Journal of Heuristics (2021) 27:615–648
https://doi.org/10.1007/s10732-021-09467-z

Enhancing large neighbourhood search heuristics for Benders' decomposition

Stephen J. Maher[1,2]

Received: 1 July 2019 / Revised: 24 August 2020 / Accepted: 8 January 2021 /
Published online: 23 February 2021

Abstract
A general enhancement of the Benders' decomposition (BD) algorithm can be achieved through the improved use of large neighbourhood search heuristics within mixed-integer programming solvers. While mixed-integer programming solvers are endowed with an array of large neighbourhood search heuristics, few, if any, have been designed for BD. Further, typically the use of large neighbourhood search heuristics is limited to finding solutions to the BD master problem. Given the lack of general frameworks for BD, only ad hoc approaches have been developed to enhance the ability of BD to find high quality primal feasible solutions through the use of large neighbourhood search heuristics. The general BD framework of SCIP has been extended with a trust region based heuristic and a general enhancement for large neighbourhood search heuristics. The general enhancement employs BD to solve the auxiliary problems of all large neighbourhood search heuristics to improve the quality of the identified solutions. The computational results demonstrate that the trust region heuristic and a general large neighbourhood search enhancement technique accelerate the improvement in the primal bound when applying BD.

Keywords Benders' decomposition · Large neighbourhood search · Enhancement techniques · Mixed integer programming

1 Introduction

Benders' decomposition (BD) (Benders 1962) is a popular mathematical programming technique that is used to solve large-scale optimisation problems. Such problems typically arise from industrial contexts, for example airline planning (Cordeau et al. 2001; Mercier et al. 2005; Papadakos 2009), maintenance scheduling (Canto 2008),

✉ Stephen J. Maher
s.j.maher@exeter.ac.uk

1 Department of Management Science, Lancaster University, Lancaster, UK

2 College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK

production planning (Oliveira et al. 2014), or network design (Costa 2005), or in the form of stochastic programming problems (Boland et al. 2016; Santoso et al. 2005; Oliveira et al. 2014; Bodur et al. 2017). While BD is an effective exact solution algorithm for large-scale optimisation problems, there are many situations where finding high-quality feasible solutions can be challenging. In such situations, BD is a valuable technique to exploit problem separability in the design of effective primal heuristic methods.

The effectiveness of BD as a heuristic algorithm relies on algorithm design and the availability of enhancement techniques. Additionally, the underlying mixed integer programming (MIP) solver and the included heuristics are critical to the ability of the BD algorithm in finding high-quality feasible solutions. This is particularly important when implementing BD using the branch-and-cut approach, i.e. using callback functions available within a MIP solver, where more interaction with the solver is supported. While the branch-and-cut approach to BD provides more flexibility in the algorithm implementation, design decisions can significantly affect the algorithmic performance. The appropriate enhancement techniques must be selected to achieve the best performance when using BD as a heuristic algorithm.

Large neighbourhood search (LNS) heuristics (Ahuja et al. 2002) are a valuable tool for finding good quality solutions for difficult MIPs. Previous work has demonstrated their potential for enhancing the BD algorithm (Rei et al. 2009; Boland et al. 2016). While MIP solvers are endowed with many LNS heuristics, their effectiveness when used within decomposition algorithms—such as Dantzig-Wolfe reformulation (Dantzig and Wolfe 1960) or BD–has not been fully realised. This is primarily due to the fact that the solution algorithms associated with decomposition techniques typically use MIP solvers as black boxes. As a result, the master and subproblems arising from the application of decomposition techniques are each solved separately. Thus, primal heuristics are applied without any knowledge of the original problem. This has a negative effect on the quality of the solutions found and, hence, the heuristic performance of the BD algorithm. Further, very few, if any, of the many primal heuristics within MIP solvers have been designed with a consideration of decomposition techniques.

This paper proposes two matheuristics to enhance the BD algorithm and improve the ability to find primal feasible solutions. The matheuristics exploit the combined power of BD and LNS heuristics to develop general enhancements for BD. The developed heuristic methods are embedded within the BD framework available in the MIP solver SCIP (Gleixner et al. 2018; Maher 2021), to support all applications of this popular algorithm. While BD is an exact solution algorithm for MIP problems, this paper takes a particular focus on its use as a metaheuristic. The computational results will demonstrate the potential of the proposed matheuristics to find high-quality solutions for difficult large-scale optimisation problems.

The contributions of this paper are:

– The formal presentation of a general enhancement technique derived from the improved use of LNS heuristics within the BD algorithm—the Large neighbourhood Benders' search (LNBS).

– The development of a trust region (TR) heuristic to enhance the BD algorithm by complementing the available heuristics within SCIP.
– A computational study evaluating the improved heuristic performance of BD through the use of the LNBS and TR heuristic.

This paper is structured as follows: An overview of SCIP, LNS heuristics and their interaction with BD will be presented in Sect. 2. Previous work on LNS heuristics and trust region methods for BD will be discussed. Section 3 will present the general enhancement approach of the LNBS and a trust region heuristic for use within a branch-and-cut implementation of BD. The implementation of the LNBS and the TR heuristic within the solver SCIP will be described in Sect. 4. A comprehensive computational study of the proposed methods will be presented in Sect. 5. Finally, concluding remarks and possible directions for future work will be given in Sect. 6.

2 Background

A core focus of this paper is the development of novel LNS heuristics to enhance the BD framework available within SCIP. To aid the exposition of this paper, a short background to SCIP and the BD framework is provided in Sect. 2.1. This will guide the discussion related to the relationship between LNS heuristics and BD in Sects. 2.2 and 2.3.

2.1 Benders' decomposition framework in SCIP

SCIP is a general purpose branch-and-cut framework that has been developed with a plugin-based design. This design enables the simple extension of the framework with specialised algorithms to improve the effectiveness of the solver. Plugins exists for the implementation of various mathematical and constraint programming algorithms, including primal heuristics, separation routines, constraint handlers and branching rules. The most relevant plugin types for this paper are constraint handlers and primal heuristics.

A critical feature of general purpose solvers is the effective coordination of various sophisticated component algorithms within a branch-and-cut framework. The solving process of SCIP, demonstrating the coordination of various features, is presented in Fig. 1. The relevant features of the solving process are highlighted in orange, in particular the primal heuristics and the BD constraint handlers. The BD constraint handlers were added as part of the BD framework that was released in SCIP 6.0 (Gleixner et al. 2018). For more details about the BD framework and constraint handlers in SCIP, the reader is referred to Maher (2021).

Within the node processing stage, the BENDERSDECOMPLP and BENDERSDECOMP constraints supply fractional and integral LP solutions, respectively, to the BD subproblems for the generation of Benders' cuts. Primal heuristics are executed within the solving loop to identify primal feasible solutions and improve the global upper bound. After the execution of the primal heuristics the BENDERSDECOMP constraint supplies a candidate primal feasible solution to the BD subproblems to evaluate its feasibility

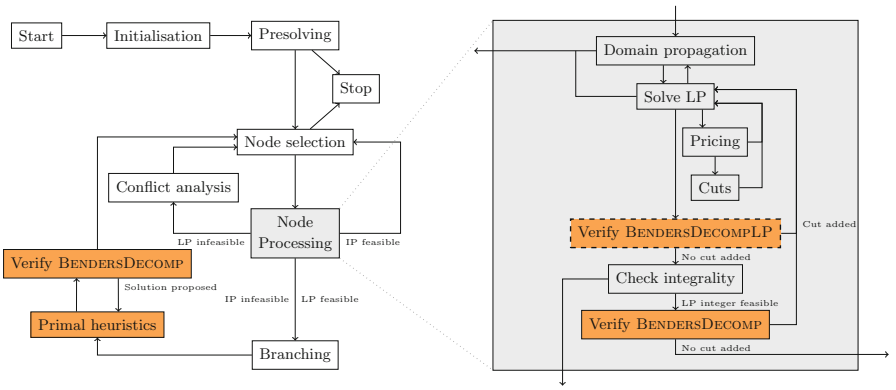

Fig. 1 The solving process of SCIP including the Benders' decomposition framework

or optimality. An important class of primal heuristics within general purpose MIP solvers are LNS heuristics.

2.2 Large neighbourhood search and Benders' decomposition

LNS heuristics aim to find improving solutions for a general MIP instance by exploring a restricted set of primal feasible solutions. There are two fundamental aspects of LNS heuristics, the definition of the neighbourhood that is searched for improving solutions and searching this neighbourhood by solving a sub-MIP, described as the auxiliary problem of the LNS heuristic. The neighbourhood is defined by fixing variables or adding constraints to the auxiliary problem that will hopefully form a problem that is easier to solve than the original MIP. The auxiliary problem is then solved over the restricted feasible region. The objective function of such an auxiliary problem may also be modified to aid the search for improving solutions.

Consider the MIP given by:

$$\min_x \{c^\top x \mid Ax \leq b, x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}. \tag{1}$$

The set of primal feasible solution for (1) is given by $\mathcal{I} := \{x \mid Ax \leq b, x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}$. Now consider a polyhedron given by $\mathcal{N}_x$ that defines the neighbourhood for a given LNS heuristic. The auxiliary problem of the LNS heuristic is given by

$$\min_x \{f(x) \mid x \in \mathcal{I} \cap \mathcal{N}_x\}, \tag{2}$$

where $f(x)$ is a general function, which need not be linear, that is used to guide the search.

Ideally the design of an LNS heuristic will achieve two main goals: containing high quality solutions in $\mathcal{I} \cap \mathcal{N}_x$ and finding such solutions is not "too difficult". Attempts to achieve these two goals have lead to the development of a number of LNS heuristics. Examples within SCIP are *Crossover* (Rothberg 2007), *DINS* (Ghosh 2007), *Local*

branching (Fischetti and Lodi 2003), *Proximity search* (Fischetti and Monaci 2014) and *RINS* (Danna et al. 2005). For further details about LNS heuristics and those implemented within SCIP, the reader is referred to the PhD thesis of Berthold (2014).

2.3 Interaction between LNS heuristics and Benders' decomposition

While LNS heuristics are a valuable component of MIP solvers, there has been little focus on their interaction with BD. To discuss the use of LNS heuristics with BD, consider the following classical example of a decomposable problem:

$$\min_{x,y}\{c^\top x + d^\top y \mid Ax \leq b,\, Bx + Hy \leq h,\, x \in \mathbb{Z}^p \times \mathbb{R}^{n-p},\, y \in \mathbb{R}^q\}. \tag{3}$$

Problem (3) is commonly described as a two-stage problem, with the first and second stage variables given by x and y respectively. BD exploits the property that for a fixed solution $\hat{x}$, the first and second stages become separable.

