Large neighbourhood Benders' search

Stephen J. Maher

Department of Management Science, Lancaster University, Bailrigg, Lancaster LA1 4YX, UK

Abstract

A general enhancement of the Benders' decomposition 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, their use is typically limited to finding solutions to the Benders' decomposition master problem, which may be of poor quality for the original problem or even infeasible. Given the lack of general frameworks for Benders' decomposition, only ad hoc approaches have been developed to enhance Benders' decomposition through the use of large neighbourhood search heuristics. Within the solver SCIP, a Benders' decomposition framework has been developed to achieve a greater integration with the internal large neighbourhood search heuristics. Benders' decomposition is employed to solve the auxiliary problems of all large neighbourhood search heuristics, the Large Neighbourhood Benders' Search, to improve the quality of the identified solutions and generate additional cuts that can be used to accelerate the convergence of the main solution algorithm. The computational results demonstrate the performance improvements achieved by this general enhancement technique for Benders' decomposition.

Key words: Bender' decomposition, large-neighbourhood search, enhancement techniques, mixed integer programming

1 Introduction

Benders' decomposition is a popular mathematical programming technique for solving large scale optimisation problems. Originally proposed by Benders [4] to improve the tractability of problems with complicating variables, one of the major benefits of Benders' decomposition is that it effectively distributes the computational burden for solving problems exhibiting a block diagonal structure. Stochastic programming problems are particularly suited to the application of Benders' decomposition, where the L-shaped method is a regularly used solution technique. For an in depth review of Benders' decomposition, including the numerous applications and algorithmic enhancement techniques, the reader is

directed to Rahmaniani et al. [32].

Consider the following two-stage problem (P):

$$\min \quad c^\top x + \sum_{s \in S} d_s^\top y_s, \tag{1}$$

$$\text{subject to:} \quad Ax \geq b, \tag{2}$$

$$B_s x + D_s y_s \geq g_s \qquad \forall s \in S, \tag{3}$$

$$x \in \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}, \tag{4}$$

$$y_s \in \mathbb{Z}_+^{q_s} \times \mathbb{R}_+^{m_s - q_s} \qquad \forall s \in S. \tag{5}$$

The first stage consists of the decision variables x and constraints (2). The second stage comprises a set of disjoint scenarios, denoted by $s \in S$, that consist of the variables y_s and constraints (3). This problem is a mixed integer program (MIP) that consists of p integer variables in the first stage and q_s integer variables in each scenario s.

The decomposition approach developed by Benders [4] was originally applied to problems with only continuous variables in the second-stage, i.e. $q_s = 0 \; \forall s \in S$. Since its first introduction, many techniques have been developed to extend the capabilities of Benders' decomposition to solve general MIPs, such as the methods developed by Laporte and Louveaux [22], Carøe and Tind [11] and, more recently, Angulo et al. [2]. This paper is concerned with two problem classes of two stage MIPs to which Benders' decomposition can be applied: problems with continuous recourse, given by $q_s = 0 \; \forall s \in S$ and problems with a pure binary first-stage but a mixed integer second second stage, i.e. $x \in \{0,1\}^n$ and $q_s \geq 0 \; \forall s \in S$. While this paper focus on these two problem classes, the presented results are relevant in general to any application Benders' decomposition. To provide a brief introduction to Benders' decomposition, and the related algorithms, the discussion in the remainder of this section will focus on problems with a continuous second stage.

The application of Benders' decomposition results in the separation of the first and second stage variables and constraints into a master problem and a collection of subproblems. During the solution process, the subproblems takes as input a vector of values for the first-stage variables, which is a feasible solution to the first stage problem and is denoted as $\hat{x}$. The subproblems are then solved to evaluate the feasibility or optimality of the input first-stage solution. The subproblem for scenario s (SP-s) takes the form:

$$z_s(\hat{x}) = \min \quad d_s^\top y_s, \tag{6}$$

$$\text{subject to:} \quad D_s y_s \geq g_s - B_s \hat{x}, \tag{7}$$

$$y_s \in \mathbb{R}_+^{m_s}. \tag{8}$$

The dual of SP-s (SPd-s) is given by:

$$z_s(\hat{x}) = \max \quad w^\top (g_s - B_s\hat{x}), \tag{9}$$

$$\text{subject to:} \quad D_s^\top w \leq d_s, \tag{10}$$

$$w \in \mathbb{R}_+^{m_s'}, \tag{11}$$

where m_s' are the number of rows in D_s. The solution vector obtained by solving SPd-s is used to generate either an optimality or feasibility cut. An optimality cut is computed from an optimal solution to SPd-s—a dual extreme point of SP-s—and is added to the master problem as an underestimator of the optimal subproblem objective value. A feasibility cut is computed from an unbounded solution to SPd-s, which is a dual extreme ray of SP-s. This cut, when added to the master problem, eliminates $\hat{x}$ from the master problem, which has been verified as infeasible with respect to the subproblem constraints.

The Benders' decomposition master problem (BMP) is given by:

$$\min \quad c^\top x + \mathbf{1}\varphi, \tag{12}$$

$$\text{subject to:} \quad Ax \geq b, \tag{13}$$

$$\varphi_s \geq w(g_s - B_s x) \qquad \forall s \in S, \forall w \in \bar{\mathcal{P}}_s, \tag{14}$$

$$0 \geq w(g_s - B_s x) \qquad \forall s \in S, \forall w \in \bar{\mathcal{R}}_s, \tag{15}$$

$$x \in \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}, \tag{16}$$

$$\varphi \in \mathbb{R}^s, \tag{17}$$

where $\mathbf{1}$ is a vector of 1's of size $|S|$ and $\bar{\mathcal{P}}_s \subseteq \mathcal{P}_s$ and $\bar{\mathcal{R}}_s \subseteq \mathcal{R}_s$ are subsets of all dual extreme points and dual rays, respectively, of SP-s for scenario s. An additional variable, φ_s, is added to the mater problem for each scenario s that takes a value in the optimal solution that is a lower bound for SP-s. While there are a finite number of dual extreme points and rays of SP-s, the complete enumeration of $\mathcal{P}_s$ and $\mathcal{R}_s$ is practically impossible and would result in an intractable problem. This issue is addressed by initially formulating the BMP with $\bar{\mathcal{P}}_s = \emptyset$ and $\bar{\mathcal{R}}_s = \emptyset$ and then employing a constraint generation procedure to iteratively update the sets $\bar{\mathcal{P}}_s$ and $\bar{\mathcal{R}}_s$ until the optimal solution to the original problem is found.

The classical Benders' decomposition algorithm is described in Algorithm 1. There are two main phases of this algorithm. The first phase (lines 3 and 4) solves the BMP to find a candidate solution $(\hat{x}, \hat{\varphi})$ and set the lower bound to $LB \leftarrow c^\top \hat{x} + \mathbf{1}\hat{\varphi}_s$. The second phase (lines 6–17) solves SPd-s for each scenario s using $\hat{x}$ as input to generate optimality and feasibility cuts. Using the results from the second phase the upper bound is set either to infinity if one feasibility cut is generated or $UB \leftarrow c^\top \hat{x} + \sum_{s \in S} z_s(\hat{x})$ otherwise. The algorithm terminates when the difference between LB and UB is less than a predefined tolerance, given by ϵ.

Algorithm 1 Traditional Benders' decomposition algorithm

$UB \leftarrow \infty$, $LB \leftarrow -\infty$, $\bar{\mathcal{P}}_s \leftarrow \emptyset, \bar{\mathcal{R}}_s \leftarrow \emptyset, \varphi_s \leftarrow -\infty, \forall s \in S$.
while $UB - LB > \epsilon$ **do**
  solve MP, set $(\hat{x}, \hat{\varphi})$ to the solution of MP.
  $LB \leftarrow c^\top \hat{x} + \mathbf{1}\hat{\varphi}_s$.
  $UB \leftarrow c^\top \hat{x}$.
  for all $s \in S$ **do**
    solve SPd-s with $\hat{x}$ as input.
    if SPd-s is unbounded **then**
      add unbounded ray w of SPd-s to $\bar{\mathcal{R}}_s$.
      $UB \leftarrow \infty$.
    else
      $UB \leftarrow UB + z_s(\hat{x})$.
      if $z_s(\hat{x}) > \hat{\varphi}_s$ **then**
        add optimal solution w of SPd-s to $\bar{\mathcal{P}}_s$.
      end if
    end if
  end for
end while

1.1 Related work

Benders' decomposition, while theoretically appealing for many applications, has a number of shortcomings that impact its computational effectiveness in practice. As a result, various algorithm enhancement techniques have been developed. The Magnanti-Wong method [24]—an approach for strengthening optimality cuts—is one of the earliest examples of Benders' decomposition enhancement techniques. The cut strengthening approach developed by Magnanti and Wong [24] solves an auxiliary problem to find a cut that is closest to a given relative interior point, or core point. While the solution to the auxiliary problem helps to generate a strengthened optimality cut, solving this problem is a point of inefficiency for the technique. Papadakos [29] developed improvements to the Magnanti-Wong method that are specifically designed to address this inefficiency. The method proposed by Papadakos [29] involves using a core point in place of the master problem solution for some iterations of the Benders' decomposition algorithm. A cut enhancement technique similar to the Magnanti-Wong method is described by Calik et al. [10]. The technique of Calik et al. [10] is presented as a stabilisation procedure that uses a convex combination of an interior point and the current master problem solution as input for the Benders' decomposition subproblems. Finally, a detailed investigation of cut enhancement techniques for Benders' decomposition is presented by Petersen [31].

