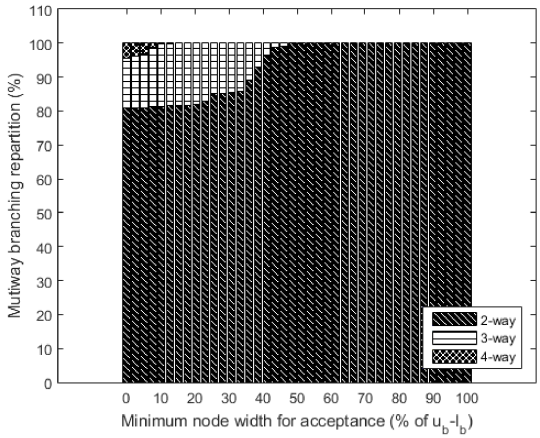

Fig. 7: Multiway branching repartition for $(\Delta_1, \Delta_2) = (0.07, 0.13)$

An adaptative multiway branching tailored for the guided dive has been described and benchmarked. While the results of the guided dive for the 2-way partitioning are very encouraging without the need to fine tune the parameters, the results for the multiway branching are more mitigated. A noticeable gain can only be achieved if the parameters are somewhat benchmarked.

There is still room for improvement in the branching process. We focused on the local branching around a known feasible solution, but all the known feasible solutions still carry more information about the whereabouts of the other feasible solutions. While most parameters in our heuristics need not be fine-tuned, one may obtain better results if they could be adapted online.