The application of BD results in the formation of a subproblem, given by

$$z(\hat{x}) = \min_{y}\{d^\top y \mid Hy \leq h - B\hat{x},\, y \in \mathbb{R}^q\}. \tag{4}$$

An important observation from (4) is that the feasible region is parameterised by the variables x. More importantly, the dual feasible region of (4), which is given by $\mathcal{U} := \{u \mid uH = d, u \leq 0, u \in \mathbb{R}^m\}$, does not depend on the value of x. The BD solution algorithm uses the extreme points and rays from $\mathcal{U}$, denoted by $\mathcal{P}$ and $\mathcal{R}$ respectively, to generate cuts that are added to the master problem.

A reformulation of the original problem feasible region using the extreme points and rays of $\mathcal{U}$ is given by

$$\begin{aligned} & Ax \leq b, \\ & 0 \geq u^\top(h - Bx) \quad \forall u \in \mathcal{R}, \\ & \varphi \geq u^\top(h - Bx) \quad \forall u \in \mathcal{P}, \\ & \varphi \geq \underline{\varphi}, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \\ & \varphi \in \mathbb{R}, \end{aligned} \tag{5}$$

where φ is an auxiliary variable that is added as an underestimator of the subproblem objective function value. We assume that a lower bound on φ can be found, denoted by $\underline{\varphi}$, and imposed in the master problem as a constraint. The set of solutions satisfying (5) is denoted by $\mathcal{I}$. Since the sets $\mathcal{P}$ and $\mathcal{R}$ induce a exponential number of constraints, the BD algorithm initially relaxes the feasible region of (5) by replacing $\mathcal{P}$ and $\mathcal{R}$ with $\bar{\mathcal{P}}$ and $\bar{\mathcal{R}}$, respectively, which can both be empty. The resulting feasible region is denoted by $\bar{\mathcal{I}}$. It can be observed that $\mathcal{I} \subseteq \bar{\mathcal{I}}$, a property that will be used later in this paper.

Exploiting the separability of (3) to form the subproblem leads relaxation of the original problem—by removing the second stage variables and constraints. As a result, a master problem is created, given by

$$\min_{x,\varphi}\{c^\top x + \varphi \mid (x, \varphi) \in \bar{\mathcal{I}}\}, \tag{6}$$

Throughout this paper, problems (6) and (4) are labelled the BD master (MP) and subproblem (SP) respectively. Given a solution $(\hat{x}, \hat{\varphi})$ to MP, its feasibility and optimality with respect to the second stage constraints is verified by solving SP. In the case that $\hat{x}$ induces an infeasible instance of SP, then the unbounded dual ray $u \in \mathcal{R}$ is used to construct the feasibility cut $u^\top(h - Bx) \leq 0$ and u is appended to $\bar{\mathcal{R}}$. Alternatively, if $\hat{x}$ induces a feasible instance of SP, then an optimal dual extreme point $u \in \mathcal{P}$ is used to form an optimality cut $u^\top(h - Bx) \leq \varphi$. The dual extreme point u is appended to $\bar{\mathcal{P}}$ if $z(\hat{x}) > \hat{\varphi}$. If no feasibility or optimality cut is added to the master problem, then $(\hat{x}, \hat{\varphi})$ is an optimal solution to (6).

Now, consider the MP at an intermediate stage during the solution algorithm. Since $\mathcal{I} \subseteq \bar{\mathcal{I}}$, the LNS auxiliary problem (2) searches for improving solutions in a feasible region that is a neighbourhood within a relaxation of the original problem. As such, there is no guarantee on the feasibility of solutions with respect to the original problem. Also, solving the auxiliary problem (2) may find a solution that improves upon the current best known solution from $\bar{\mathcal{I}}$; however, the solution $\hat{x}$ may not improve upon the best known solution from $\mathcal{I}$.

The LNBS, described in Sect. 3.1, addresses the issues of feasibility and solution quality by solving the auxiliary problem by BD. The proposed approach solves the auxiliary problem by implicitly considering all second stage constraints. Thus, all solutions found by the auxiliary problem are contained in $\mathcal{I}$, as opposed to $\bar{\mathcal{I}}$. This leads to higher quality solutions found by LNS heuristics when employing BD.

2.4 Related work

Various approaches have been proposed to enhance the BD algorithm through the use of LNS heuristics. These approaches have been in the form of bespoke algorithms, and there has been no attempt made to systematically integrate LNS heuristics and BD. The first example of a bespoke approach integrating LNS heuristics and BD, in the form of an enhancement technique, is the use of Local Branching (Fischetti and Lodi 2003) to accelerate BD by Rei et al. (2009). The BD algorithm is augmented by phases of Local Branching in an effort to improve the incumbent solution. It is shown by Rei et al. (2009) that employing Local Branching significantly improves the convergence of the BD algorithm.

More recently, the Proximity Search heuristic developed by Fischetti and Monaci (2014) has been applied in a Benders' decomposition context by Boland et al. (2016). Similar to Proximity Search (Fischetti and Monaci 2014), each iteration of Proximity Benders' (Boland et al. 2016) sets an upper bound relative to the incumbent solution in an attempt to find an improving solution. A proximity function, such as the Hamming distance, replaces the objective function to focus the search around the current

incumbent solution. The results presented by Boland et al. (2016) demonstrate that employing BD within the Proximity Search algorithm aids in improving the primal bound more quickly than by the default BD algorithm.

This paper proposes an alternative integration of LNS heuristics and BD compared to the custom-built methods presented above. A major contribution this work is the generalisation of the integration of BD and LNS heuristics—going beyond the custom integration methods of Rei et al. (2009) and Boland et al. (2016).

The use of LNS heuristics to enhance BD shares many similarities with trust region methods (Conn et al. 2000; Yuan 2015). Briefly, trust region methods are a numerical optimisation approach employing an iterative algorithm that starts from an incumbent solution and progressively steps towards the optimum. In each step, the incumbent solution is updated by finding a trial step from the solution of a trust region subproblem, which is an optimisation problem that can be defined over a neighbourhood of the original feasible region. The algorithm terminates when convergence limits are reached. For further details regarding trust region methods, the reader is referred to Conn et al. (2000).

Trust region methods have been previously applied with BD to aid the convergence of the algorithm and reduce the solving time of the master problem (Ruszczyński 1986; Santoso et al. 2005; Maher et al. 2014). The use of these techniques is motivated by the observed large distances between master problem solutions from consecutive iterations of the BD algorithm. Using a trust region subproblem—solved by BD—to find the next trial step reduces the distance between master problem solutions and focuses the generated cuts on a restricted feasible region.

There have been many examples of using trust region techniques with the BD algorithm. One of the earliest examples is the regularisation decomposition of Ruszczyński (1986) that involves the addition of a quadratic regularisation term to the objective function. While effective, the inclusion of the quadratic terms results in the addition of a large number of variables and constraints when linearisation techniques are applied. A trust region method is employed by Linderoth and Wright (Linderoth and Wright 2003) for a parallel implementation of the L-Shaped method, which is a BD algorithm for two-stage stochastic programs with continuous recourse. Santoso et al. (2005) propose a trust region method that involves the addition of a linear constraint in the master problem to impose a limit on the number of changes in solution between consecutive algorithm iterations. Alternatively, Maher et al. (2014) minimise the Hamming distance between the previous and current master problem solutions while imposing an upper bound on the objective function value. Building upon the successes of previous trust region approaches, the TR heuristic is proposed in Sect. 3.2 to improve the heuristic performance of BD.

Finally, the main premise of this work is to improve the ability of the BD algorithm to find high-quality primal feasible solutions. The combination of LNS heuristics and trust region methods with BD is characteristic of metaheuristic (Raidl 2015) and matheuristic (Fischetti and Fischetti 2018; Boschetti et al. 2009) approaches. In particular, Raidl (2015) discuss the use of decomposition techniques in the development of hybrid metaheuristics. A core idea of Raidl (2015) is the integration of metaheuristics and exact algorithms—building upon previous work from Puchinger et al. (2005). The integrative combinations discussed by Puchinger et al. (2005) lays the foundation for

the concepts of inner and outer algorithms presented in this paper. These concepts will be developed in the follow section with a specific focus on LNS heuristics and BD.

3 Improving heuristic performance of BD

The following methods are proposed to enhance the BD algorithm through an improved execution of LNS heuristics. To highlight the significance of the proposed methods, the concept of *inner* and *outer* algorithms are introduced. A inner algorithm is a method that supports a main algorithm, sometimes described as a sub-module. As an example, a pure branch-and-bound algorithm can solve MIPs to optimality; however, this algorithm is significantly more effective when augmented by the inner algorithms, or sub-modules, of cutting plane methods or primal heuristics. Similarly, a textbook BD algorithm can be enhanced with the inner algorithm of the Magnanti and Wong (1981) cut strengthening technique.

Alternatively, an outer algorithm has a more dominant role in the solution process by wrapping around the main algorithm. For example, local branching is an outer algorithm that is used as an algorithmic framework to guide the BD algorithm in the work by Rei et al. (2009). Also, the trust region approaches have been developed as outer algorithms that restrict the search neighbourhood of the BD algorithm. Inner and outer algorithms have vastly different implementation approaches and potentially different computational effectiveness.

The previous work on enhancements for BD that exploit heuristic or trust region methods presented above are outer algorithms for BD. For example, Proximity Benders' (Boland et al. 2016) employs Proximity Search as the main algorithm and BD is used as the solution approach for the auxiliary problem. Similarly, in previous trust region approaches (Ruszczyński 1986; Linderoth and Wright 2003; Santoso et al. 2005; Maher et al. 2014) the trust region algorithm is the outer algorithm that imposes a restriction on the master problem and then BD is the inner algorithm that solves the resulting restricted problem.

The following methods described in Sects. 3.1 and 3.2 are designed to fit within the branch-and-cut implementation of BD. This implementation treats BD as an inner algorithm of the branch-and-cut algorithm of the underlying MIP solver; however, BD has a dominate role in proving optimality and feasibility of solutions. Since BD is a dominate algorithm in the branch-and-cut implementation, all of the inner algorithms of a MIP solver can be viewed as inner algorithms for BD. The LNBS, presented in Sect. 3.1, extends previous work on outer trust region and heuristic algorithms by enhancing all available inner LNS heuristic methods. Further, the TR heuristic, described in Sect. 3.2, is an inner LNS heuristic variant of trust region methods and Proximity Search.

3.1 Large neighbourhood Benders' search

Consider MP at an intermediate stage in the BD solution algorithm, as given in Sect. 2.3. The feasible region of the corresponding MP is denoted by $\bar{\mathcal{I}}$. The LNS

heuristic auxiliary problem, with a feasible region denoted by $\bar{\mathcal{I}}_{\mathcal{N}} := \bar{\mathcal{I}} \cap \mathcal{N}_x$, is given by

$$\min_{x,\varphi}\{f(x) + \varphi \mid (x, \varphi) \in \bar{\mathcal{I}}_{\mathcal{N}}\}, \tag{7}$$

where $f(x)$ is either $c^\top x$ or an alternative objective function that is selected to guide the large neighbourhood search, such as a proximity function (Fischetti and Monaci 2014). Note that $\bar{\mathcal{I}}_{\mathcal{N}}$ includes the optimality and feasibility cuts that have been previously generated within the BD algorithm, given by the dual solutions and rays contained in $\bar{\mathcal{P}}$ and $\bar{\mathcal{R}}$ respectively.