A commonly reported impediment to the convergence of the Benders' decomposition algorithm is the large distances between master problem solutions from consecutive iterations. Trust region methods have been proposed as a potential method for addressing this issue. An early example of a trust region approach for Benders' decomposition is presented by Ruszczyński [35]. The approach of Ruszczyński [35] 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 is applied. Santoso et al. [36] proposes a trust region approach, amongst many other enhancement techniques, that involves the addition of a linear constraint in the master problem. The additional constraint imposes a limit on the number of changes to the master problem solution between consecutive iterations. Alternatively, Maher et al. [25] minimises the Hamming distance between the previous and current master problem solutions while imposing an upper bound on the objective function value.

The heuristic use of Benders' decomposition is at the heart of many enhancement techniques. For example, the two- and three-phase methods presented by Cordeau et al. [12], Mercier et al. [27], Papadakos [30] and Froyland et al. [18] solve a relaxation of the original problem by Benders' decomposition in the first stage. An alternative heuristic approach is presented by Geoffrion and Graves [19] where the master problem is only solved to within a certain tolerance of the current upper bound in each iteration. In this case, it is demonstrated by Geoffrion and Graves [19] that valid Benders' cuts can be generated from suboptimal solution to the master problem. Finally, the trust region approaches can be viewed as large neighbourhood search heuristics where the neighbourhood is progressively increased until the optimal solution is found. These examples demonstrate that Benders' decomposition is particularly suited for use within heuristics and as a heuristic itself.

1.1.1 Benders' decomposition and large neighbourhood search

Large neighbourhood search heuristics are a key feature of many modern MIP solvers. Traditionally, the use of large neighbourhood search within a Benders' decomposition algorithm is limited. This is primarily due to the fact that MIP solvers are used as a black-box to solve the Benders' decomposition master and subproblems. Consequently, all primal heuristics can only applied separately to the master and subproblems. Applying primal heuristics to the master problem without any consideration of the subproblem constraints reduces the efficacy of the algorithms in finding good primal bounds. In fact, there is no guarantee that the primal solutions found by large neighbourhood search heuristics are feasible for the original problem. If the solutions are feasible, it is only possible to determine their quality by solving the subproblems after the execution of the algorithm: At which point there is no recourse to improve the solution quality. This limitation of large neighbourhood search heuristics will be addressed in this paper through a tighter integration with Benders' decomposition such that any improving solutions found are guaranteed to achieve an improvement in the upper bound for the original

problem.

The integration of large neighbourhood search heuristics and Benders' decomposition as a form of enhancement technique was first investigated by Rei et al. [33] with an adaptation of Local Branching [15]. The Benders' decomposition algorithm is augmented by phases of Local Branching in an effort to improve the incumbent solution. Additionally, the cuts generated during each Local Branching phase are stored and added to the original problem via the cutpool of CPLEX [21]. It is shown by Rei et al. [33] that employing Local Branching significantly improves the convergence of the Benders' decomposition algorithm.

More recently, the Proximity Search heuristic developed by Fischetti and Monaci [16] is applied in a Benders' decomposition context by Boland et al. [9]. Similar to Proximity Search [16], each iteration of Proximity Benders' [9] 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 heuristic behaviour of Benders' decomposition has been shown to be greatly improved with the integration of Proximity Search.

Many other enhancement techniques for Benders' decomposition employing objective function constraints and proximity functions have been previously investigated. For an application of recoverable robustness, Maher et al. [25] describes an enhancement technique that uses a proximity function and an upper bounding constraint that greatly improves the ability to solve the considered problems. Further, Proximity Benders' draws upon ideas from enhancement techniques that employ a trust region, such as the approaches by Ruszczyński [35] and Santoso et al. [36]. The performance improvements achieved by employing trust region approaches and Proximity Benders' demonstrate the value of this type of enhancement technique.

1.2 Contributions and paper structure

The limited use of large neighbourhood search heuristics with Benders' decomposition has been previously addressed in bespoke implementations integrating the two. To the best of the authors knowledge Rei et al. [33] and Boland et al. [9] are the only examples that consider the enhancement of Benders' decomposition through the use of large neighbourhood search. The work presented in this paper generalises this enhancement technique through the development of an algorithmic framework for the integration of large neighbourhood search heuristics and Benders' decomposition. Further, this general algorithmic framework also generalises the use of trust regions for Benders' decomposition as an enhancement approach. To this end, the work presented in this paper will provide a general enhancement approach for the Benders' decomposition algorithm.

The contributions of this paper are:

- The formal presentation of a general enhancement technique derived from the improved use of large neighbourhood search heuristics within the Benders' decomposition algorithm—the large

neighbourhood Benders' search.

- The development of a computational framework integrating Benders' decomposition with large neighbourhood search heuristics.
- A detailed assessment of the potential enhancements that can be achieved by integrating Benders' decomposition and large neighbourhood search heuristics.

This paper is structured as follows. The proposed enhancement technique, the large neighbourhood Benders' search, will be presented in Section 2. A major component of this work is the implementation of the large neighbourhood Benders' search within the constraint integer programming solver SCIP [26]. The specific details regarding the implementation are presented in Section 3. The results from computational experiments to assess the enhancement potential of the large neighbourhood Benders' search will be presented in Section 4. Finally, Section 5 will report on the conclusions and provide some future research directions.

2 Large neighbourhood Benders' search

The general algorithm of large neighbourhood search heuristics involves i) forming an auxiliary problem with a feasible region, the neighbourhood, that is a restriction of the original problem and ii) solving the auxiliary problem to identify primal solutions with improved bounds. The method by which the neighbourhood is constructed is a critical component of the algorithm. Ideally this neighbourhood contains improving solutions that are easy to find when solving an auxiliary problem. The strength of a large neighbourhood search heuristic lies in the quality of the solutions contained in the neighbourhood, how difficult they are to find and the computational difficulty of the resulting auxiliary problem.

Consider the BMP given by (12)–(17). We denote the feasible region of BMP by $X := \{(x, \varphi) \,|\, (13)\text{–}(17)\}$, so that the master problem can be written as

$$\min\{c^\top x + \mathbf{1}\varphi \,|\, (x, \varphi) \in X\}. \tag{18}$$

For large neighbourhood search heuristics, the restrictions on the feasible region are typically made to ensure that the neighbourhood is small while all feasible solutions to the auxiliary problem are also feasible for the original problem. The auxiliary problem, with a feasible region denoted by $\hat{X} \subseteq X$, is given by

$$\min\{f(x) + \mathbf{1}\varphi \,|\, (x, \varphi) \in \hat{X}\}, \tag{19}$$

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 [16].

When solving (19), an optimal x and φ is found with respect to a fixed set of Benders' cuts given by $\bar{\mathcal{P}}_s$ and $\bar{\mathcal{R}}_s$. Since $\bar{\mathcal{P}}_s$ and $\bar{\mathcal{R}}_s$ are only subsets of $\mathcal{P}_s$ and $\mathcal{R}_s$, the solution $(\hat{x}, \hat{\varphi})$ is only valid for a

relaxation of the original problem. Hence, there is no guarantee that $(\hat{x}, \hat{\varphi})$ is a primal feasible solution, let alone an improving solution.

Given a solution $\hat{x}$ to (19), the upper bound on the original problem is computed by

$$UB(\hat{x}) = \begin{cases} c^\top \hat{x} + \sum_{s \in S} z_s(\hat{x}) & \text{if SP-}s \text{ is feasible } \forall s \in S, \\ \infty & \text{otherwise,} \end{cases} \tag{20}$$

where $z_s(\hat{x})$ is the objective function value of SP-s when solved using $\hat{x}$ as input.

Since $\hat{X}$ represents a restriction on the feasible region of the original problem, the second stage constraints, denoted by $Y := \{(x, y_1, \ldots, y_{|S|}) \,|\, B_s x + D_s y_s \geq g_s;\, y_s \in \mathbb{Z}_+^{q_s} \times \mathbb{R}_+^{m_s - q_s}\}$, are still valid for the auxiliary problem. Given solution $\hat{x}$ to (19), it is possible to compute an upper bound for the auxiliary problem when considering the second stage constraints Y by solving SP-s $\forall s \in S$. Such an upper bound is given by

$$UB_{aux}(\hat{x}) = \begin{cases} f(\hat{x}) + \sum_{s \in S} z_s(\hat{x}) & \text{if SP-}s \text{ is feasible } \forall s \in S, \\ \infty & \text{otherwise.} \end{cases} \tag{21}$$

Considering (20) and (21), there are two possible methods for solving the auxiliary problem of large neighbourhood search heuristics. The classical approach is to solve (19) directly without considering the constraints that have been transferred from the original problem to the subproblems. The second stage constraints are only considered at the termination of the large neighbourhood search 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.