	60% primal gap time (s)							Total time (s)						
Problem	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain
alkyl	0.217	0.143	-33.87%	**0.13**	-40.01%	0.143	-33.87%	**0.23**	0.267	13.76%	0.247	6.77%	0.26	11.54%
chem	252	0.37	-99.85%	0.387	-99.85%	**0.357**	-99.86%	936	**914**	-2.32%	924	-1.28%	920	-1.68%
chenery	1.85	**1.05**	-43.52%	1.08	-41.55%	1.08	-41.90%	**7.11**	12.5	43.32%	12.7	43.85%	12.7	43.81%
ex14_1_8	**0.117**	0.153	23.87%	0.163	28.54%	0.157	25.53%	**0.12**	0.157	23.42%	0.163	26.52%	0.157	23.42%
ex14_2_8	0.41	0.44	6.82%	**0.137**	-66.66%	**0.137**	-66.66%	0.41	0.44	6.82%	**0.137**	-66.66%	**0.137**	-66.66%
ex14_2_9	0.383	0.347	-9.55%	**0.173**	-54.79%	**0.173**	-54.79%	0.383	0.347	-9.55%	**0.173**	-54.79%	**0.173**	-54.79%
ex4_1_9	0.0533	**0.0433**	-18.76%	0.0467	-12.38%	**0.0433**	-18.76%	0.09	0.0867	-3.67%	0.0867	-3.67%	**0.0833**	-7.44%
ex5_2_2_case1	0.173	**0.153**	-11.54%	0.217	20.03%	0.193	10.35%	0.223	0.203	-8.96%	0.217	-2.96%	**0.193**	-13.43%
ex5_3_2	0.647	**0.457**	-29.38%	0.463	-28.36%	0.467	-27.83%	0.647	**0.457**	-29.38%	0.463	-28.36%	0.467	-27.83%
ex5_4_2	0.847	0.73	-13.78%	**0.717**	-15.35%	0.753	-11.03%	0.847	0.73	-13.78%	**0.717**	-15.35%	0.753	-11.03%
ex5_4_4	**0.293**	0.337	12.89%	0.34	13.74%	0.347	15.40%	**9.83**	35	71.89%	43.8	77.58%	43.9	77.64%
ex6_1_1	18.9	**17.9**	-5.22%	21.5	12.45%	23.3	18.97%	106	105	-0.03%	**101**	-3.87%	104	-1.07%
ex6_1_2	0.07	0.0767	8.74%	0.0633	-9.57%	**0.0533**	-23.86%	0.177	0.18	1.83%	0.17	-3.79%	**0.15**	-15.11%
ex6_1_3	1.95	1.25	-35.96%	1.2	-38.36%	**1.18**	-39.22%	**410**	449	8.63%	420	2.33%	457	10.12%
ex6_1_4	0.453	0.21	-53.67%	**0.2**	-55.88%	0.22	-51.47%	0.627	0.653	4.07%	**0.593**	-5.33%	0.633	1.04%
ex7_2_2	**0.393**	0.43	8.53%	0.403	2.48%	0.423	7.09%	**0.393**	0.43	8.53%	0.403	2.48%	0.423	7.09%
ex7_2_4	2.47	0.317	-87.16%	0.327	-86.76%	**0.303**	-87.70%	3.16	2.92	-7.59%	2.92	-7.80%	**2.84**	-10.22%
ex7_2_7	0.143	0.113	-20.94%	**0.11**	-23.24%	0.14	-2.30%	0.193	0.2	3.35%	0.19	-1.71%	**0.18**	-6.88%
ex7_2_8	1.82	0.363	-80.04%	0.357	-80.40%	**0.34**	-81.32%	3.26	3.24	-0.61%	3.22	-1.03%	**3.19**	-2.05%
ex7_2_9	**6.94**	7.53	7.88%	13.7	49.27%	13.5	48.66%	47.3	46.1	-2.60%	13.7	-71.10%	**13.5**	-71.44%
ex7_3_4	2.76	0.673	-75.61%	0.513	-81.40%	**0.493**	-82.13%	2.78	**2.67**	-4.07%	3.11	10.41%	3	7.12%
ex7_3_5	23.2	22.4	-3.36%	22.4	-3.55%	**22.1**	-4.91%	23.2	22.4	-3.36%	22.4	-3.55%	**22.1**	-4.91%
ex8_1_7	0.173	**0.14**	-19.22%	0.153	-11.54%	0.167	-3.81%	0.26	**0.253**	-2.58%	0.263	1.25%	0.257	-1.27%
ex8_4_1	11.3	**0.447**	-96.06%	**0.447**	-96.06%	**0.447**	-96.06%	43.4	**41.5**	-4.46%	41.9	-3.50%	42.4	-2.44%
ex8_4_3	**8.43**	8.94	5.71%	8.58	1.75%	8.96	5.99%	**8.43**	8.94	5.71%	8.58	1.75%	8.96	5.99%
ex8_4_4	1.13	**0.4**	-64.60%	0.417	-63.12%	0.413	-63.42%	2.14	**2.08**	-2.65%	2.18	2.13%	2.19	2.43%
ex8_4_5	0.24	0.243	1.36%	0.22	-8.33%	**0.207**	-13.87%	6.81	7.06	3.54%	5.57	-18.30%	**5.51**	-19.18%
ex8_5_3	0.61	0.207	-66.11%	0.213	-65.03%	**0.197**	-67.75%	1.2	1.32	9.07%	1.36	11.74%	**1.19**	-0.83%
ex8_5_4	0.28	0.167	-40.46%	0.163	-41.68%	**0.153**	-45.25%	**2.18**	2.3	5.22%	2.26	3.69%	2.41	9.80%
ex8_5_5	0.177	**0.0867**	-50.93%	**0.0867**	-50.93%	0.0933	-47.20%	1.14	1.39	17.75%	**1.1**	-3.79%	1.21	5.77%
ex9_2_4	0.03	0.03	0.00%	**0.0267**	-11.00%	0.03	0.00%	0.03	0.03	0.00%	**0.0267**	-11.00%	0.03	0.00%
expfit	**0.35**	0.38	7.89%	0.353	0.93%	0.37	5.41%	**0.35**	0.38	7.89%	0.353	0.93%	0.37	5.41%
expfitb	**152**	162	6.15%	166	8.59%	165	7.83%	**152**	162	6.15%	166	8.59%	165	7.83%
fccu	**0.19**	0.197	3.41%	0.203	6.54%	0.2	5.00%	**0.19**	0.197	3.41%	0.203	6.54%	0.2	5.00%
fletcher	0.503	0.123	-75.50%	0.137	-72.84%	**0.12**	-76.16%	1.03	**0.813**	-21.29%	0.83	-19.67%	0.95	-8.06%