Given a solution $\hat{x}$ to (7), the upper bound on the original problem is computed by

$$UB(\hat{x}) = \begin{cases} c^\top \hat{x} + z(\hat{x}) & \text{if SP is feasible,} \\ \infty & \text{otherwise.} \end{cases} \tag{8}$$

Since $\bar{\mathcal{I}}_{\mathcal{N}}$ represents a restriction of the feasible region given by $\bar{\mathcal{I}}$, the second stage constraints, denoted by $\Theta := \{(x, \varphi) \mid \varphi \geq u^\top (h - Bx), \forall u \in \mathcal{P};\ 0 \geq u^\top (h - Bx), \forall u \in \mathcal{R}\}$, are still valid for the auxiliary problem. Given solution $\hat{x}$ to (7), it is possible to compute an upper bound for the auxiliary problem when considering the second stage constraints Θ by solving SP. Such an upper bound is given by

$$UB_{aux}(\hat{x}) = \begin{cases} f(\hat{x}) + z(\hat{x}) & \text{if SP is feasible,} \\ \infty & \text{otherwise.} \end{cases} \tag{9}$$

Considering (8) and (9), there are two possible methods for solving the auxiliary problem of LNS heuristics. The classical approach is to solve (7) directly without considering the constraints that have been transferred from the original problem to the subproblems. The second stage constraints are only then considered at the termination of the LNS heuristic algorithm when verifying the feasibility or optimality of the candidate solution. This classical approach will be described as solving the auxiliary problem by branch-and-bound. Limitations of the classical approach are that candidate solutions may be infeasible with respect to the second stage constraints, or more commonly, feasible but of poor quality due to a large second stage cost.

The alternative method, the LNBS, is to employ BD to enforce the subproblem constraints in the auxiliary problem. Consider the sets of dual extreme points and rays given by $\hat{\mathcal{P}} \subseteq \mathcal{P} \backslash \bar{\mathcal{P}}$ and $\hat{\mathcal{R}} \subseteq \mathcal{R} \backslash \bar{\mathcal{R}}$, respectively, and denote the set of solutions satisfying the Benders' cuts computed using $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$ as $\hat{\Theta} := \{(x, \varphi) \mid \varphi \geq u^\top (h - Bx), \forall u \in \hat{\mathcal{P}};\ 0 \geq u^\top (h - Bx), \forall u \in \hat{\mathcal{R}}\}$. When employing the LNBS, the auxiliary problem of the LNS heuristics is of the form

$$\min_{x,\varphi}\{f(x) + \varphi \mid (x, \varphi) \in \bar{\mathcal{I}}_{\mathcal{N}} \cap \hat{\Theta}\}. \tag{10}$$

Since the sets $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$ are prohibitively large, a constraint generation procedure, such as BD, is useful for solving (10). In the following, this alternative method will be described as solving the auxiliary problem by BD.

The solving process of SCIP, presented in Fig. 1, aids in explaining the practical difference between these two solution methods. Specifically, solving (7) by branch-and-bound is achieved by executing the SCIP solving process without including the BENDERSDECOMP and BENDERSDECOMPLP constraint handlers. Alternatively, solving (10) by BD is achieved by executing the SCIP solving process as presented in Fig. 1.

Solving (10) by BD executes feasibility and optimality verification, by solving SP, during the LNS heuristic algorithm. Therefore, when employing LNBS all feasible solutions to the auxiliary problem satisfy the second stage constraints. This feature is a major advantage of the LNBS compared to the classical use of LNS heuristics. Performing the optimality check during the execution of the LNS heuristic can lead to improved solution quality with respect to the original problem. This is explained by Lemma 1 and Theorem 1.

Remark 1 All solutions feasible for (10) are feasible for the original problem.

Lemma 1 *If x' and x'' are optimal solutions to* (7) *and* (10) *respectively, then*

$$UB_{aux}(x') \geq UB_{aux}(x'').$$

Theorem 1 *Let $f(x) = c^\top x$ in* (7) *and* (10)*. If x' and x'' are optimal solutions to* (7) *and* (10) *respectively, then*

$$UB(x') \geq UB(x'').$$

Proof Since $f(x) = c^\top x$, the computation of the upper bound for the original and auxiliary problems, given by (8) and (9) respectively, are identical. Since $UB_{aux}(x) = UB(x)$, using the result from Lemma 1, $UB_{aux}(x'') \leq UB_{aux}(x')$ if and only if $UB(x'') \leq UB(x')$. □

While the assumption on the objective function in Theorem 1 could seem overly restrictive, this situation is commonly observed in practice. Within SCIP, the vast majority of LNS heuristics that are currently available do not modify the objective function coefficients in the auxiliary problem. The most notable exception to this is Proximity Search, where the objective function in the auxiliary problem is replaced with a proximity function. Therefore, in the majority of LNS heuristics, the LNBS will achieve an upper bound at least as good as that achieved when solving the auxiliary problem by branch-and-bound.

3.2 Trust region heuristic

The proposed TR heuristic is designed as an LNS heuristic to be embedded within a branch-and-cut solver. As such, this heuristic is called periodically during the branch-and-cut solution process along with the suite of heuristics available within a MIP solver. This significantly changes the focus of the trust region approach since the trade-off between primal and dual improvement must be balanced.

Consider the problem given by (3), which is decomposed by applying BD. The TR heuristic is applicable in cases where $x \in \{0, 1\}^r \times \mathbb{Z}^{p-r} \times \mathbb{R}^{n-p-r}$, since the constraints used to define the neighbourhood $\mathcal{N}_x$ rely on the existence of binary variables. However, if MP doesn't contain binary variables, but integer and continuous variables, then a suitable reformulation of the integer variables can be performed for the application of this heuristic. The trust region is formed around an incumbent solution, which is denoted by $\bar{x}$, and is bounded by its objective value, denoted by $\bar{z}$. Let $\mathcal{B}$ be the index set of the binary variables and $\mathcal{B}^+$ contain the indices of binary variables that are non-zero in the incumbent solution. The auxiliary problem within the TR heuristic is formed by adding the following constraint to (6),

$$\sum_{i \in \mathcal{B}^+} (1 - x_i) + \sum_{i \in \mathcal{B} \setminus \mathcal{B}^+} x_i \leq \theta, \tag{11}$$

where $\theta \in \mathbb{R}$ is a variable to count the number of changes between $\hat{x}$ and all other feasible solutions. Additionally, an upper bounding constraint of $c^\top x + \varphi \leq \bar{z} - \epsilon$ is added to (6). The setting of ϵ must consider the expected bound change for solutions within the neighbourhood of the current incumbent. It is possible to set ϵ dynamically during the solution process. The objective function for the TR heuristic is modified to $c^\top x + \varphi + M\theta$. The constant M is a large value that penalises the difference between the feasible solutions in the current iteration and the incumbent solution.

4 Implementation

The proposed enhancement techniques are general approaches for BD implemented within a branch-and-cut solver. While general, the LNBS modifies the algorithm behaviour of the LNS heuristics. Thus, a MIP solver that is available in source code is a necessary part of the implementation. This is due to the fact that, to the best of the authors' knowledge, the callback functions provided by most modern MIP solvers are not called when solving auxiliary problems of LNS heuristics.

The implementation of the LNBS extends the BD framework available in SCIP. Additionally, the TR heuristic extends the set of available LNS heuristics within SCIP, making it the first general purpose LNS heuristic designed for BD. In this section, practical implementation details for the LNBS will be presented followed by a description of the TR heuristic implementation.

While the BD algorithm can be implemented within many different solvers, this is not true for the LNBS. Callbacks that are typically used to implement BD in modern branch-and-cut solvers, such as the generic callback with the `CANDIDATE` context in CPLEX are, at present, not called while solving sub-MIPs. Hence, in CPLEX it is only possible to perform the SP check at the termination of the LNS heuristic through the use of callbacks. In this context, only the final solution from a LNS heuristic is passed to the generic callback from which a Benders' cut can be generated. The ability to modify the internal algorithms of the solver SCIP to implement the LNBS makes this general enhancement technique a unique feature of the BD framework.

4.1 Large neighbourhood Benders' search

The general BD framework in SCIP provides the necessary ground work for the implementation of the LNBS. Most importantly, the BD framework provides a tighter integration with the internal algorithms of SCIP than custom built branch-and-cut implementations. Also, extending the general BD framework with the LNBS means that the proposed methods are available to all applications of BD.

The implementation exploits the fact that the auxiliary problems of LNS heuristics are also MIP instances, termed sub-MIPs. As explained in Sect. 3.1, the key differences between the LNBS and the traditional application of LNS heuristics are related to the solving process. For the LNBS the SCIP solving process, as shown in Fig. 1, applied to solve the sub-MIP includes the BENDERSDECOMP and BENDERSDECOMPLP constraints. Whereas, for the traditional application of LNS heuristics the BD constraint handlers are omitted.

The different features of the solving process for the auxiliary problem sub-MIP are enabled during the initialisation of the LNS heuristic. This involves a *copying* process, whereby the require data structures from the original MIP are made available to the sub-MIP. As such, the BD algorithm is enabled in the solving process of the sub-MIP by copying the BENDERSDECOMP and BENDERSDECOMPLP constraint handlers from the original MIP.

An important feature of the implementation from a software design perspective is the copying of the BD subproblems to the sub-MIP. The implementation of the LNBS avoids copying the BD subproblems from the original MIP to the sub-MIP by exploiting properties of the LNS auxiliary problem. As explained in Sect. 3.1, the feasible region $\bar{\mathcal{I}}_{\mathcal{N}}$ is a subset of $\bar{\mathcal{I}}$. However, all second stage constraints from the original problem are valid for the auxiliary problem and thus, SP is a valid Benders' cut generating LP. A copy of each SP is not necessary when applying BD to solve a sub-MIP, since the original SPs can be used.

Since the LNBS employs BD to solve the auxiliary problem, this process can be very time consuming—especially considering that the auxiliary problem may already be a difficult optimisation problem. Thus, working limits have been imposed to reduce the amount of time that is spent generating Benders' cuts while solving the auxiliary problem. The working limits available in the implementation are:

(i) A maximum tree depth at which the LNBS is employed. The maximum depth is based upon the branch-and-bound tree created while solving the master problem. If an LNS heuristic is called at a node depth greater than the maximum depth, then the LNBS is not employed. This parameter is set to either 20 or ∞ for the computational results.
(ii) The number of times the subproblems are called when solving the auxiliary problem is limited to 10.