The alternative method, the large neighbourhood Benders' search, is to employ Benders' decomposition to enforce the subproblem constraints in the auxiliary problem. Consider the sets of dual extreme points and rays given by $\hat{\mathcal{P}}_s \subseteq \mathcal{P}_s \backslash \bar{\mathcal{P}}_s$ and $\hat{\mathcal{R}}_s \subseteq \mathcal{R}_s \backslash \bar{\mathcal{R}}_s$ respectively and denote the set of solutions satisfying the Benders' cuts computed using $\hat{\mathcal{P}}_s$ and $\hat{\mathcal{R}}_s$ as $\hat{Y} := \{(x, \varphi) \,|\, \varphi_s \geq w(g_s - D_s x), \forall s \in S, \forall w \in \hat{\mathcal{P}}_s;\, \varphi_s \geq w(g_s - D_s x), \forall s \in S, \forall w \in \hat{\mathcal{R}}_s\}$. When employing the large neighbourhood Benders' search the auxiliary problem of the large neighbourhood search heuristics is of the form

$$\min\{f(x) + \mathbf{1}\varphi \,|\, (x, \varphi) \in \hat{X} \cap \hat{Y}\}. \tag{22}$$

Since the sets $\hat{\mathcal{P}}_s$ and $\hat{\mathcal{R}}_s$ are prohibitively large, a constraint generation procedure, i.e. Benders' decomposition, is required to solve (22). In the following, this alternative method will be described as solving the auxiliary problem by Benders' decomposition.

Proposition 1. *Solution $\hat{x}$ is feasible for* (22) *if and only if SP-s, when solved with $\hat{x}$ as input, is feasible for all $s \in S$.*

Proof. This is a condition of solving (22) by Benders' decomposition. □

When solving (19) by branch-and-bound, the solutions found are not verified for feasibility or optimality with respect to the second-stage constraints until then end of the large neighbourhood search heuristic algorithm. Thus, it is possible that through this verification the optimal solution to (19) is deemed infeasible. More commonly, since the verification is performed at the completion of the large neighbourhood search heuristic algorithm, the optimal solution may be sub-optimal with respect to the second-stage constraints. Solving (22) by Benders' decomposition executes the feasibility verification during the large neighbourhood search heuristic algorithm. Therefore, when employing large neighbourhood Benders' search all feasible solutions to the auxiliary problem satisfy all second stage constraints, as stated by Proposition 1. This feature is a major advantage of the large neighbourhood Benders' search compared to the classical use of large neighbourhood search heuristics.

Proposition 2. *All solutions feasible for* (22) *are feasible for the original problem.*

Proof. Assume that when using the large neighbourhood Benders' search the solution $\hat{x}$ is a feasible for (22) but is not feasible for the original problem. Since the neighbourhood of (22), given by $\hat{X}$, is contained within the feasible region of the Benders' decomposition master problem, then $\hat{x} \in X$. Thus, $\hat{x}$ can only be infeasible for the original problem if it violates constraints contained in $\hat{Y}$.

The verification of the constraints contained in $\hat{Y}$ is performed by solving SP-s for all $s \in S$ given $\hat{x}$ as input. As a condition of Benders' decomposition, a constraint from $\hat{Y}$ is violated if and only if there exists an $s \in S$ where SP-s is infeasible. Since $\hat{x}$ is feasible for (22), then by Proposition 1 SP-s is feasible for all $s \in S$. This is a contradiction, so when employing large neighbourhood Benders' search all solutions that are feasible for (22) must also be feasible for the original problem. □

The large neighbourhood Benders' search improves the use of large neighbourhood search heuristics for Benders' decomposition by ensuring that the optimal solutions to the auxiliary problem (22) are feasible with respect to the original problem. This particular feature of the large neighbourhood Benders' search is an improvement over the traditional use of large neighbourhood search heuristics where feasibility with respect to the original problem is not guaranteed. In addition to ensuring that all feasible solutions to the auxiliary problem are feasible for the original problem, the large neighbourhood Benders' search improves the quality of the primal bounds found by large neighbourhood search heuristics. This is explained by the following lemma.

Lemma 1. *If x' and x'' are the optimal solutions to* (19) *and* (22) *respectively, then*

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

Proof. Assume that $UB_{aux}(x') < UB_{aux}(x'')$. Lets consider the large neighbourhood Benders' search algorithm, i.e. (22) is used as the auxiliary problem of the large neighbourhood search heuristics in place

of (19). In the first iteration of the Benders' decomposition algorithm, since no Benders' cuts have been generated the feasible region of (22) is $\hat{X}$, which is identical to the feasible region of (19). As such, the solution found in the first iteration of the Benders' decomposition algorithm to solve (22) is x'. Now, the Benders' decomposition algorithm terminates when the condition $UB_{aux}(x) \leq LB_{aux}(x, \varphi) = f(x) + \mathbf{1}\varphi$ is satisfied, where φ is given by the solution to (22). This presents two cases: i) $UB_{aux}(x') \leq LB_{aux}(x', \varphi)$ and ii) $UB_{aux}(x') > LB_{aux}(x', \varphi)$.

i) in this case the termination condition for Benders' decomposition has been satisfied. Thus, $x'' = x'$ and $UB_{aux}(x') = UB_{aux}(x'')$, which is a contradiction to the original assumption.

ii) while x' is not the optimal solution to (22), $UB_{aux}(x')$ is still a valid upper bound. The optimal objective value to (22), denoted by z, lies in the range $LB_{aux}(x') \leq z \leq UB_{aux}(x')$. Since x'' is the optimal solution to (22) when solved by Benders' decomposition, $z = UB_{aux}(x'') \Rightarrow UB_{aux}(x'') \leq UB_{aux}(x')$, which is a contradiction to the original assumption.

Since both cases contradict the original assumption, then $UB_{aux}(x') \geq UB_{aux}(x'')$. □

In addition to improving the quality of the solutions found by the large neighbourhood Benders' search, with respect to $f(x)$, it is also possible to achieve improved bounds for the original problem. In cases where $f(x) = c^\top x$ in (19) and (22), i.e. the original objective function is the objective of the auxiliary problem, it can be shown that the large neighbourhood Benders' search achieves improved primal bounds. This is demonstrated in the following theorem.

Theorem 1. *Let $f(x) = c^\top x$ in* (19) *and* (22)*. If x' and x'' are the optimal solutions to* (19) *and* (22) *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 (20) and (21) 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 large neighbourhood search 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 large neighbourhood search heuristics, the large neighbourhood Benders' search will achieve an upper bound at least as good as that achieved when solving the auxiliary problem by branch-and-bound.

2.1 Additional features

Stabilisation. Since the large neighbourhood Benders' search shares characteristics with trust region approaches, the proposed algorithm can be viewed as a stabilisation technique for the generation of cuts. During the large neighbourhood Benders' search, the cuts are generated using solutions found within a restricted domain of the original problem. Therefore, the distance between consecutive solutions when solving the auxiliary problem by Benders' decomposition is potentially smaller than when solving the original problem—especially when employing large neighbourhood Benders' search with Proximity Search. This leads to more cuts being generated in specific neighbourhood of the feasible region. It is expected that the inherent stabilisation afforded by the large neighbourhood search heuristics will help improve the dual bound for the original problem

An additional feature of the large neighbourhood Benders' search is that all solutions used to generate cuts are in the interior of the original polyhedron. Many studies have shown that interior solutions can generate stronger optimality cuts [17, 24, 30]. Thus, it is possible to generate enhanced Benders' cuts using large neighbourhood Benders' search.

Transfer of cuts. During the large neighbourhood Benders' search many cuts are generated while solving (22). It is stated by Geoffrion and Graves [19] than any feasible solution to the Benders' decomposition master problem can be used to generate cuts. Since all feasible solutions to the auxiliary problem are also feasible for the master problem, every cut generated during the large neighbourhood Benders' search is valid for the original problem. The transfer of cuts is achieved by updating the sets $\bar{\mathcal{P}}$ and $\bar{\mathcal{R}}$ by $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \hat{\mathcal{P}}'$ and $\bar{\mathcal{R}} \leftarrow \bar{\mathcal{R}} \cup \hat{\mathcal{R}}'$, where $\hat{\mathcal{P}}' \subseteq \hat{\mathcal{P}}$ and $\hat{\mathcal{R}}' \subseteq \hat{\mathcal{R}}$ are the dual extreme points and rays used to generate cuts for the auxiliary problem respectively. This results in a greater number of cuts in the Benders' decomposition master problem that could potentially improve the lower bound. This property is exploited by Rei et al. [33] where the cuts generated during the Local Branching phases are transferred to the original problem.