	60% primal gap time (s)							Total time (s)						
Problem	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain
gtm	0.33	0.327	-1.00%	**0.31**	-6.06%	0.313	-5.06%	0.33	0.327	-1.00%	**0.31**	-6.06%	0.313	-5.06%
haifas	11.4	11.1	-2.52%	10.7	-6.27%	**10.6**	-6.80%	34.3	34.3	-0.17%	**28.5**	-16.97%	46.2	25.73%
haldmads	11.6	1.5	-87.11%	1.41	-87.88%	**1.22**	-89.54%	11.6	1.5	-87.11%	1.41	-87.88%	**1.22**	-89.54%
hanging	107	106	-1.63%	**105**	-2.56%	106	-1.31%	107	106	-1.63%	**105**	-2.56%	106	-1.31%
harker	1.45	1.09	-25.06%	1.09	-24.60%	**1.03**	-28.97%	1.45	1.09	-25.06%	1.09	-24.60%	**1.03**	-28.97%
hart6	1.17	**1.08**	-7.71%	1.08	-7.43%	1.11	-5.14%	1.17	**1.08**	-7.71%	1.08	-7.43%	1.11	-5.14%
hatfldh	0.04	0.0367	-8.25%	**0.03**	-25.00%	0.0367	-8.25%	0.04	0.0367	-8.25%	**0.03**	-25.00%	0.0367	-8.25%
haverly	0.28	0.277	-1.18%	**0.257**	-8.32%	0.26	-7.14%	0.28	0.277	-1.18%	**0.257**	-8.32%	0.26	-7.14%
hhfair	7.85	1.12	-85.70%	**1.05**	-86.67%	1.07	-86.38%	**12.8**	22.3	42.50%	21.1	39.45%	21.4	40.22%
himmel16	18.1	**0.527**	-97.09%	0.533	-97.05%	0.54	-97.01%	28.8	13.1	-54.59%	12	-58.21%	**12**	-58.44%
himmelbk	**1.96**	2.2	10.93%	2.16	9.27%	2.13	7.99%	**1.96**	2.2	10.93%	2.16	9.27%	2.13	7.99%
himmelp1	0.927	0.123	-86.69%	0.13	-85.97%	**0.12**	-87.05%	**1.36**	1.39	2.16%	1.39	1.93%	1.36	0.24%
himmelp2	0.933	0.987	5.41%	0.14	-85.00%	**0.137**	-85.35%	1.53	1.56	2.34%	**1.49**	-2.62%	**1.49**	-2.62%
himmelp3	0.313	0.127	-59.56%	**0.117**	-62.75%	0.12	-61.70%	**0.313**	0.34	7.85%	0.34	7.85%	0.337	6.95%
himmelp4	0.41	0.16	-60.98%	**0.153**	-62.61%	0.16	-60.98%	**0.417**	0.423	1.56%	0.433	3.83%	0.433	3.83%
himmelp5	0.397	0.36	-9.25%	**0.34**	-14.29%	0.353	-10.94%	0.397	0.36	-9.25%	**0.34**	-14.29%	0.353	-10.94%
himmelp6	0.477	0.163	-65.74%	0.15	-68.53%	**0.147**	-69.23%	**0.477**	0.517	7.74%	0.513	7.13%	0.5	4.66%
house	0.25	**0.23**	-8.00%	0.377	33.63%	1.04	75.96%	2.71	3.12	13.25%	**2.67**	-1.48%	2.85	5.03%
hs005	0.03	0.0267	-11.00%	**0.0233**	-22.33%	**0.0233**	-22.33%	0.03	0.0267	-11.00%	**0.0233**	-22.33%	**0.0233**	-22.33%
hs010	0.663	0.493	-25.63%	**0.49**	-26.13%	0.5	-24.62%	0.663	0.493	-25.63%	**0.49**	-26.13%	0.5	-24.62%
hs012	0.0233	0.0267	12.73%	0.0233	0.00%	**0.0167**	-28.33%	0.0233	0.0333	30.03%	0.0267	12.73%	**0.02**	-14.16%
hs029	0.0633	0.237	73.26%	**0.0467**	-26.22%	**0.0467**	-26.22%	0.25	0.253	1.30%	0.247	-1.32%	**0.24**	-4.00%
hs035	0.0667	0.0467	-29.99%	**0.0433**	-35.08%	**0.0433**	-35.08%	0.177	0.183	3.60%	0.18	1.83%	**0.167**	-5.66%
hs037	0.117	0.103	-11.48%	0.0833	-28.62%	**0.08**	-31.45%	0.117	0.103	-11.48%	0.0833	-28.62%	**0.08**	-31.45%
hs038	0.23	**0.103**	-55.09%	0.113	-50.74%	0.11	-52.17%	0.463	0.5	7.34%	0.497	6.72%	**0.44**	-5.03%
hs040	**0.0567**	0.08	29.12%	0.0767	26.08%	0.07	19.00%	**0.0567**	0.08	29.12%	0.0767	26.08%	0.07	19.00%
hs041	0.06	**0.0433**	-27.83%	0.0667	10.04%	0.0533	-11.17%	0.06	**0.0433**	-27.83%	0.0667	10.04%	0.0533	-11.17%
hs054	0.0433	0.0333	-23.09%	**0.03**	-30.72%	0.0333	-23.09%	0.0433	0.0333	-23.09%	**0.03**	-30.72%	0.0333	-23.09%
hs057	6.85	2.42	-64.64%	**2.39**	-65.17%	**2.39**	-65.17%	6.85	2.43	-64.59%	2.39	-65.13%	**2.39**	-65.17%
hs060	**0.0333**	0.04	16.75%	**0.0333**	0.00%	0.0367	9.26%	**0.0333**	0.04	16.75%	**0.0333**	0.00%	0.0367	9.26%
hs062	0.477	0.0833	-82.53%	**0.0733**	-84.62%	0.1	-79.02%	**0.973**	1.01	3.95%	1.09	10.44%	1.15	15.37%
hs063	0.05	0.05	0.00%	**0.0433**	-13.40%	0.0467	-6.60%	0.05	0.05	0.00%	**0.0433**	-13.40%	0.0467	-6.60%
hs071	0.197	**0.17**	-13.57%	0.19	-3.41%	0.187	-5.08%	0.197	**0.17**	-13.57%	0.19	-3.41%	0.187	-5.08%
hs073	**0.04**	**0.04**	0.00%	**0.04**	0.00%	**0.04**	0.00%	**0.04**	**0.04**	0.00%	**0.04**	0.00%	**0.04**	0.00%
hs074	3.68	3.44	-6.52%	**1.19**	-67.69%	1.23	-66.52%	18.7	17.9	-4.05%	8.74	-53.20%	**8.43**	-54.88%