(iii) A parameter is provided to limit number times the subproblems are called when solving the LP of the auxiliary problem. In the computational results, this parameter is set to ∞.

4.2 Trust region heuristic

The TR heuristic is an LNS heuristic implemented within the plugin structure of SCIP. The first step of the implemented algorithm is to check whether it is possible execute the TR heuristic. For the TR heuristic to be executed there must be (i) at least 2 binary variables in the master problem, (ii) an incumbent solution, (iii) at least one node has been processed since the incumbent was found and (iv) the incumbent solution was not found by the trivial heuristic. If these conditions are satisfied, then, similar to other LNS heuristics available within SCIP, a sub-MIP is created as a copy of the original problem. The sub-MIP is then modified as described in Sect. 3.2.

The resulting sub-MIP is solved, with working limits, to find an improving solution for the original problem. The working limits are an important part of the implementation that helps to ensure the execution of the TR heuristic does not have a negative effect on the overall performance of the BD algorithm. Similar to many LNS heuristics, the working limits that are imposed are related to the calling frequency, the number of nodes to process, the number of stalling nodes and the number of best solutions found. For the TR heuristic, the working limits are set as follows:

(i) The calling frequency is used to limit how often the heuristic is executed. This is imposed by setting a depth frequency f, so that the heuristic is called at nodes at each f-th depth level. By default, the TR heuristic is disabled by setting f to -1. In the computational experiments when the TR heuristic is enabled, f is set to 25.

(ii) To avoid excessive run times for the heuristic, a limit l on the number of nodes to process is imposed. This is given by a function of the number of nodes processed (l_p), solutions found by the TR heuristic (s) and TR heuristic calls (e)

$$l = Ql_p(1 + 2(s + 1)/(e + 1)) - Se + l_o, \tag{12}$$

where $Q \in [0, 1]$ is a factor of the total number of nodes, $S \geq 0$ represents a setup cost in terms of nodes and l_o is a constant offset for the number of nodes to be processed. It is possible that the maximum number of nodes for the auxiliary problem could decrease to zero.

The different aspects that are captured in (12) are: (a) the l_p factor aims to provide more nodes to the heuristic at later stages in the solving process, (b) $(s + 1)/(e + 1)$ supplies more solving effort to the heuristic the more successful it is, (c) Se accounts for the fixed cost of performing the heuristic set up, taking nodes away from the allocated solving effort, and finally, (d) l_o is an initial node budget provided to the heuristic.

(iii) The stall nodes is the number of nodes without incumbent improvement. This is set to $\max\{n/10, 10\}$.

(iv) During the solving of the sub-MIP, the incumbent solution can be updated at most 3 times.

A feature of the TR heuristic is that if the sub-MIP solves to find an improving solution, then the heuristic is immediately executed again. In this case, the sub-MIP is set up with the solution that was found during the previous execution of the heuristic. The TR heuristic will terminate only when no improving solution is found.

As described in Sect. 3.2, a set of constraints are added to the sub-MIP and an additional term is added to the objective function. The parameters for these modifications are $\epsilon = 10^{-2}$ and $M = 100$. For working limit (ii), in the computational experiments the values of the run time parameters are set to $Q = 0.05$, $S = 100$ and $l_0 = 1000$.

5 Computational experiments

The computational experiments will assess the performance of the LNBS and TR heuristic within the general BD framework of SCIP 6.0. Since the primary focus of this paper is the BD algorithm, the computational results will focus purely of the impact that the proposed approaches have on this algorithm. The results presented in this section will provide insight into the main drivers of improved heuristic performance for the LNBS and TR heuristic.

Two different problem sets will be used for the computational results: the unrooted set covering connected subgraph problem (USCCSP) and the stochastic capacitated facility location problem (SCFLP). These test sets were selected because they contain instances that are unable to be solved to optimality within the given time limit. Since the motivation for developing the proposed methods is to improve the heuristic performance of the BD algorithm, the results will be evaluated using the primal integral (Berthold 2013), which is described below. The shifted geometric mean (shmean), with a shift of 100, is used when reporting the aggregation of results across subsets of instances.

Figures have been used to aid the visual interpretation of the results. These figures, except for Fig. 2, are presented as performance profiles (Dolan and Moré 2002) to highlight the overall effect of the evaluated algorithms. To improve the readability of the figures, the x-axis of the performance profiles have been cut off at 30. It is deemed that ratios greater than 30 do not provide any further meaningful information. The data used to produce the figures can be retrieved from www.stephenjmaher.com/publications/. All of the analysis in this paper has been performed using IPET (Hendel).

All computational experiments have been performed using the BD framework in SCIP with SoPlex 4.0 as LP solver. The computational experiments have been conducted using an Intel(R) Xeon(R) CPU E5-2670 2.5GHz processors with 64 GB RAM.

Primal integral. The primal integral is a measure that aims to capture the trade-off between the computational effort and the quality of solutions found. Let $\hat{\delta}$ denote the best known primal bound and $\delta(t)$ the primal bound at time t. The primal gap at time t is given by

$$\gamma(t) = \begin{cases} 1 & \text{if } \delta(t) = \infty \text{ or } \delta(t) \times \hat{\delta} < 0, \\ \frac{\delta(t) - \hat{\delta}}{\max\{|\delta(t)|, |\hat{\delta}|\}} & \text{otherwise.} \end{cases}$$

The primal integral is then given by

$$PI = \int_0^T \gamma(t)\mathrm{d}t,$$

where T is either the run time to solve the instance or the time limit if the instance is unsolved.

Settings used for computational experiments. A number of different settings for SCIP and the BD framework are used in the computational experiments. For all experiments *presolving* and *propagation* are disabled and for the SCFLP instances *separation* is disabled. In addition, the ALNS and RENS heuristics are disabled. Within the BD framework, the generation of cuts on improving, non-optimal integer feasible solutions is enabled. Also, the LNBS max depth is set to 20 and ∞ for the USCCSP and SCFLP instances respectively. All other settings are used as per default for SCIP 6.0.

The settings used for the evaluation of the LNBS and TR heuristic (including the non-default settings stated previously) are:

- *Benders*: the default BD algorithm within SCIP 6.0.
- *LNS check*: *Benders* with the LNBS enabled.
- *Trust Region*: *Benders* with the TR heuristic enabled by setting the frequency to 25.
- *TR LNS check*: *Trust Region* with the LNBS enabled.

A time limit of 3600 seconds has been used for all experiments.

5.1 Unrooted set covering connected subgraph problem

The USCCSP is a variant of graph connectivity problems, which was originally proposed by Maher and Murray (2016) for the analysis of HIV amino acid sequences. Consider a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges connecting the vertices V. The set P contains features that are observed on the vertices V and P^v denotes the set of features that are observed on vertex v, where $P^v \subseteq P$. The objective of the USCCSP is to identify a minimum cost set of vertices $\bar{V} \subset V$, such that $\cap_{v \in \bar{V}} P^v = P$ and the graph induced by the selected vertices $G(\bar{V}) = (\bar{V}, E(\bar{V}))$ is connected. An alternative formulation to that presented by Maher and Murray (2016) has been used in this paper since initial experiments showed it to be more computationally effective. The problem formulation for the USCCSP is provided in Appendix 1.

The instances for the USCCSP have been constructed using set covering problem (SCP) instances from the OR-Library (Beasley 1990). Specifically, `scp4{1-10}`, `scp6{1-5}`, `scpclr1{0-3}`, `scpcyc{06-10}` (`scpcyc11` was too large and exceeded the memory of the available machine), and `scpe{1-5}`. The remaining SCP instances from the OR-Library formed USCCSP instances that were too easy for the computational experiments and thus did not provide any useful insights into the proposed algorithms.

The instances for the USCCSP are formed by using the constraint matrix structure of the SCP instances. We define the set $P := \{p_1, p_2, \ldots, p_m\}$, where m is the number

of rows in the constraint matrix. Each row in the constraint matrix corresponds to a feature contained on the vertices of graph G. Let $V := \{v_1, v_2, \ldots, v_n\}$, where n is the number of columns in the constraint matrix, i.e. each column corresponds to a vertex in G. The set P^v contains the features corresponding to the row indices of the non-zero elements of column v. As such, the columns of the SCP instances represent the vertices of the underlying graph and the rows represent the features that must be covered in an optimal solution. Edges for the graph are randomly generated using the random number generator within SCIP as follows: For every vertex pair (i, j), where $i, j \in V$, select a random number α between 0 and 1. If $\alpha < \rho$, where ρ is the probability that two edges are connected, then an edge is added connecting i and j. For the computational experiments $\rho = 0.05$. Using the 29 SCP instances, random seeds of 1, 2, 3, 5 and 7 were provided to the random number generator in SCIP to produce different USCCSP instances. A total of 145 instances were used in the computational experiments.

5.2 Stochastic capacitated facility location problem

The SCFLP formulation is adapted from the model developed by Louveaux (1986). The stochastic variant of this problem involves selecting a set of facilities in the first stage and determining the amount of demand from each customer that is serviced by the open facilities in the second stage. The formulation of the SCFLP used in the paper is provided in Appendix 1.

The data for the deterministic capacitated facility location problem has been collected from the CAP instances of the OR-Library (Beasley 1990). For this paper, the instances with 25 and 50 customers are used. Taking these deterministic instances, the stochastic variant is constructed using the method described by Bodur et al. (2017). Specifically, for each scenario the demand for customer j is sampled from a normal distribution with a mean of λ_j and a standard deviation sampled from a uniform distribution in the range $[0.1\lambda, 0.3\lambda]$. Using this distribution, a random generator has been implemented in SCIP to produce 250 scenarios for each stochastic problem instance. To produce a large number of instances, the random generator is initialised with the seeds of 1, 2, 3, 5 and 7. From the 24 CAP instances, a total of 120 instances are used in the computational experiments.

5.3 Solution quality using the LNBS

The theoretical basis for the LNBS is that improved primal bounds from LNS heuristics can be achieved by solving the auxiliary problem by BD. While possible theoretically, MIP solvers are a complex collection of algorithms and the imposed working limits may mean that this behaviour is not observed in practice. Thus, the computational study presented in this section evaluates the potential of the LNBS to improve the primal bounds found by LNS heuristics within the solver SCIP.