3 Implementation

There are many variants of the Benders' decomposition algorithm, each of which can employ the large neighbourhood Benders' search. The main difference between the various algorithms are the points at which the subproblems are solved to generate Benders' cuts. Algorithm 1 is an example of the traditional variant of the Benders' decomposition algorithm. This algorithm is characterised by the complete disconnect between solving the master problem and the subproblems. Within this variant the master problem is typically solved to optimality before the subproblems are solved to generate cuts. Branch-and-check [37] is an alternative algorithm that arose in response to a greater integration of constraint programming and mixed integer programming techniques. A major advantage of the

branch-and-check algorithm over Algorithm 1 is that it draws upon much more of the underlying solver technology. In particular, the subproblems are solved during the enforcement of constraints for LP solutions or when candidate primal solutions are checked for feasibility. During each of these stages Benders' cuts can be generated from the LP and primal feasible solutions respectively.

Large neighbourhood Benders' search exploits many features of the branch-and-check algorithm. Through the use of an internal Benders' decomposition framework for SCIP, the large neighbourhood Benders' search is implemented by adapting the set up process of the large neighbourhood search heuristics. As observed from (18) and (19), there is little difference between the Benders' decomposition master problem and the auxiliary problem. Similarly, the initial formulation of (22) is identical to (19). However, (22) implicitly considers all second stage constraints through a constraint generation process, which is the same algorithm used to solve (18). The main difference between (18) and (22) comes from the set $\hat{X}$ being a subset of X. Internally within a MIP solver, the same algorithmic framework available to solve (18) is available for solving the auxiliary problem of large neighbourhood search heuristics, albeit with changes to some parameter settings. Thus, if branch-and-bound or branch-and-check is used to solve the Benders' decomposition master problem, then they can be easily applied to solve the auxiliary problems of (19) or (22) respectively.

A generic Benders' decomposition framework within SCIP has been developed for the implementation of the large neighbourhood Benders' search. The user is able to interact with the generic Benders' decomposition framework in three different ways. The first is by providing SCIP with an instance in the SMPS format [7]. By default, an instances provided as an SMPS file to SCIP is solved as the deterministic equivalent, but an automatic Benders' decomposition process is available. The second method of interaction is by developing a *custom* solver using SCIP and invoking the default Benders' decomposition plugin. The user calls the default Benders' decomposition plugin by providing the calling function with a SCIP instance of the master problem and an array of SCIP instances corresponding to the subproblems. This approach automatically generates a mapping between the master and subproblem variables that is required for the subproblem setup and cut generation processes. The final method for using the generic Benders' decomposition framework is by developing a custom Benders' decomposition plugin. This approach requires the user to specify the form of the subproblems, how to solve each subproblem and the variable mapping between the master and subproblems.

During the execution of a large neighbourhood search heuristic, a copying process is performed to construct the auxiliary problem. The auxiliary problem is called a sub-MIP since it comprises many features of the original MIP. Within SCIP, each plugin type, such as heuristics, separators and presolvers, provides a callback function that is invoked during the copying process of a SCIP instance. This callback is typically used to copy all relevant data structures and make the appropriate plugins available to the target sub-MIP. Initially, the auxiliary problem sub-MIP is constructed by copying the Benders' decomposition master problem and then making the necessary feasible region restrictions as given by

the neighbourhood definition. The resulting sub-MIP could then be solved by branch-and-bound, as in the traditional use of large neighbourhood search heuristics for Benders' decomposition. For the large neighbourhood Benders' search, Benders' decomposition is used to solve the auxiliary problem by making the Benders' decomposition data structures and callback functions available to the sub-MIP through the copying process. The copying mechanism of SCIP has been extended and a copy callback included in the Benders' decomposition plugins to achieve this. In the case of the default Benders' plugin, the copying process simply creates the Benders' decomposition data structures by calling the creation function with the copied master problem instance and the original subproblem instances as input parameters. It is important to note that it is not necessary to copy the subproblem SCIP instances since they are unmodified when the neighbourhood for the master problem is created. The copying process creates a new Benders' decomposition data structure for the sub-MIP, using the copied master problem and the original subproblems, and ensures that the Benders' cut generating plugins are available.

The Benders' decomposition algorithm used to solve the auxiliary problem sub-MIP differs from that employed to solve the original problem. Specifically, when the subproblem solving process is called to generate cuts, only the LP relaxations of the subproblems are solved. In the classical application of Benders' decomposition—where the subproblems are linear programs—this modification has no impact on the algorithm behaviour. In the case where the subproblems are MIPs, the integer cuts are not generated while solving the large neighbourhood search auxiliary problem to optimality, only the LP cuts are generated. This algorithmic change is an attempt to reduce the run time devoted to subproblem solving during the large neighbourhood Benders' search.

3.1 Employing aggressive heuristics settings

The large neighbourhood Benders' search is proposed as a general enhancement technique for the Benders' decomposition algorithm. Additionally, the large neighbourhood Benders' search can be employed as as aggressive heuristic approach. The use of large neighbourhood search to improve the heuristic behaviour of Benders' decomposition has been discussed by Rei et al. [33] and Boland et al. [9]. To further extend this idea, aggressive settings for a subset of the large neighbourhood search heuristics available within SCIP have been created to evaluate the heuristic behaviour of the large neighbourhood Benders' search. The primary feature of the aggressive settings is that the selected the large neighbourhood search heuristics are executed after every LP solve. Typically large neighbourhood search heuristics are executed after the node processing has been completed, thus with this modification to the execution timing the selected heuristics will be called more often. Since this aggressive setting results in many more calls to the large neighbourhood search heuristic, the sub-MIP run time has a large impact on the overall solution time. This issue has been addressed by imposing additional working limits for the execution of the aggressive large neighbourhood search heuristics. Aggressive settings have been created for following heuristics for an additional set of experiments evaluating the heuristic behaviour

of the large neighbourhood Benders' search.

Crossover. Concepts from genetic algorithms motivated the development of Crossover [34]. More than one feasible solution is required by Crossover to identify variable fixings and define the neighbourhood. Variables are fixed if their solution values are the same in more than one of the considered primal feasible solutions. The aggressive implementation applies some changes to the calling frequency of this heuristic. Specifically, Crossover selects the next node where it will be applied based upon the number of failures of this algorithm—defined as times when the heuristic is unable to find a solution. For these experiments, while the number of failures is less than 10, then Crossover will continue to be executed at the current nodes. Once the number of failures exceeds 10, but is still less than 25, the next node is given by the sum of the current number of nodes and 10 times the number of failures. Finally, if the number of failures exceeds 25, then Crossover will no longer be called.

DINS. Distance induced neighbourhood search [20] is an improvement heuristics that draws upon the concepts from Local Branching [15], Crossover [34], RENS [6] and RINS [13]. A main feature of DINS is that it searches for primal solutions that are closer to the relaxation optima than the current incumbent. The implementation of DINS for the aggressive large neighbourhood Benders' search sets the gap limit to 10%.

Local branching. Local branching [15] is one of the earliest examples of large neighbourhood search primal heuristics for MIP. The neighbourhood for Local Branching is all solutions that are within a specified distance of the current incumbent. The distance function for Local Branching is the Manhattan distance. The Local Branching concept has been adapted for use with Benders' decomposition by Rei et al. [33]. No changes to the working limits have been made for the aggressive implementation.

Proximity Search. Proposed by Fischetti and Monaci [16], Proximity Search was later extended for use with Benders' decomposition by Boland et al. [9]. The main idea of Proximity Search is to identify primal solutions that are close to the current incumbent and provide a primal bound improvement. Specifically, the objective function is replaced with a proximity function and an objective constraint is added to reduce the neighbourhood size. The working limits of Proximity Search are modified to restrict the number of improving solutions found to 3 and the gap limit is set to 10%.

RINS. Relaxation induced neighbourhood search [13] is an improvement heuristic that fixes variables using the solution to the LP relaxation and the current incumbent. The variables fixed to form the RINS neighbourhood are those that take the same value in the two reference solutions. The additional working limit imposed for the aggressive implementation is a gap limit of 10%.

3.2 Parameter settings

The large neighbourhood search heuristics available in SCIP have numerous parameters that when modified can alter the algorithm behaviour. The default parameter settings that are provided within SCIP have been selected to provide the best performance on benchmark MIP instances. These default setting—with presolving and propagation deactivated—will be used when evaluating the general enhancement achieved by employing the large neighbourhood Benders' search.

In regards to the aggressive use of large neighbourhood search heuristics, the default heuristic settings are not suitable. This is primarily due to the fact that large neighbourhood search heuristics are computationally expensive and hence their frequent execution is not typically desired. For the experiments aggressively using large neighbourhood search heuristics the parameter settings have been modified to ensure that the heuristics are executed during the early stages of the computation. While this may not be the most satisfactory setting for heuristics that require multiple primal solutions, such as Crossover, the same behaviour is demanded from each large neighbourhood search heuristic to maintain consistency in the aggressive use experiments. Additionally, for each of the heuristics listed above, separate settings are used that deactivate all other large neighbourhood search heuristics except the one of interest: All other heuristics are executed as per default. The parameter settings used for the related experiments are provided in the appendix.