	60% primal gap time (s)							Total time (s)						
Problem	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain
hs075	0.1	0.0867	-13.30%	**0.0833**	-16.70%	0.09	-10.00%	0.1	0.0867	-13.30%	**0.0833**	-16.70%	0.09	-10.00%
hs077	0.18	0.18	0.00%	0.183	1.80%	**0.167**	-7.39%	0.18	0.18	0.00%	0.183	1.80%	**0.167**	-7.39%
hs078	0.3	0.177	-41.10%	0.18	-40.00%	**0.163**	-45.57%	**1.57**	1.69	7.10%	1.63	3.68%	1.62	3.09%
hs079	0.123	0.113	-8.11%	0.11	-10.79%	**0.1**	-18.90%	0.123	0.113	-8.11%	0.11	-10.79%	**0.1**	-18.90%
hs080	1.47	**0.35**	-76.24%	0.923	-37.33%	0.96	-34.84%	1.47	**0.35**	-76.24%	0.923	-37.33%	0.96	-34.84%
hs081	1.02	1	-1.64%	0.923	-9.19%	**0.857**	-15.74%	1.02	1	-1.64%	0.923	-9.19%	**0.857**	-15.74%
hs086	0.38	0.2	-47.37%	0.21	-44.74%	**0.197**	-48.24%	0.67	0.6	-10.45%	0.607	-9.45%	**0.553**	-17.42%
hs099	5.59	**4.45**	-20.29%	638	99.12%	633	99.12%	646	640	-0.83%	638	-1.22%	**633**	-1.92%
hs100	**0.447**	0.533	16.24%	0.523	14.64%	0.53	15.72%	**0.503**	0.533	5.63%	0.523	3.82%	0.53	5.04%
hs100mod	0.31	0.35	11.43%	0.31	0.00%	**0.183**	-40.87%	0.347	0.35	0.94%	**0.31**	-10.59%	0.323	-6.75%
hs104	3	0.34	-88.65%	0.343	-88.54%	**0.333**	-88.88%	4.14	3.63	-12.17%	**3.6**	-13.05%	3.66	-11.52%
hs106	1.39	**1.22**	-11.99%	2.89	51.90%	2.85	51.17%	2.97	3.01	1.33%	2.89	-2.69%	**2.85**	-4.15%
hs108	2.66	22.6	88.26%	23.2	88.55%	**0.417**	-84.32%	**671**	771	12.98%	724	7.38%	739	9.27%
hs113	0.577	0.587	1.70%	**0.49**	-15.03%	0.51	-11.57%	0.577	0.587	1.70%	**0.49**	-15.03%	0.51	-11.57%
hs116	0.383	0.387	0.88%	0.377	-1.72%	**0.36**	-6.08%	0.6	0.583	-2.78%	0.587	-2.22%	**0.57**	-5.00%
hs119	6.5	**0.8**	-87.70%	0.817	-87.44%	0.81	-87.54%	**8.15**	8.71	6.43%	8.69	6.14%	8.68	6.03%
hs35mod	**0.0567**	**0.0567**	0.00%	0.0733	22.65%	0.0633	10.43%	0.07	0.07	0.00%	0.0733	4.50%	**0.0633**	-9.57%
launch	1.34	1.35	0.74%	1.3	-2.74%	**1.28**	-4.23%	1.97	**1.97**	-0.17%	2.29	13.83%	2.24	12.04%
meanvar	0.92	0.153	-83.34%	0.15	-83.70%	**0.147**	-84.05%	0.937	0.903	-3.57%	0.893	-4.63%	**0.873**	-6.77%
mhw4d	0.13	0.117	-10.23%	0.127	-2.54%	**0.113**	-12.85%	**0.203**	0.223	8.96%	0.24	15.29%	0.21	3.19%
minlphi	0.833	0.82	-1.60%	**0.797**	-4.39%	0.81	-2.80%	1.06	1.08	1.55%	**1.06**	-0.31%	1.07	0.63%
mwright	0.0633	0.05	-21.01%	**0.0367**	-42.02%	0.0467	-26.22%	0.197	0.147	-25.42%	**0.14**	-28.83%	0.153	-22.06%
obstclal	7.61	**7.18**	-5.65%	7.44	-2.32%	7.56	-0.70%	7.61	**7.18**	-5.65%	7.44	-2.32%	7.56	-0.70%
palmer2	**0.373**	39.6	99.06%	60.8	99.39%	31.2	98.80%	**0.377**	39.6	99.05%	60.8	99.38%	31.2	98.79%
palmer3c	**0.06**	**0.06**	0.00%	**0.06**	0.00%	**0.06**	0.00%	**0.06**	**0.06**	0.00%	**0.06**	0.00%	0.0633	5.21%
palmer5d	**0.05**	0.0567	11.82%	0.06	16.67%	0.06	16.67%	**0.05**	0.0567	11.82%	0.06	16.67%	0.06	16.67%
palmer8c	0.0867	0.09	3.67%	**0.0667**	-23.07%	**0.0667**	-23.07%	0.0867	0.09	3.67%	**0.0667**	-23.07%	**0.0667**	-23.07%
polak3	2.59	**2.58**	-0.39%	2.6	0.26%	2.71	4.42%	2.59	**2.58**	-0.39%	2.6	0.26%	2.71	4.42%
process	37.1	26.9	-27.37%	0.13	-99.65%	**0.127**	-99.66%	1135	346	-69.26%	**53.9**	-95.21%	54.6	-95.15%
prodpl0	0.21	**0.203**	-3.19%	0.207	-1.57%	0.217	3.09%	0.21	**0.203**	-3.19%	0.207	-1.57%	0.217	3.09%
prodpl1	0.37	**0.307**	-17.11%	0.313	-15.32%	**0.307**	-17.11%	0.463	**0.453**	-2.16%	0.47	1.43%	0.463	0.00%
qcqp2_4478	0.0233	0.02	-14.16%	**0.0167**	-28.33%	0.0233	0.00%	0.0233	0.02	-14.16%	**0.0167**	-28.33%	0.0233	0.00%
qcqp2_7212	0.06	0.07	14.29%	0.06	0.00%	**0.0567**	-5.50%	0.06	0.07	14.29%	0.06	0.00%	**0.0567**	-5.50%
qcqp2_7442	**0.09**	0.0967	6.93%	0.0967	6.93%	0.0933	3.54%	**0.0933**	0.0967	3.52%	0.0967	3.52%	**0.0933**	0.00%
qpcboei2	6.33	6.35	0.37%	6.08	-3.90%	**6.04**	-4.53%	7.25	7.29	0.50%	6.08	-16.09%	**6.04**	-16.64%