This experiment focuses on the impact of the LNBS on the primal bounds found by the LNS heuristic RINS when solving the SCFLP instances by BD. Each time RINS is called during the solution process the heuristic is called twice with default working

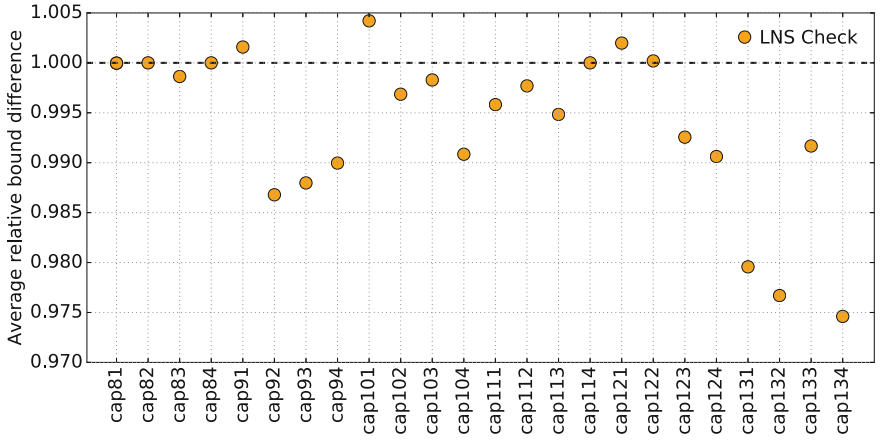

Fig. 2 Comparison of bounds achieved by executing RINS by default and with the LNBS for the SCFLP instances using random seed 1

limits: First, the auxiliary problem is solved by branch-and-bound and second, the auxiliary problem is solved by BD. To ensure that the solving behaviour of the second algorithm is not affected by the first, none of the solutions found are added to the master problem, i.e. RINS is executed for information purposes only. Also, the two auxiliary problems are formed prior to either of the algorithms being executed. Finally, when the final solutions from each execution of RINS are checked for feasibility by the BD subproblems, no cuts are generated for the master problem.

Figure 2 displays a comparison of the quality of solutions found by RINS when solving the SCFLP instances generated with a random seed of 1. The presented results are the arithmetic means of the ratio between the best solutions found at each call of the RINS heuristic when the auxiliary problem is solved by branch-and-bound and by BD. For each call of the RINS heuristic, the ratio is only calculated if both algorithms find valid solutions. A ratio less than 1 indicates that the LNBS produces a better primal bound than solving the auxiliary problem by branch-and-bound.

It can be seen in Fig. 2 that the LNBS produces a better bound on average. For many of the calls to RINS, the solution found by both algorithms is identical. As such, the average results do not fully show the extent to which the LNBS can aid in improving the primal bound; however, it does provide an overview of the performance. The results show that an improvement of up to 2.5% per call of RINS can be achieved, where the largest improvement for a single call of RINS is 12.5% for `cap134`.

5.4 Improving heuristic performance using the LNBS

As demonstrated in Sect. 5.3, solving the auxiliary problem of LNS heuristics by BD can have a significant effect on the quality of the solutions found. The main aim of the LNBS is to improve the overall heuristic performance of the BD algorithm by finding high quality solutions early in the solution process. This effect is observed through a decrease in the primal integral compared to the default BD implementation.

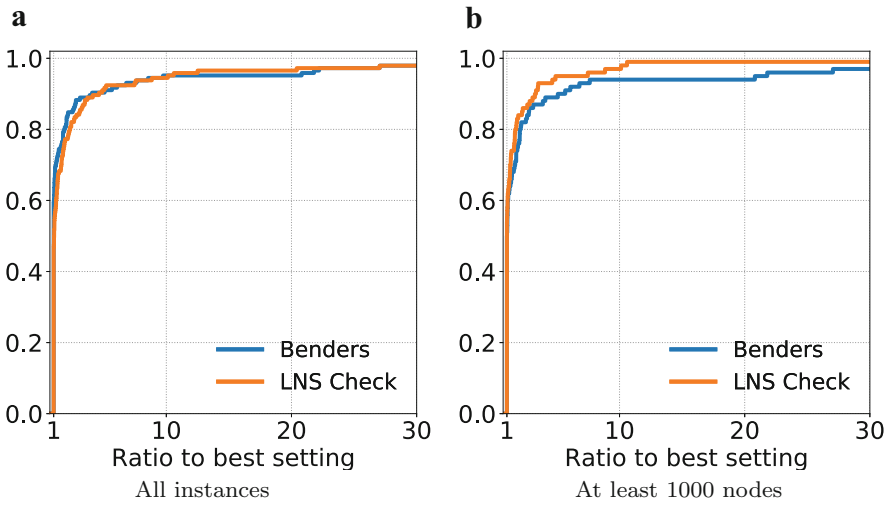

Fig. 3 Comparing the primal integral using *Benders* and *LNS check* for the USCCSP instances

5.4.1 Unrooted set covering connected subgraph problem

The solver performance on the USCCSP instances—in terms of run time, nodes and primal integral—is mixed and thus these instances provide a good test set to demonstrate the potential and limitations of the LNBS. A comparison of the primal integral between *Benders* and *LNS check* is presented in Fig. 3. It can be seen in Fig. 3a that across the complete test set *LNS check* achieves a similar performance to *Benders*. However, when considering the difficult instances—those where at least one setting (*Benders*, *LNS check*, *Trust Region* or *TR LNS check*) requires at least 1000 branch-and-bound nodes—Fig. 3b shows that *LNS check* dominates *Benders*. This demonstrates a clear advantage of employing the LNBS when applying BD to solve difficult instances.

The performance of the LNBS on the most difficult instances is further highlighted with shmean of the primal integral. In particular, the shmean of the primal integral for *Benders* and *LNS check* is 43,301 and 36,654 respectively. This is a 15.35% decrease in the primal integral when using the LNBS. In comparison to the complete test set, the shmean of the primal integral for *Benders* and *LNS check* is 28,813 and 30,439 respectively—a 5.64% degradation. Overall, the results demonstrate that for sufficiently difficult instances the LNBS can achieve significant performance improvements in the heuristic behaviour of BD.

5.4.2 Stochastic capacitated facility location problem

The performance of BD, and evaluated algorithms, has much less variation for the SCFLP test set compared to the USCCSP test set. This is primarily due to the fact that the underlying instances, the `cap` instances from the OR-Library, are much more

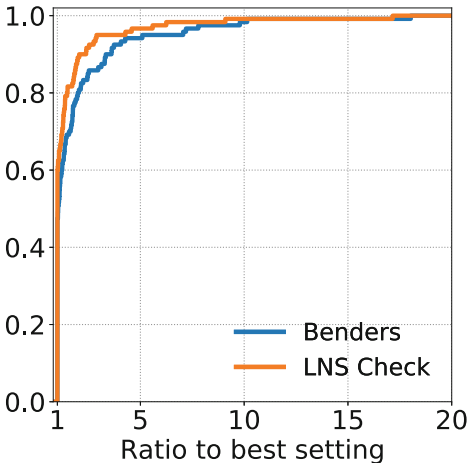

Fig. 4 Comparing the primal integral using *Benders* and *LNS check* for the SCFLP instances

similar to each other than the `scp` instances used for the USCCSP. As such, the results for the LNBS are more consistent across the SCFLP test set.

A comparison between *Benders* and *LNS check* with respect to the primal integral is presented in Fig. 4. None of the SCFLP instances could be solved to optimality within the time limit. As such, the ability to quickly find high quality primal bounds is an important feature of any solution algorithm, which is captured by the primal integral. The performance profiles in Fig. 4 show that *LNS check* outperforms *Benders*. This is more clearly demonstrated by the shmean of the primal integral for *Benders* and *LNS check*, which is 3955 and 3453 respectively across the complete test set. This result corresponds to a decrease of 12.7% in the primal integral.

Typically, LNS heuristics are called periodically during the branch-and-bound search with the frequency of execution in SCIP related to the node depth. For example a frequency of 10 means that the heuristic is called at nodes with a depth of 10, 20, 30, and so on. Considering this algorithmic design for LNS heuristics, a factor contributing to the decrease in the primal integral by *LNS check* with respect to *Benders* is that the minimum number of nodes processed by *Benders* for the SCFLP instances is 4094 and the shmean of nodes processed is 11,084. A further important observation is that the shmean of the number of nodes to find the best solution for *Benders* and *LNS check* is 1472 and 1217 respectively. The large number of nodes to find the best solution means that the LNS heuristics are called more often, thus the LNBS has more opportunities to find higher quality solutions. If the number of nodes to find the best solution is not sufficiently large, then the use of the LNBS will typically degrade the performance of the BD algorithm. In this respect, the SCFLP and difficult USCCSP instances are well suited to the use of the LNBS.

5.5 Comparing the LNBS and TR heuristic

The TR heuristic is proposed as an alternative to the Local Branching (Fischetti and Lodi 2003) and Proximity Search (Fischetti and Monaci 2014) heuristics, the latter

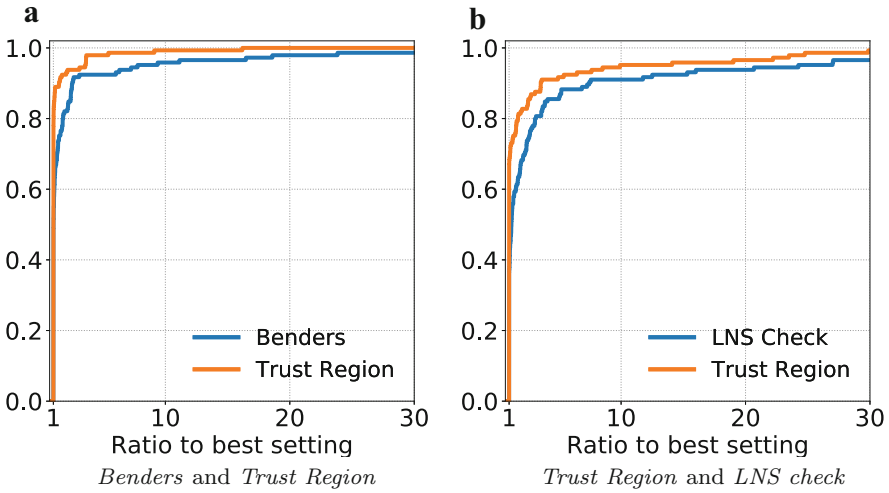

Fig. 5 Comparing the primal integral using *Benders*, *Trust Region* and *LNS check* for all USCCSP instances

which has been used as an outer algorithm for BD by Boland et al. (2016). It must be noted that both Local Branching and Proximity Search are not enabled in SCIP 6.0 by default and thus the default internal BD algorithm does not employ either of these two inner algorithms. While Boland et al. (2016) demonstrate the potential of using Proximity Search for enhancing BD, its potential as a component algorithm within a solver has not been evaluated. Similarly, previous variants of the TR heuristic have also been proposed as outer algorithms for BD in the form of trust region methods. Thus, the following computational experiments evaluate the potential of using trust regions as an inner algorithm for BD.