4 Computational experiments

The performance of the large neighbourhood Benders' search is evaluated using three stochastic programming problems: the capacitated facility location problem (SCFLP), the network interdiction problem (SNIP) and the multiple knapsack problem (SMKP). The experiments performed using these test sets will be evaluated using the primal integral comparison metric, which is described by Berthold [5]. For all experiments the results are aggregated using the shifted geometric mean. A shift of 10 is used for the primal integral and a shift of 1 is used for the final gap. In addition, performance profiles [14] are used to provide a visual analysis of the evaluated solution algorithms.

The Benders' decomposition framework and large neighbourhood Benders' search have been implemented within a development version of SCIP 4.0 [26]. All computational experiment have been performed using SoPlex 3.0 as LP solver. The computational experiments have been conducted using a cluster consisting of Intel(R) Xeon(R) CPU E5-2670 2.5GHz processors with 64 GB ram.

4.1 Stochastic Capacitated Facility Location Problem

The SCFLP formulation is adapted from the model developed by Loveaux [23]. The stochastic variant of this problem involves selecting the set of facilities in the first stage and determining what facilities service each customer in the second stage. 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 current experiments:

$$\begin{aligned}
\min \quad & \sum_{i\in I} f_i x_i + \frac{1}{|S|}\sum_{s\in S}\sum_{i\in I}\sum_{j\in J} q_{ij} y_{ij}^s, && \\
\text{subject to} \quad & \sum_{i\in I} y_{ij}^s \geq \lambda_j^k && \forall j \in K, \forall s \in S, \\
& \sum_{j\in J} y_{ij}^s \leq k_i x_i && \forall i \in I, \forall s \in S, \\
& \sum_{i\in I} k_i x_i \geq \max_{s\in S} \sum_{j\in J} \lambda_j^s, && \\
& x_i \in \{0,1\} && \forall i \in I, \\
& y_{ij}^s \geq 0 && \forall i \in I, \forall j \in J, \forall s \in S.
\end{aligned} \tag{23}$$

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 variables x_i equal 1 if facility i is opened, which has a cost of f_i, and 0 otherwise. If customer j is serviced by facility i in scenario s, the variable y_{ij}^s equal 1, which has a cost of q_{ij}, and 0 otherwise.

The data for the deterministic capacitated facility location problem has been collected from the CAP instances of the OR-Library [3]. For this paper, only the instance with 25 and 50 customers are used. Taking these deterministic instances, the stochastic variant is constructed using the method described by Bodour et al. [8]. 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 $[0.1\lambda, 0.3\lambda]$. Stochastic instances consisting of 250 and 500 scenarios have been generated for these experiments.

4.2 Stochastic network interdiction problem

The SNIP, as introduced by Pan and Moreton [28], is solved to identify a set of arcs that are used to position sensors in order to maximise the probability of catching an intruder. This problem consists of a graph G comprised by a set of nodes N and directed arcs A, of which the subset D can be used to position sensors. The interdictor knows the probability of an intruder being detected on arc (i, j) with and without sensors, which is denoted by r_{ij} and q_{ij} respectively. Only the arcs $(i, j) \in D$ can be used to install a sensor, with a maximum number of sensors being restricted to b. The cost of installing a sensor is given by c_{ij}. The scenarios of the stochastic program are contained in the set K and each is described by a source and a sink node, s^k and t^k respectively, that the intruder will use. Each scenario has a probability of realisation given by p^k. In order to reduce numerical instability within the solver, the objective function coefficients have been multiplied by 1000. The subproblem for scenario k identifies the path from s^k to t^k that has the greatest probability of being traversed by the intruder undetected. The final parameter, ψ_j^k, is the value of the maximum-reliability path from j to t^k if no sensors were

placed on the network. The value of this parameter can be easily computed by solving a shortest path problem.

The formulation of the SNIP, as presented by Bodour et al. [8], is

$$
\begin{aligned}
\min \quad & \sum_{k \in K} p^k \pi^k_{s^k}, \\
\text{subject to} \quad & \sum_{(i,j) \in D} c_{ij} x_{ij} \leq b, \\
& \pi^k_{t^k} = 1 && \forall k \in K, \\
& \pi^k_i - q_{ij} \pi^k_j \geq 0 && \forall (i,j) \in D, \forall k \in K, \\
& \pi^k_i - r_{ij} \pi^k_j \geq 0 && \forall (i,j) \in A \backslash D, \forall k \in K, \\
& \pi^k_i - r_{ij} \pi^k_j \geq -(r_{ij} - q_{ij}) \psi^k_j x_{ij} && \forall (i,j) \in D, \forall k \in K, \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in D, \\
& \pi^k_i \geq 0 && \forall i \in N, \forall k \in K.
\end{aligned} \tag{24}
$$

The binary variable x_{ij} equals 1 to indicate a sensor is placed of arc (i,j) and 0 otherwise and the continuous variable π^k_i is the probability that an intruder can travel from i to t^k undetected. The first constraint imposes an upper bound on the number of sensors that can be installed by the interdictor. The probability of the intruder passing through the network undetected is given by the remaining constraints.

The SNIP instances presented by Bodour et al. [8] are used for the computational experiments. The collected instances consist of 456 scenarios and the network comprises 783 nodes and 2586 arcs. There are 320 possible locations for sensors, as such the first stage consists of 320 binary variables.

Four classes of instances are proposed by Pan and Morton [28]. Each of the classes are differentiated by the method for setting the probability parameters of q_{ij} and r_{ij} for each arc $(i,j) \in D$. The results presented by Pan and Morton [28] and Bodour et al. [8] indicate that parameter settings for classes 3 and 4 yield the most difficult instances. In both classes the value of r_{ij} is sampled from a uniform distribution ranging between $[0.3, 0.6]$. However, $q_{ij} = 0.1 r_{ij}$ for class 3 and $q_{ij} = 0$ for class 4. Within each instance class, five instances are generated for each problem formulated with a budget of {30, 40, 50, 60, 70, 80, 90}, making a total of 70 instances.

4.3 Stochastic binary multiple knapsack problem

The stochastic binary multiple knapsack problem (SMKP) consists of two decision stages with a set of knapsacks in each of the stages. The first stage contains two sets of items, given by I and J, and the second stage involve a single set of items given by K. The multiple knapsacks are denoted by M and N in the first and second stages respectively. Each knapsack has a weight, which is denoted by W_m for knapsack m in the first stage and T_n for knapsack n in the second stage. The uncertainty of the second stage is realised in the costs of the second stage items, which is given by a set of scenarios denoted by

S. Using the above notation, an SMKP instance is given by the following:

$$\begin{aligned}
\min \quad & \sum_{i\in I} c_i x_i + \sum_{j\in J} d_j y_j + \sum_{s\in S}\sum_{k\in K} g_k^s z_k^s, & \\
\text{subject to} \quad & \sum_{i\in I} a_i^m x_i + \sum_{j\in J} b_j^m y_j \geq W_m & \forall m \in M, \\
& \sum_{i\in I} p_i^n x_i + \sum_{k\in K} q_k^n z_k^s \geq T_n & \forall s \in S, \forall n \in N, \\
& x_i \in \{0,1\} & \forall i \in I, \\
& y_j \in \{0,1\} & \forall j \in J, \\
& z_k^s \in \{0,1\} & \forall s \in S, \forall k \in K.
\end{aligned} \tag{25}$$

The binary variables x_i indicates if item i is selected in the first stage at a cost of c_i. Similarly, y_j indicates if item j is selected in the first stage at a cost of d_j. In the second stage, for each scenario s the binary variable z_k^s indicate whether item k is selected at a cost of q_k^s. In the first stage, for each knapsack m contained in M, the items have weights denoted by a_i^m and b_j^m for items i and j respectively. In the second stage, for each knapsack n contained in N, the weights for items i and k denoted by p_i^n and q_j^n respectively.

The SMKP instances have been collected from the SIPLIB [1]. A description of the instances, including the method by which they were generated, is provided by Angulo et al. [2].