	60% primal gap time (s)							Total time (s)						
Problem	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain	Couenne	Test 2	Gain	Test 6	Gain	Test 7	Gain
qpnboei2	11.2	**9.72**	-13.50%	12.8	12.13%	12.8	11.94%	20.4	17.4	-14.89%	12.8	-37.34%	**12.8**	-37.47%
rk23	**1.11**	3.16	64.81%	3.16	64.81%	3.15	64.66%	**1.12**	3.18	64.85%	3.17	64.81%	3.16	64.66%
robot	113	108	-4.31%	0.413	-99.63%	**0.403**	-99.64%	212	214	0.70%	**170**	-20.00%	192	-9.70%
sambal	**0.0933**	**0.0933**	0.00%	**0.0933**	0.00%	0.0967	3.52%	**0.0933**	**0.0933**	0.00%	**0.0933**	0.00%	0.0967	3.52%
sisser	0.383	0.373	-2.61%	0.367	-4.33%	**0.347**	-9.55%	0.383	0.373	-2.61%	0.367	-4.33%	**0.347**	-9.55%
spanhyd	**4.5**	4.52	0.52%	5.48	17.99%	5.49	18.04%	**4.5**	4.52	0.52%	5.48	17.99%	5.49	18.04%
tointqor	1.66	1.7	2.54%	**1.62**	-2.41%	1.69	1.58%	1.66	1.7	2.54%	**1.62**	-2.41%	1.69	1.58%
trimloss	**2.94**	2.98	1.23%	2.99	1.67%	2.96	0.56%	**2.94**	2.98	1.23%	2.99	1.67%	2.96	0.56%
twobars	0.04	0.04	0.00%	0.0367	-8.25%	**0.0333**	-16.75%	0.0467	0.0433	-7.28%	0.04	-14.35%	**0.0367**	-21.41%
wall	3.98	**3.86**	-2.93%	3.87	-2.85%	3.97	-0.33%	3.98	**3.86**	-2.93%	3.87	-2.85%	3.97	-0.33%
zecevic4	0.03	0.03	0.00%	**0.0233**	-22.33%	0.0333	9.91%	0.03	0.03	0.00%	**0.0233**	-22.33%	0.0333	9.91%
zy2	**0.06**	0.0733	18.14%	0.0633	5.21%	**0.06**	0.00%	**0.06**	0.0733	18.14%	0.0633	5.21%	**0.06**	0.00%