Since the TR heuristic is an LNS heuristic, it can be further enhanced by the use of the LNBS. The computational experiments will first present the improved heuristic performance for BD from the use of the TR heuristic, comparing against *Benders* and *LNS check*. For the SCFLP instances, the improved performance achieved by combining the LNBS with the TR heuristics will be shown.

5.5.1 Unrooted set covering connected subgraph problem

Figure 5 compares the primal integral for the settings *Benders*, *Trust Region* and *LNS check*. Similar to Sect. 5.4.1, the performance profiles including only the most difficult instances are also constructed and displayed in Fig. 6.

The first observation from comparing Figs. 3 and 5 is that *Trust Region* achieves a much greater reduction in the primal integral than *LNS check* with respect to *Benders*. The performance profiles using all USCCSP instances show that *Trust Region* completely dominates *Benders*. Most notably, for 64% of instances *Trust Region* achieves the best primal integral, compared to 36% for *Benders*. The shmean of the primal integral achieved by *Trust Region* and *Benders* is 22,798 and 28,813 respectively, a 20.9% decrease. This result further confirms the benefit from employing the TR heuristic.

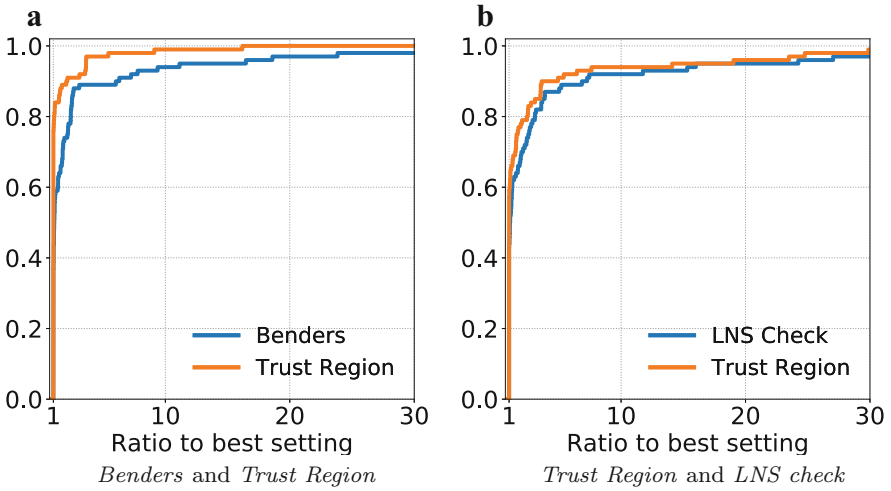

Fig. 6 Comparing the primal integral using *Benders*, *Trust Region* and *LNS check* for the USCCSP instances requiring at least 1000 nodes

When only considering the instances that require at least 1000 nodes, shown in Fig. 6, *Trust Region* achieves the best primal integral for 74% of the instances compared to 26% for *Benders*. In regards to the shmean, *Trust Region* achieves a 27.4% decrease in the primal integral compared to *Benders* for the difficult instances (31,429 and 43,301 respectively). The improved performance of *Trust Region* when focusing on the more difficult instances is due to the fact that for easier instances the achieved primal integral is similar to that achieved by *Benders*.

This result can be explained by the frequency setting of 25 for the TR heuristic, meaning that if *Trust Region* does not produce a tree with a depth greater than 25 then the TR heuristic may not be called. Many of the easier USCCSP instances are solved at the root node. As such, in these cases there will be no difference in the algorithm behaviour between *Trust Region* and *Benders*. An important observation from the computational results is that if the TR heuristic is called, then it generally finds an improving solution—leading to a decrease in the primal integral. This highlights the potency of the TR heuristic for the USCCSP instances.

Comparing *Trust Region* and *LNS check*, Fig. 5b shows that *Trust Region* is a superior method for improving the heuristic performance of BD. Specifically, across the complete test set, the shmean of the primal integral for *Trust Region* is 22,798, which is a 25.1% decrease compared to *LNS check*.

When considering the most difficult instances, the performance profiles presented in Fig. 6b indicates that *Trust Region* is less effective on these instances compared to *LNS check*. However, *Trust Region* still achieves a shmean of the primal integral that is 14.3% better than that achieved by *LNS check*. The results presented in Figs. 5b and 6b suggest the balanced use of *Trust Region* and *LNS check* can significantly improve the heuristic performance of BD.

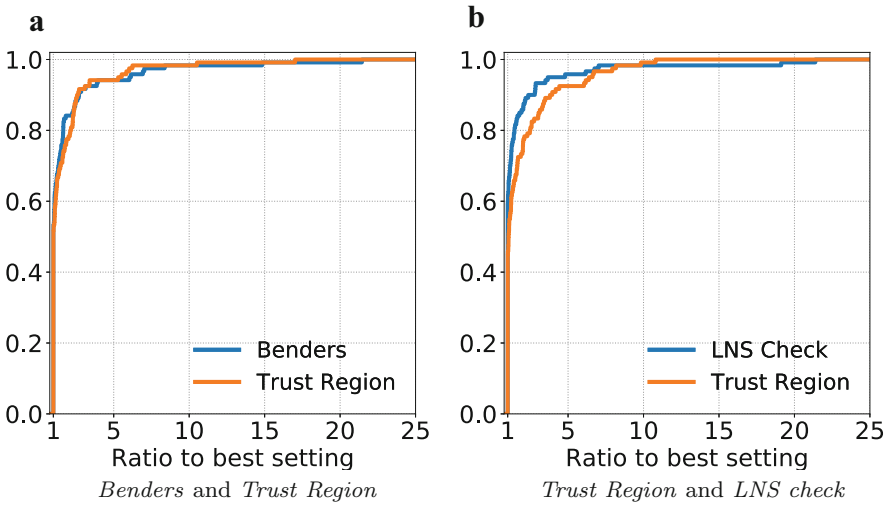

Fig. 7 Comparing the primal integral using *Benders*, *Trust Region* and *LNS check* for the SCFLP instances

5.5.2 Stochastic capacitated facility location problem

Figure 7 shows that *Trust Region* is not very effective when applied to the SCFLP instances. These results demonstrate that the problem structure is an important factor driving the performance of the TR heuristic. For *Trust Region*, the TR heuristic was called when solving a single instance at most 9 times across the complete test set, and 2.9 times on average. Only on 11 instances the TR heuristic managed to find an integer feasible solution. In comparison, for the USCCSP instances the TR heuristic was called up to 25 times, with average of 4.5 times per instance. When solving the USCCSP instances using *Trust Region*, the TR heuristic manages to find an average of 24.24 integer feasible solution, all of which are improving solutions.

The primal integral results for *TR LNS check* presented in Fig. 8 show that the LNBS significantly improves the performance of the TR heuristic for the SCFLP instances. While the average number of times the TR heuristic is called slightly decreases—from 2.9 to 2.85—improving solutions are now found in at least one call per instance. This result highlights one of the key features of the LNBS, where the solution quality is improved by solving the auxiliary problem by BD.

Figure 8a and b also show that *TR LNS check* achieves the best primal integral on approximately 62% of instances relative to both *Benders* and *Trust Region*. Specifically, the shmean of the primal integral for *Benders*, *Trust Region* and *TR LNS check* is 3955, 4043 and 3130 respectively. That corresponds to a respective improvement of 20.9% and 22.6% by *TR LNS check* over *Benders* and *Trust Region*. This improvement is due, in part, to the performance of *LNS check* for the SCFLP instances, since the LNS heuristics that are active in the *Benders* setting are still active with the *TR LNS check* setting. However, Fig. 8(c) shows that *TR LNS check* also achieves an improvement in the primal integral over *LNS check*. In regards to the shmean, *TR LNS check* achieves and improvement of 9.3% over *LNS check*. While the TR heuristic alone fails

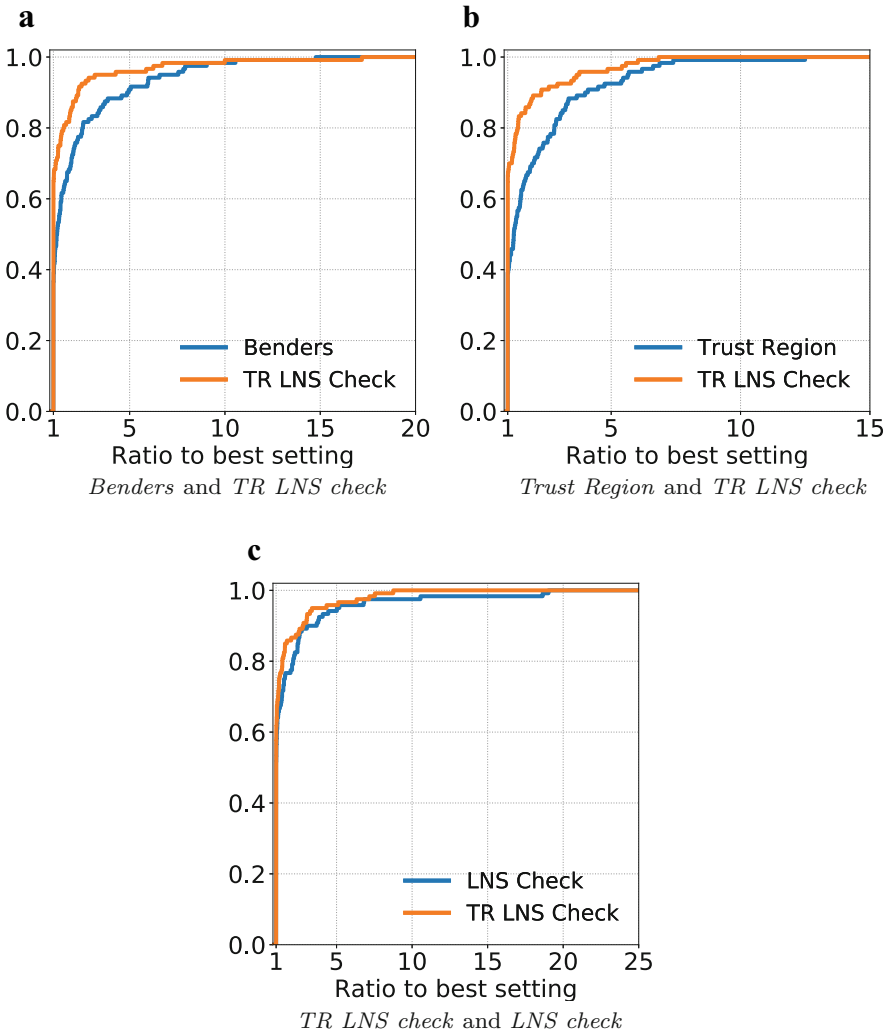

Fig. 8 Comparing the primal integral using *Benders*, *Trust Region*, *TR LNS check* and *LNS check* for the SCFLP instances

to improve the heuristic performance of BD for the SCFLP instances, the combined use with the LNBS results in a larger number of solutions being found and reduces the primal integral.