4.4 Using default settings

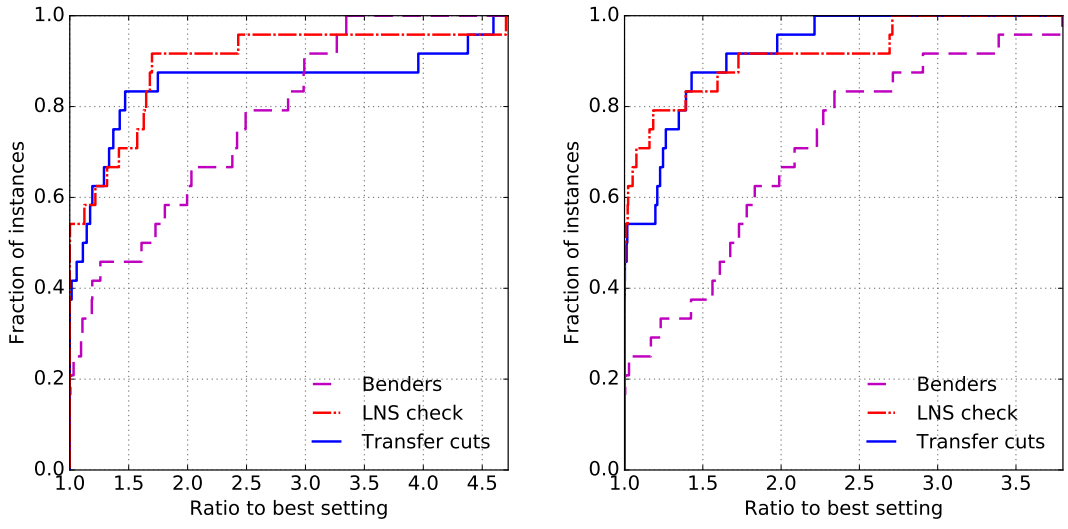
The first set of experiments are conducted to evaluate the performance of the large neighbourhood Benders' search using the default settings for SCIP. The base comparison algorithm for this set of experiments is Benders' decomposition where the auxiliary problem of the large neighbourhood search heuristics are solved by branch-and-bound, i.e. solving (19). In all reported results, this base comparison algorithm will be labelled as *Benders*. Two different large neighbourhood Benders' search algorithms are being evaluated in these experiments. The first is the default large neighbourhood Benders' search, where Benders' decomposition is employed to solve the auxiliary problem of the large neighbourhood search heuristics, which is given by (22). This algorithm is labelled as *LNS check*. The second is a variation on the large neighbourhood Benders' search where the cuts generated while solving the auxiliary problem using Benders' decomposition are transferred to the original master problem. In this algorithm, at the termination of the large neighbourhood search heuristic the dual extreme point and rays in the master problem are updated by $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \hat{\mathcal{P}}'$ and $\bar{\mathcal{R}} \leftarrow \bar{\mathcal{R}} \cup \hat{\mathcal{R}}'$ respectively. The computational experiments performed to evaluate the transfer of cuts within the large neighbourhood Benders' search are labelled *Transfer cuts*.

4.4.1 Stochastic Capacitated Facility Location Problem

The SCFLP instances have been separated into two groups based upon the number of scenarios. This classification is used as a proxy for their difficulty. Also, this classification will aid in assessing whether the increase in the number of scenarios has a negative impact on the performance of the large neighbourhood Benders' search.

The performance of the large neighbourhood Benders' search with respect to the primal integral is shown in Figure 1. This figure shows that the large neighbourhood Benders' search significantly improves the performance of the Benders' decomposition algorithm to solve the SCFLP instances. When solving the instances with 250 scenarios, all expect 1 and 3 instances, for *LNS check* and *Transfer cuts* respectively, exhibit good performance with respect to the best setting. Also, for the majority of instances *LNS check* and *Transfer cuts* achieve a smaller primal integral than *Benders*, specifically 14 and 15 instances respectively.

The results for the instances with 500 scenarios show a much larger performance improvement when employing the large neighbourhood Benders' search compared to the instances with 250 scenarios. Figure 1(b) shows that both *LNS check* and *Transfer cuts* dominate *Benders* for this collection of test instances. It is clear from the results presented in Figure 1 that the increase in the number of scenarios does not impact the performance of the large neighbourhood Benders' search when solving the SCFLP instances. A possible explanation is that as the number of scenarios increases, the difficulty in finding a good feasible solution also increases. A particular strength of the large neighbourhood Benders' search is that it is very effective in finding good primal solutions early during the solution process for difficult



(a) 250 Scenarios

(b) 500 Scenarios

Figure 1: The performance profile of the primal integral for the SCFLP instances comparing different Benders' decomposition settings using default settings for SCIP.

Table 1: The shifted geometric mean of the primal integral and final gap for the SCFLP instances

Scenarios	Num	∅ Primal integral			∅ Final gap		
and Facilities	Inst	*Benders*	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
250-25	12	4013.94	3727.67	**3615.09**	15.02	15.3	**15.01**
250-50	12	11199.06	**7426.51**	9000.56	24.96	**23.08**	23.52
250-all	24	6705.99	**5262.12**	5705.25	19.39	18.81	**18.81**
500-25	12	5754.31	**3685.23**	3848.32	16.18	**15.54**	15.57
500-50	12	10443.94	7958.16	**7751.34**	24.32	**23.35**	23.64
500-all	24	7752.72	**5416.25**	5462.27	19.85	**19.07**	19.2

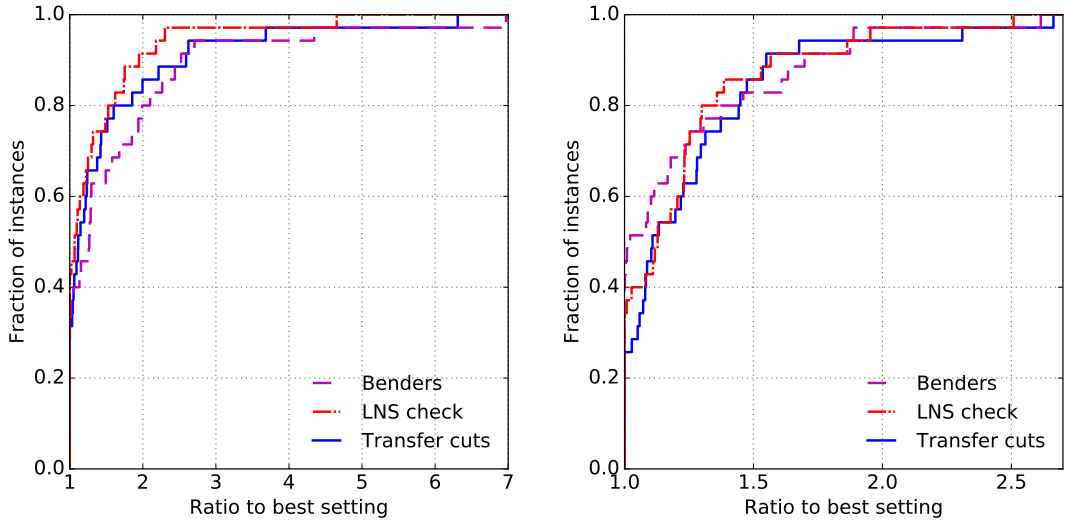
problems.

The aggregated results for the primal integral and final gap are presented in Table 1. These results highlight the extent to which employing the large neighbourhood Benders' search reduces the primal integral. For the 250 scenario instances, *LNS check* and *Transfer cuts* achieve a reduction in the primal integral of 22.96% and 18.61% respectively. This reduction in the primal integral appears to be consistent across all instances. Interestingly, a larger reduction in the primal integral is observed when considering only the instances with 50 facilities—35.4% and 24.14% for *LNS check* and *Transfer cuts* respectively. Unfortunately this reduction in the primal integral is not mirrored by a significant change to the final gap. Specifically, the geometric mean of the final gap achieved by *LNS check* and *Transfer cuts* over all of the instances with 250 scenarios is within 1% of that reported by *Benders*.

Similar to the results presented in Figure 1(b), Table 1 shows that the 500 scenario instances achieve an even greater average reduction in the primal integral by employing the large neighbourhood Benders' search compared to the 250 scenario instances. An average reduction in the primal integral of 30.1% and 29.5% compared to *Benders* is achieved by *LNS check* and *Transfer cuts* respectively. While *Transfer cuts* is shown to achieve performance improvement compared to *Benders*, this does not appear to be significant when compared to *LNS check*. Additionally, *Transfer cuts* does not achieve any improvement in the average final gap, which is a desired outcome from employing this algorithm. Thus, the results from the SCFLP instances suggest that *LNS check* is the superior large neighbourhood Benders' search algorithm.

4.4.2 Stochastic network interdiction problem

The SNIP instances display varying degrees of difficulty across the complete test set. Figure 2 shows that the relative performance of the large neighbourhood Benders' search with respect to the primal integral, is different for the class 3 and 4 instances. For the class 3 instances, both *Transfer cuts* and *LNS check* are shown in Figure 2(a) to achieve an improvement in the primal integral compared to *Benders*. *LNS check* demonstrates a particularly strong performance, where it dominates both other settings, . While the best primal integral value is achieved on 28.6%, 40% and 31.4% of the instances for *Benders*, *LNS check* and *Transfer cuts* respectively, *LNS check* achieves a much smaller primal integral across the test