Table 6: Results for the easy and medium size problems

Appendix

See Table 6.

References

1. P. Abichandani, H. Y. Benson, and M. Kam. Multi-vehicle path coordination under communication constraints. In *Proceedings of the American Control Conference*, pages 650–656, 2008.
2. C. Adjiman, S. Dallwig, C. Floudas, and A. Neumaier. A global optimization method, alpha bb, for general twice-differentiable constrained NLPs - I. theoretical advances. *Computers & Chemical Engineering*, 22:1137–1158, 1998.
3. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.
4. T. Berthold. Rens - relaxation enforced neighborhood search. Technical Report 07-28, ZIB, Takustr.7, 14195 Berlin, 2007.
5. T. Berthold, G. Gamrath, A. M. Gleixner, S. Heinz, T. Koch, and Y. Shinano. Solving mixed integer linear and nonlinear problems using the SCIP optimization suite. Technical Report 12-27, ZIB, 2012.
6. L. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics, 2010.
7. L. T. Biegler. *System Modeling and Optimization: 23rd IFIP TC 7 Conference, Cracow, Poland, July 23-27, 2007, Revised Selected Papers*, chapter Efficient Nonlinear Programming Algorithms for Chemical Process Control and Operations, pages 21–35. Springer Berlin Heidelberg, 2009.
8. P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1 1995.
9. R. H. Byrd, R. B. Schnabel, and G. A. Shultz. A trust region algorithm for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis*, 24(5):1152–1170, 1987.
10. T. T. Chung and T. C. Sun. Weight optimization for flexural reinforced concrete beams with static nonlinear response. *Structural optimization*, 8(2):174–180, 1994.
11. E. Costa-Montenegro, F. J. González-Castaño, P. S. Rodríguez-Hernández, and J. C. Burguillo-Rial. *Computational Science – ICCS 2007: 7th International Conference, Beijing, China, May 27 - 30, 2007, Proceedings, Part IV*, chapter Nonlinear Optimization of IEEE 802.11 Mesh Networks, pages 466–473. Springer Berlin Heidelberg, 2007.
12. E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 2005.
13. V. Donde, V. Lopez, B. Lesieutre, A. Pinar, C. Yang, and J. Meza. Identification of severe multiple contingencies in electric power networks. In *Power Symposium, 2005. Proceedings of the 37th Annual North American*, pages 59–66, 2005.
14. A. V. Fiacco and G. P. McCormick. *Nonlinear programming; sequential unconstrained minimization techniques*. Wiley New York, 1968.
15. C. Fielding. *Advanced techniques for clearance of flight control laws*, volume 283 of *Engineering online library*. Springer, 2002.
16. M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1):23–47, 2003.
17. R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.
18. A. Fügenschuh, M. Herty, A. Klar, and A. Martin. Combinatorial and continuous models for the optimization of traffic flows on networks. *SIAM Journal on Optimization*, 16(4):1155–1176, 2006.
19. E. P. Gatzke, J. E. Tolsma, and P. I. Barton. Construction of convex relaxations using automated code generation techniques. *Optimization and Engineering*, 3(3):305–326, 2002.
20. B. H. Gebreslassie, M. Slivinsky, B. Wang, and F. You. Life cycle optimization for sustainable design and operations of hydrocarbon biorefinery via fast pyrolysis, hydrotreating and hydrocracking. *Computers & Chemical Engineering*, 50:71 – 91, 2013.
21. I. Gentilini, F. Margot, and K. Shimada. The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software*, 28(2):364–378, 2013.