6 Concluding remarks

This paper presents two different, but complementary, approaches for improving the heuristic performance of the BD algorithm—the LNBS and the TR heuristic. The main contribution of the LNBS is a systematic way to integrate LNS heuristics and BD as

a general algorithmic enhancement technique. Specifically, through a software and algorithmic change to SCIP, the auxiliary problems of LNS heuristics can be solved by BD. The proposed approach extends the integration of these two methods from bespoke approaches previously presented in the literature (Rei et al. 2009; Boland et al. 2016) to all available LNS heuristics within SCIP. The TR heuristic has been developed by building upon the close relationship of LNS heuristics and trust region methods. Employing a trust region approach as an inner algorithm for BD presents an alternative method for using successful trust region algorithms.

A detailed computational study is presented to evaluate the performance of the proposed techniques within the general BD framework available in SCIP 6.0. The initial study shows that the quality of the solutions found by the RINS heuristic improves with the use of the LNBS. These results are shown to translate into an overall improvement to the heuristic performance of BD when solving the USCCSP and SCFLP instances, especially the difficult instances from the USCCSP test set. The TR heuristic is shown to significantly improve the heuristic performance of BD when solving the USCCSP instances. The same performance is not achieve for the SCFLP instances. However, for the SCFLP instances, the performance of the TR heuristic is shown to be greatly improved with the use of the LNBS. A major conclusion from the results is that while the individual strategies of the LNBS and the TR heuristic can greatly improve the heuristic performance of BD, a balanced mix of strategies is crucial for achieving the best performance from the algorithm. The availability of the BD framework within SCIP greatly simplifies the task of developing such balanced strategies.

The proposed approaches and developed software open many possible directions for future research. A main focus of this paper is the extension of previously proposed outer algorithms for use as inner algorithms. Continuing this investigation is an interesting course of future research; for example using scenario decomposition (Ahmed 2013) as a start heuristic for BD. Also, an investigation and analysis of different algorithmic strategies for BD implemented within a branch-and-cut framework will provide great insight for future users of this popular mathematical programming algorithm. Finally, the LNBS is an extension of LNS heuristics, as such the variable fixings and neighbourhood definitions are still made with respect to the master problem. An area of future research will focus on extending neighbourhood definitions to include second stage variables and constraints.

Acknowledgements This work was support by the Engineering and Physical Sciences Research Council (EPSRC) Grant EP/P003060/1. The author would like to thank Gregor Hendel, Ambros Gleixner, Leona Gottwald, Felipe Serrano, Stefan Vigerske and Jakob Witzig for performing code reviews for the general Benders' decomposition framework in SCIP 6.0 and SCIP 7.0. The author would also like to thank the anonymous reviewers and the editor for the helpful comments that greatly improved the quality of this paper.

A Unrooted set covering connected subgraph problem formulation

In Maher and Murray (2016), the USCCSP is formulated by integrating the set covering problem with a connected network problem, in the form of a series of shortest path problems. In this paper, an alternative formulation is proposed that significantly reduces the number of variables and constraints in the connected network problem. As opposed to using the shortest path problem to model connectivity, a network flow problem is used. Two additional nodes are added to the network, dummy source and sink nodes, along with the edges (s, j) and (j, t) for all $j \in V$. A flow equal to the number of nodes selected in the set cover is input at the source, and constraints are added to enforce that exactly one edge (s, j), $j \in V$, can have a positive flow. The edge from each of the vertices selected in the set cover to the sink node has a flow capacity of 1, with all other edges connected to the sink node having a capacity of 0. No flow capacity is imposed on the edges in E. Thus, the graph $G(\bar{V})$, induced by the set covering solution $\bar{V}$ is connected if there exists a feasible flow from the source to the sink.

The formulation of the USCCSP used for the computational experiments is given by,

$$\min \quad \sum_{i \in V} c_i x_i + Q \sum_{i \in V} z_i + R \sum_{i \in V} \left(\epsilon_i^+ + \epsilon_i^- \right), \tag{13a}$$

$$\text{subject to} \quad \sum_{i \in V} a_{ip} x_i \geq 1 \qquad \forall p \in P, \tag{13b}$$

$$\sum_{j \in V} y_{sj} = \sum_{i \in V} x_i, \tag{13c}$$

$$y_{sj} \leq |N| w_j \qquad \forall j \in V, \tag{13d}$$

$$y_{sj} \leq |N| x_j \qquad \forall j \in V, \tag{13e}$$

$$w_j \leq w_{j-1} \qquad \forall j \in V \backslash \{0\}, \tag{13f}$$

$$w_j \leq 1 - x_{j-1} \qquad \forall j \in V \backslash \{0\}, \tag{13g}$$

$$\sum_{i \in V} y_{it} \leq x_j \qquad \forall j \in V, \tag{13h}$$

$$\sum_{i \in V} y_{ij} - \sum_{k \in V} y_{jk} = \epsilon_j^+ - \epsilon_j^- \qquad \forall j \in V, \tag{13i}$$

$$\sum_{i \in V} y_{ij} - x_j \leq z_j \qquad \forall j \in V, \tag{13j}$$

$$x_i, w_i \in \{0, 1\} \qquad \forall i \in V, \tag{13k}$$

$$y_{ij} \geq 0 \qquad \forall (i, j) \in E, \tag{13l}$$

$$z_i, \epsilon_i^+, \epsilon_i^+ \geq 0, \qquad \forall i \in V. \tag{13m}$$

The objective function minimises the cost of the set cover and any penalties resulting from sending a flow of $|\bar{V}|$ through a disconnected induced graph $G(\bar{V})$. The set of features that must be covered are contained in the set P. Feature p is observed on

vertex i if the parameter a_{ip} equals 1. Constraints (13b) ensure that all features are covered by the selected vertices. Connectivity of the induced graph $G(\bar{V})$ is given by the set of constraints (13c)–(13j). The input flow is given by constraint (13c) and the restriction of the flow on a single edge leaving the source is given by the set of constraints (13d)–(13g). Specifically, the bounding constraints (13d)–(13e) and the precedence constraints (13f)–(13g) permits flow only on the vertex with the lowest index that is selected in the set cover. To ensure that flow enters every vertex selected in set cover, the flow into the sink is restricted by constraints (13h). The flow balance in the network is given by constraints (13i). Since $\bar{V}$ may not induce a connected graph, a penalty is imposed for each vertex $v \notin \bar{V}$ that has a non-zero flow passing through it. This is achieved by the constraints (13j). Finally, since it is not guaranteed that G is connected, the variables ϵ_i^+ and ϵ_i^- are include in constraints (13i) to penalise any flow between disconnected vertices.

B Stochastic capacitated facility location problem formulation

The sets of facilities and customers are denoted by I and J respectively. The stochasticity for this problem is in the demands of the customers, which are given by the scenarios in set S. The following formulation is used for the computational experiments:

$$
\begin{aligned}
\min \quad & \sum_{i \in I} f_i x_i + \frac{1}{|S|} \sum_{s \in S} \sum_{i \in I} \sum_{j \in J} q_{ij} y_{ij}^s, \\
\text{subject to} \quad & \sum_{i \in I} y_{ij}^s \geq \lambda_j^s \qquad \forall j \in J, \forall s \in S, \\
& \sum_{j \in J} y_{ij}^s \leq k_i x_i \qquad \forall i \in I, \forall s \in S, \\
& \sum_{i \in I} k_i x_i \geq \max_{s \in S} \sum_{j \in J} \lambda_j^s, \\
& x_i \in \{0, 1\} \qquad \forall i \in I, \\
& y_{ij}^s \geq 0 \qquad \forall i \in I, \forall j \in J, \forall s \in S.
\end{aligned}
\tag{14}
$$

In the above formulation, the demand of customer j in scenario s is denoted by λ_j^s and and the capacity of facility i is denoted by k_i. The variable x_i equals 1 if facility i is opened, which has a cost of f_i, and 0 otherwise. The variable y_{ij}^s represents the amount of demand of customer j that is served by facility i in scenario s, which has a cost of q_{ij} per unit of demand.

C Instance sizes

The following tables present the instance sizes of the USCCSP and SCFLP instances. The number of variables and constraints reported are for the master problem and each subproblem. There is only a single subproblem for the USCCSP instances and 250 subproblems for the SCFLP instances.

Table 1 The number of variables and constraints in the master and subproblem for the USCCSP instances using random seed 1

ProblemName	*Mastervars*	*Mastercons*	*Subproblemvars*	*SubproblemCons*
scp41	1000	200	56,142	7000
scp410	1000	200	56,142	7000
scp42	1000	200	56,142	7000
scp43	1000	200	56,142	7000
scp44	1000	200	56,142	7000
scp45	1000	200	56,142	7000
scp46	1000	200	56,142	7000
scp47	1000	200	56,142	7000
scp48	1000	200	56,142	7000
scp49	1000	200	56,142	7000
scp61	1000	200	56,142	7000
scp62	1000	200	56,142	7000
scp63	1000	200	56,142	7000
scp64	1000	200	56,142	7000
scp65	1000	200	56,142	7000
scpclr10	210	511	3346	1470
scpclr11	330	1023	7328	2310
scpclr12	495	2047	15,096	3465
scpclr13	715	4095	29,984	5005
scpcyc06	192	240	2904	1344
scpcyc07	448	672	12,644	3136
scpcyc08	1024	1792	58,678	7168
scpcyc09	2304	4608	279,030	16,128
scpcyc10	5120	11,520	1,341,424	35,840
scpcyc11	11,264	28,160	6,411,332	78,848
scpe1	500	50	15,426	3500
scpe2	500	50	15,426	3500
scpe3	500	50	15,426	3500
scpe4	500	50	15,426	3500
scpe5	500	50	15,426	3500

Table 2 The number of variables and constraints in the master and subproblem for the USCCSP instances using random seed 2

ProblemName	*Mastervars*	*MasterCons*	*Subproblemvars*	*SubproblemCons*
scp41	1000	200	56,458	7000
scp410	1000	200	56,458	7000
scp42	1000	200	56,458	7000
scp43	1000	200	56,458	7000
scp44	1000	200	56,458	7000
scp45	1000	200	56,458	7000
scp46	1000	200	56,458	7000
scp47	1000	200	56,458	7000
scp48	1000	200	56,458	7000
scp49	1000	200	56,458	7000
scp61	1000	200	56,458	7000
scp62	1000	200	56,458	7000
scp63	1000	200	56,458	7000
scp64	1000	200	56,458	7000
scp65	1000	200	56,458	7000
scpclr10	210	511	3542	1470
scpclr11	330	1023	7442	2310
scpclr12	495	2047	15,366	3465
scpclr13	715	4095	30,256	5005
scpcyc06	192	240	3042	1344
scpcyc07	448	672	12,898	3136
scpcyc08	1024	1792	58,886	7168
scpcyc09	2304	4608	279,490	16,128
scpcyc10	5120	11520	1,340,662	35,840
scpcyc11	11,264	28,160	6,408,958	78,848
scpe1	500	50	15,658	3500
scpe2	500	50	15,658	3500
scpe3	500	50	15,658	3500
scpe4	500	50	15,658	3500
scpe5	500	50	15,658	3500