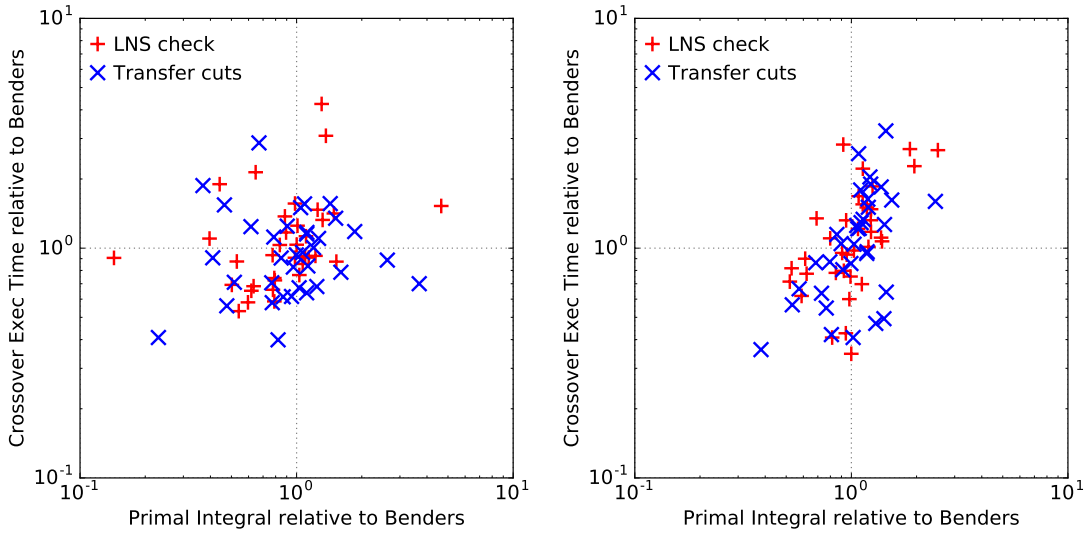
(a) Class 3 (b) Class 4

Figure 2: The performance profile of the primal integral for the SNIP instances comparing different Benders' decomposition settings using default settings for SCIP.

set of all class 3 instances. Similar to the results presented in Section 4.4.1, the transfer of the cuts that were generated while solving (22) to the master problem does not appear to achieve any performance improvement compared to *LNS check*. In fact, the results in Figure 2(a) suggest that the transfer of cuts has a negative impact on the primal integral for the class 3 instances.

The results for the primal integral when solving the class 4 instances, as presented in Figure 2(b), tell a very different story. For this set of instances there is no clear "best" algorithm. Each of the algorithms achieve the lowest primal integral for a subset of the instances and the difference in performance for 60% of the instances is very small. All three of the settings appear to have a similar average behaviour for the majority of instance with no setting exhibiting any large difference to the "best" setting. While the results presented in Figure 2(b) do not show a significant performance improvement from employing the large neighbourhood Benders' search, a positive outcome from these results is that its use does not cause a deterioration to the solving performance.

The results for the SNIP instances highlight a limitation of the large neighbourhood Benders' search in achieving performance improvements. When using Benders' decomposition to solve the auxiliary problem of the large neighbourhood search heuristics, this requires more computational time for heuristics compared to *Benders*. This is most obviously seen when comparing the execution time of the large neighbourhood search heuristic Crossover—the most frequently called large neighbourhood search heuristic—and the primal integral value. To evaluate the impact of employing the large neighbourhood Benders' search, a ratio between the base comparison algorithm of *Benders* and the evaluated settings, *LNS check* and *Transfer cuts*, is computed for the Crossover execution time and the primal integral. This ratio is then presented in the form of a scatter plot in Figure 3. It can be seen that as the execution



(a) Class 3 (b) Class 4

Figure 3: Analysing the relationship between the relative time spent executing the Crossover heuristic and the primal integral. The values are the ratio between *Benders* and the large neighbourhood Benders' search implementations, *LNS check* and *Transfer cuts*.

time of Crossover increases, the value of the primal integral also increases, relative to *Benders*. The increases execution time of Crossover is primarily due to two factors. The first is that the later the optimal solution is found, the more times the heuristic loop is executed. This leads to an increase in the execution time for all heuristics. The second is that when employing the large neighbourhood Benders' search more work is performed within the large neighbourhood search heuristics since (22) is solved instead of (19). Consequently, a linear relationship is observed between the execution time of Crossover and the primal integral relative to *Benders*.

While RINS is also regularly called during the solution process, the same relationship is not observed. This is predominately due to the fact that the execution time of RINS is much shorter than for Crossover when solving these instances. As such, the increase in execution time for RINS does not have a significant impact on the overall run times and the primal integral.

Table 2 presents the average results for the primal integral and the final gap split by instance class and budget. The results in Table 2 show that *LNS check* achieves an overall improvement in the primal integral, with a 6.57% reduction in the primal integral. This overall reduction in the primal integral is predominately driven by the large neighbourhood Benders' search performance on the class 3 instances. A significant reduction in the primal integral is observed for most of the class 3 instance sets, except SNIP3-60, with an average improvement of 13.61%. Specifically, the smallest average improvement of 5.92% is observed for the SNIP3-30 instances; however a deterioration in performance of 4.83% is observed for the SNIP3-60 instances, and the largest improvement of 28.92% is achieved when solving the SNIP3-50 instances.

Table 2: The shifted geometric mean of the primal integral and final gap

SNIP Num and Budget	Num Inst	∅ Primal integral			∅ Final gap (Number solved)		
		Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
SNIP3-30	5	2015.17	**1895.93**	2054.97	**0** (5)	**0** (5)	**0** (5)
SNIP3-40	5	3125.57	**2912.25**	2924.32	10.47 (0)	**9.75** (0)	9.84 (0)
SNIP3-50	5	6575.21	**4673.71**	6287.49	14.4 (0)	**13.55** (0)	14.59 (0)
SNIP3-60	5	**6144.55**	6441.38	9465.02	**16.4** (0)	18.02 (0)	21.19 (0)
SNIP3-70	5	5767.87	4916.15	**4183.85**	19.6 (0)	19.94 (0)	**19.57** (0)
SNIP3-80	5	6139.39	4459.35	**4069.67**	23.13 (0)	23.01 (0)	**21.75** (0)
SNIP3-90	5	10613.52	**9428.8**	9615.35	28.24 (0)	**24.3** (0)	26.99 (0)
SNIP3	35	5148.07	**4447.64**	4799.7	11.38 (5)	**11.09** (5)	11.55 (5)
SNIP4-30	5	**1968.8**	2100.2	2207.74	**0** (5)	**0** (5)	**0** (5)
SNIP4-40	5	**2104.85**	2296.69	2508.87	**2.53** (2)	4.13 (1)	3.66 (1)
SNIP4-50	5	3193.82	3279.37	**2828.25**	10.83 (0)	9.19 (0)	**9.18** (0)
SNIP4-60	5	**3146.61**	3196.82	4978.91	**11.74** (0)	12.86 (0)	14.04 (0)
SNIP4-70	5	4240.1	3918.43	**3598.89**	14.76 (0)	14.14 (0)	**13.25** (0)
SNIP4-80	5	5085.62	**4084.12**	4102	14.6 (0)	15.3 (0)	**10.97** (0)
SNIP4-90	5	5563.08	6641.74	**5484.39**	16.3 (0)	19.25 (0)	**13.88** (0)
SNIP4	35	**3376.51**	3411.84	3491.2	7.09 (7)	7.65 (6)	**6.83** (6)
all	70	4169.45	**3895.55**	4093.62	9.01 (12)	9.23 (11)	**8.92** (11)

The average primal integral results show that the lowest value is achieved by using the *Benders* setting for the class 4 instances. Comparing the performance profile of Figures 2(b) with the results in Table 2, it is possible that this performance improvement could be due to random variation or performance variability. More importantly, this result verifies the observation from Figure 2(b) that there is no clear best setting for the class 4 instances. While the aggregated result for the class 4 indicates that no performance improvement is achieved by employing the large neighbourhood Benders' search, this is not true for all instances within this test set. For 2 and 4 test sets for *LNS check* and *Transfer cuts* respectively the average primal integral is observed to be smaller than for *Benders*. This demonstrates that the large neighbourhood Benders' search has some potential for achieving performance improvements for the class 4 instances.

4.4.3 Stochastic binary multiple knapsack problem

The performance profiles for *Benders*, *LNS check* and *Transfer cuts* when solving the SMKP instances are presented in Figure 4. It can be observed from these profiles that the large neighbourhood Benders' search does not achieve a large improvement in the primal integral over the standard Benders' decomposition implementation. For all of the algorithms, 80% of the instances are within 10% of the best algorithm. The *LNS check* appears to achieve the smallest ratio to the best setting for the majority of instances compared to *Benders* and *Transfer cuts*. However, *Transfer cuts* displays a better overall performance. The results presented in Figure 4 indicate that a small, general improvement is achieved by employing the large neighbourhood Benders' search.

Table 3 details the average performance of the Benders' decomposition algorithms when solving the

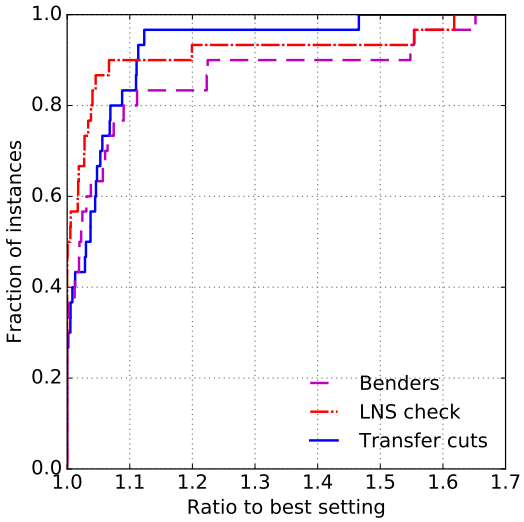


Figure 4: The performance profile of the primal integral for the SMKP instances comparing different Benders' decomposition settings using default settings for SCIP.