22. M. Isshiki, D. Sinclair, and S. Kaneko. Lens design: Global optimization of both performance and tolerance sensitivity. In *International Optical Design*. Optical Society of America, 2006.
23. J. R. Jackson, J. Hofmann, J. Wassick, and I. E. Grossmann. A nonlinear multiperiod process optimization model for production planning in multi-plant facilities. In *Proceedings FOCAPO 2003*, pages 281–284, 2003.
24. R. Karuppiah and I. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers & Chemical Engineering*, 30:650–673, 2006.
25. L. Liberti, G. Nannicini, and N. Mladenović. *A Good Recipe for Solving MINLPs*, pages 231–244. Springer US, Boston, MA, 2010.
26. Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods Software*, 24(4-5):657–668, 2009.
27. R. Misener and C. A. Floudas. Antigone: Algorithms for continuous / integer global optimization of nonlinear equations. *J. of Global Optimization*, 59(2-3):503–526, July 2014.
28. N. Mladenovic and P. Hanse. Variable neighborhood search. *Computers and Operations Research*.
29. J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic. Challenges to optimal power flow. *Power Systems, IEEE Transactions on*, 12(1):444–455, 1997.
30. G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex minlps. *Mathematical Programming Computation*, 4(1):1–31, 2012.
31. G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. *ArXiv e-prints*, 2008.
32. Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*. SIAM studies in applied mathematics. Society for Industrial and Applied Mathematics, 1994.
33. A. Neumaier. Complete search in continuous global optimization and constraint satisfaction, acta numerica 13, 2004.
34. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.
35. M. E. Pfetsch, A. Fügenschuh, B. Geißler, N. Geißler, R. Gollmer, B. Hiller, J. Humpola, T. Koch, T. Lehmann, A. Martin, A. Morsi, J. Rövekamp, L. Schewe, M. Schmidt, R. Schultz, R. Schwarz, J. Schweiger, C. Stangl, M. C. Steinbach, S. Vigerske, and B. M. Willert. Validation of nominations in gas network optimization: Models, methods, and solutions. *Optimization Methods and Software*, 2014.
36. A. Quist, R. van Geemert, J. Hoogenboom, T. Illés, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26(5):423 – 448, 1999.
37. A. U. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad. Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft. *Journal of guidance, control, and dynamics*, 27(4):586–594, 2004.
38. E. Smith and C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23(4-5):457 – 478, 1999.
39. M. Soleimanipour, W. Zhuang, and G. Freeman. Optimal resource management in wireless multimedia wideband CDMA systems. *Mobile Computing, IEEE Transactions on*, 1(2):143–160, 2002.
40. M. Tawarmalani and N. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Nonconvex Optimization and Its Applications. Springer US, 2002.
41. M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.