Table 3 The number of variables and constraints in the master and subproblem for the USCCSP instances using random seed 3

ProblemName	*Mastervars*	*Mastercons*	*Subproblemvars*	*SubproblemCons*
scp41	1000	200	56,616	7000
scp410	1000	200	56,616	7000
scp42	1000	200	56,616	7000
scp43	1000	200	56,616	7000
scp44	1000	200	56,616	7000
scp45	1000	200	56,616	7000
scp46	1000	200	56,616	7000
scp47	1000	200	56,616	7000
scp48	1000	200	56,616	7000
scp49	1000	200	56,616	7000
scp61	1000	200	56,616	7000
scp62	1000	200	56,616	7000
scp63	1000	200	56,616	7000
scp64	1000	200	56,616	7000
scp65	1000	200	56,616	7000
scpclr10	210	511	3576	1470
scpclr11	330	1023	7670	2310
scpclr12	495	2047	15,510	3465
scpclr13	715	4095	30,226	5005
scpcyc06	192	240	3108	1344
scpcyc07	448	672	13,020	3136
scpcyc08	1024	1792	59,120	7168
scpcyc09	2304	4608	279,456	16,128
scpcyc10	5120	11,520	1,340,402	35,840
scpcyc11	11,264	28,160	6,414,710	78,848
scpe1	500	50	15,788	3500
scpe2	500	50	15,788	3500
scpe3	500	50	15,788	3500
scpe4	500	50	15,788	3500
scpe5	500	50	15,788	3500

Table 4 The number of variables and constraints in the master and subproblem for the USCCSP instances using random seed 5

ProblemName	*Mastervars*	*Mastercons*	*Subproblemvars*	*SubproblemCons*
scp41	1000	200	55,872	7000
scp410	1000	200	55,872	7000
scp42	1000	200	55,872	7000
scp43	1000	200	55,872	7000
scp44	1000	200	55,872	7000
scp45	1000	200	55,872	7000
scp46	1000	200	55,872	7000
scp47	1000	200	55,872	7000
scp48	1000	200	55,872	7000
scp49	1000	200	55,872	7000
scp61	1000	200	55,872	7000
scp62	1000	200	55,872	7000
scp63	1000	200	55,872	7000
scp64	1000	200	55,872	7000
scp65	1000	200	55,872	7000
scpclr10	210	511	3404	1470
scpclr11	330	1023	7316	2310
scpclr12	495	2047	15,220	3465
scpclr13	715	4095	29,878	5005
scpcyc06	192	240	2992	1344
scpcyc07	448	672	12,682	3136
scpcyc08	1024	1792	58,334	7168
scpcyc09	2304	4608	278,784	16,128
scpcyc10	5120	11,520	1,338,936	35,840
scpcyc11	11,264	28,160	6,411,340	78,848
scpe1	500	50	15,490	3500
scpe2	500	50	15,490	3500
scpe3	500	50	15,490	3500
scpe4	500	50	15,490	3500
scpe5	500	50	15,490	3500

Table 5 The number of variables and constraints in the master and subproblem for the USCCSP instances using random seed 7

ProblemName	*Mastervars*	*Mastercons*	*Subproblemvars*	*SubproblemCons*
scp41	1000	200	55,912	7000
scp410	1000	200	55,912	7000
scp42	1000	200	55,912	7000
scp43	1000	200	55,912	7000
scp44	1000	200	55,912	7000
scp45	1000	200	55,912	7000
scp46	1000	200	55,912	7000
scp47	1000	200	55,912	7000
scp48	1000	200	55,912	7000
scp49	1000	200	55,912	7000
scp61	1000	200	55,912	7000
scp62	1000	200	55,912	7000
scp63	1000	200	55,912	7000
scp64	1000	200	55,912	7000
scp65	1000	200	55,912	7000
scpclr10	210	511	3460	1470
scpclr11	330	1023	7364	2310
scpclr12	495	2047	15,132	3465
scpclr13	715	4095	29,768	5005
scpcyc06	192	240	2952	1344
scpcyc07	448	672	12,540	3136
scpcyc08	1024	1792	58,468	7168
scpcyc09	2304	4608	278,942	16,128
scpcyc10	5120	11,520	1,341,044	35,840
scpcyc11	11,264	28,160	6,407,838	78,848
scpe1	500	50	15,366	3500
scpe2	500	50	15,366	3500
scpe3	500	50	15,366	3500
scpe4	500	50	15,366	3500
scpe5	500	50	15,366	3500

Table 6 The number of variables and constraints in the master and each subproblem for the SCFLP instances using all random seeds. NOTE: there are 250 subproblem for each instance

	Master vars	Master cons	Subproblem vars	$Subproblemcons$
cap101	25	1	1250	75
cap102	25	1	1250	75
cap103	25	1	1250	75
cap104	25	1	1250	75
cap111	50	1	2500	100
cap112	50	1	2500	100
cap113	50	1	2500	100
cap114	50	1	2500	100
cap121	50	1	2500	100
cap122	50	1	2500	100
cap123	50	1	2500	100
cap124	50	1	2500	100
cap131	50	1	2500	100
cap132	50	1	2500	100
cap133	50	1	2500	100
cap134	50	1	2500	100
cap81	25	1	1250	75
cap82	25	1	1250	75
cap83	25	1	1250	75
cap84	25	1	1250	75
cap91	25	1	1250	75
cap92	25	1	1250	75
cap93	25	1	1250	75
cap94	25	1	1250	75

References

Ahmed, S.: A scenario decomposition algorithm for 0–1 stochastic programs. Oper. Res. Lett. **41**(6), 565–569 (2013)

Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Appl. Math. **123**(1–3), 75–102 (2002)

Beasley, J.E.: OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990)

Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**(1), 238–252 (1962)

Berthold, T.: Measuring the impact of primal heuristics. Oper. Res. Lett. **41**(6), 611–614 (2013)

Berthold, T.: Heuristic algorithms in global MINLP solvers. PhD thesis (2014)

Bodur, M., Dash, S., Günlük, O., Luedtke, J.: Strengthened Benders cuts for stochastic integer programs with continuous recourse. INFORMS J. Comput. **29**(1), 77–91 (2017)

Boland, N., Fischetti, M., Monaci, M., Savelsbergh, M.: Proximity Benders: a decomposition heuristic for stochastic programs. J. Heuristics **22**(2), 181–198 (2016)

Boschetti, M.A., Maniezzo, V., Roffilli, M., Bolufé Röhler, A.: Matheuristics: optimization, simulation and control. In: Blesa, M.J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.) Hybrid Metaheuristics, pp. 171–177. Springer, Berlin (2009)

Canto, S.P.: Application of Benders' decomposition to power plant preventive maintenance scheduling. Eur. J. Oper. Res. **184**(2), 759–777 (2008)

Conn, A., Gould, N., Toint, P.: Trust Region Methods. Society for Industrial and Applied Mathematics (2000)

Cordeau, J.-F., Stojković, G., Soumis, F., Desrosiers, J.: Benders' decomposition for simultaneous aircraft routing and crew scheduling. Transp. Sci. **35**(4), 375–388 (2001)

Costa, A.M.: A survey on Benders decomposition applied to fixed-charge network design problems. Comput. Operat. Res. **32**(6), 1429–1450 (2005)

Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Math. Program. **102**(1), 71–90 (2005)

Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Oper. Res. **8**(1), 101–111 (1960)

Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)

Fischetti, M., Fischetti, M.: Matheuristics, pp. 121–153. Springer, Cham (2018)

Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**, 23–47 (2003)

Fischetti, M., Monaci, M.: Proximity search for 0–1 mixed-integer convex programming. J. Heuristics **20**(6), 709–731 (2014)

Ghosh, S.: DINS, a MIP Improvement Heuristic, pp. 310–323. Springer, Berlin (2007)

Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. Technical Report 18-26, Zuse Institute Berlin (2018)

Hendel, G.: IPET interactive performance evaluation tools. https://github.com/GregorCH/ipet

Linderoth, J., Wright, S.: Decomposition algorithms for stochastic programming on a computational grid. Comput. Optim. Appl. **24**(2–3), 207–250 (2003)

Louveaux, F.V.: Discrete stochastic location models. Ann. Oper. Res. **6**(2), 21–34 (1986)

Magnanti, T., Wong, R.: Accelerating Benders' decomposition: algorithmic enhancement and model selection criteria. Oper. Res. **29**(3), 464–484 (1981)

Maher, S.J.: Implementing the branch-and-cut approach for a general purpose Benders' decomposition framework. Eur. J. Oper. Res. **290**(1), 479–498 (2021)

Maher, S.J., Desaulniers, G., Soumis, F.: Recoverable robust single day aircraft maintenance routing problem. Comput. Oper. Res. **51**, 130–145 (2014)

Maher, S.J., Murray, J.M.: The unrooted set covering connected subgraph problem differentiating between HIV envelope sequences. Eur. J. Oper. Res. **248**(2), 668–680 (2016)

Mercier, A., Cordeau, J., Soumis, F.: A computational study of Benders' decomposition for the integrated aircraft routing and crew scheduling problem. Comput. Oper. Res. **32**(6), 1451–1476 (2005)

Oliveira, F., Grossmann, I., Hamacher, S.: Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. Comput. Oper. Res. **49**, 47–58 (2014)

Papadakos, N.: Integrated airline scheduling. Comput. Oper. Res. **36**(1), 176–195 (2009)

Puchinger, J., Raidl, G.R.: Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In: Mira, J., Álvarez, J.R. (eds.) Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach Lecture Notes in Computer Science, vol. 3562, pp. 41–53. Springer, Berlin (2005)

Raidl, G.R.: Decomposition based hybrid metaheuristics. Eur. J. Oper. Res. **244**(1), 66–76 (2015)

Rei, W., Cordeau, J.-F., Gendreau, M., Soriano, P.: Accelerating Benders' decomposition by local branching. INFORMS J. Comput. **21**(2), 333–345 (2009)

Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS J. Comput. **19**(4), 534–541 (2007)

Ruszczyński, A.: A regularized decomposition method for minimizing a sum of polyhedral functions. Math. Program. **35**(3), 309–333 (1986)

Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A.: A stochastic programming approach for supply chain network design under uncertainty. Eur. J. Oper. Res. **167**(1), 96–115 (2005)

Yuan, Y.-X.: Recent advances in trust region algorithms. Math. Program. **151**(1), 249–281 (2015)