SMKP instances—partitioned in the same way as Angulo et al. [2]. The results presented in this paper show a worse performance than that achieved by Angulo et al. [2]. This is primarily due to the fact that SCIP is used as the underlying MIP solver for the experiments in this paper, compared to CPLEX being used by Angulo et al. [2]. Since the computational experiments aim to assess the enhancement of the large neighbourhood Benders' search for Benders' decomposition, it is expected that the relative performed with respect to SCIP demonstrates the potential of this approach.

The results in Table 3 show that employing the large neighbourhood Benders' search achieves an average reduction in the primal integral when solving the SMKP instances. While a deterioration of 0.4% is observed on the easy instance for *LNS check*, a reduction in the primal integral of 0.8% and 8.79%, compared to *Benders*, is achieved for the medium and hard instances respectively. A much more consistent result is presented for *Transfer cuts*, with a 3.51%, 2.01% and 3.93% reduction in the average primal integral for the easy, medium and hard instances respectively. Similar to the previous instance sets—the SCFLP and SNIP instances—there appears to be little benefit from transferring the cuts generated while solving (22) to the master problem compared to simply employing the large neighbourhood Benders' search. While *Transfer cuts* appears to achieve a more consistent performance, the average performance of *LNS check* and *Transfer cuts* for the complete test set is identical. As such, these results further demonstrate that the transferring of cuts do not provide any significant performance improvement for the large neighbourhood Benders' search.

One of the explanations for the lack of performance improvement when employing the large neighbourhood Benders' search is that it is not very difficult to find good feasible solutions for these instances. Across the complete test set, the median time for finding the best solution is 95.09, 97.905, 99.515 sec-

Table 3: The shifted geometric mean of the primal integral and final gap

SMKP	∅ Primal integral			∅ Final gap (Number solved)		
	Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
1-10	538.88	541.04	**519.94**	**0.06** (5)	0.07 (5)	0.07 (5)
11-20	549.22	544.83	**538.17**	0.2 (0)	**0.2** (0)	0.2 (0)
21-30	615.63	**561.49**	591.45	0.27 (0)	**0.26** (0)	0.27 (0)
all	566.93	549.05	**549.05**	0.17 (5)	**0.17** (5)	0.18 (5)

onds for *Benders*, *LNS check* and *Transfer cuts* respectively. In fact, 80% of all instances find the best primal bound within a factor of 2.1 times the best algorithm. This indicates that there is little difference between the settings in regards to the time it takes to find the best solution. The SMKP instances produce a very large branch-and-bound tree during the solution process, thus the majority of the run time for these instances is spent processing nodes. At the end of computation, the shifted geometric mean of the number of nodes for each of the settings is greater than 390 000. This result indicates that the main bottleneck for solving these instances lies in the node processing for proving optimality and not in finding good primal solutions.

4.4.4 Comparing LNS Check and Transfer cuts

The results presented in Sections 4.4.1–4.4.3 suggest that there is little algorithmic advantage of *Transfer cuts* over *LNS check*. In some of the presented cases, the *Transfer cuts* achieves a smaller primal integral; however, the performance appears to be very similar to that reported by *LNS check*. Further, there is little benefit in closing the final gap when transferring cuts from the auxiliary problem to the Benders' decomposition master problem. Only the SNIP instances exhibit a reduced gap from using *Transfer cuts*.

These results show that the transfer of cuts within the large neighbourhood Benders' search does not achieve the same enhancement to the Benders' decomposition algorithm that is reported by Rei et al. [33]. A possible explanation of the differing results is that the cut transfer implementation is different for this algorithm compared to the one developed by Rei et al. [33]. The most obvious difference is that Rei et al. [33] transfers the cuts generated while solving the auxiliary problem to the master problem via the CPLEX cutpool. By comparison, in the algorithm developed for this paper the cuts generated to solve the auxiliary problem are directly added to the master problem as linear constraints. While SCIP also has a cutpool, computational experiments indicated that its use within the Benders' decomposition algorithm resulted in a degradation in performance—the best performance was achieved by adding a linear constraint for each Benders' cut. Therefore, while the results reported in this paper do not support the results presented by Rei et al. [33], this is most likely due to the differences in the implementation of the Benders' decomposition algorithm.

4.5 Using aggressive large neighbourhood search heuristics

The three test sets presented in Sections 4.1–4.3 are used for the experiments when employing the large neighbourhood Benders' search using aggressive large neighbourhood search heuristics. The aggregated results for the primal integral are visualised using performance profiles. Each of the settings that are evaluated are labelled as the large neighbourhood search heuristic that is executed aggressively. For example, label *RINS* is used to describe the setting where RINS is employed aggressively along with the *LNS check* large neighbourhood Benders' search setting. Since *LNS check* is shown in Section 4.4 to outperform *Transfer cuts*, only aggressive heuristic variants of *LNS check* will be presented.

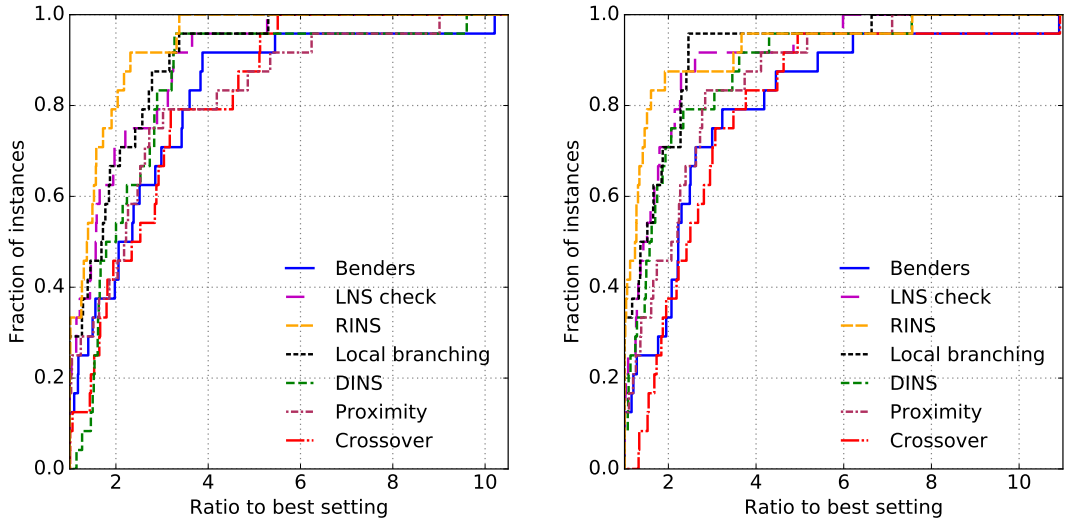
4.5.1 Stochastic Capacitated Facility Location Problem

The performance profiles presented in Figure 5 show that the aggressive heuristics settings are capable of achieving performance improvements over *LNS check*. The results for the 250 scenario instances show that the best performing aggressive heuristics settings, with respect to the primal integral, are *RINS*, *Local Branching* and *DINS*. By aggressively executing RINS, the average primal integral is reduced by 14.18% compared to that achieved by *LNS check* and 33.89% compared to *Benders*.

In regards to the 500 scenario instances, a similar reduction in the primal integral is observed. In particular, *RINS* and *Local Branching* achieve the best performance compared to *LNS check*; however, *DINS* does not show as strong performance. *RINS* still demonstrates a significant reduction in the primal integral compared to *LNS check*—specifically 13.47%. In comparison to *Benders*, *RINS* and *Local Branching* achieve a reduction in the primal integral of 39.55% and 33.05% respectively. These results demonstrate the value of using RINS and Local Branching aggressively when employing the large neighbourhood Benders' search.

Proximity Search has been shown by Boland et al. [9] to achieve reductions in the primal integral compared to a standard Benders' decomposition implementation. This performance is also observed in the results presented in Figure 5; however, the reduction is not as great compared to the other settings evaluated. While *Proximity* achieves a reduction in the primal integral compared to *Benders* for both the 250 and 500 scenario instances—1.85% and 13.03% respectively—it demonstrates poor performance compared to *LNS check*. This is likely due to the implementation of Proximity Search in SCIP not being tailored for use with Benders' decomposition. Further, Boland et al. [9] describe a "Proximity Search with incumbent" variant, which is not implemented in SCIP and could explain the difference in the reported performance. With respect to these experiments, when executing Proximity Search in an aggressive setting a large amount of the run time at the root node is consumed by this heuristic without achieving any significant improvement in the primal bound.

While *RINS*, *Local Branching*, *DINS* and *Proximity* demonstrate some improvement in performance, the aggressive use of Crossover is shown to negatively impact the performance of the Benders' decomposition algorithm for this instance set. This particular result can be explained by the fact that the



(a) 250 Scenarios

(b) 500 Scenarios

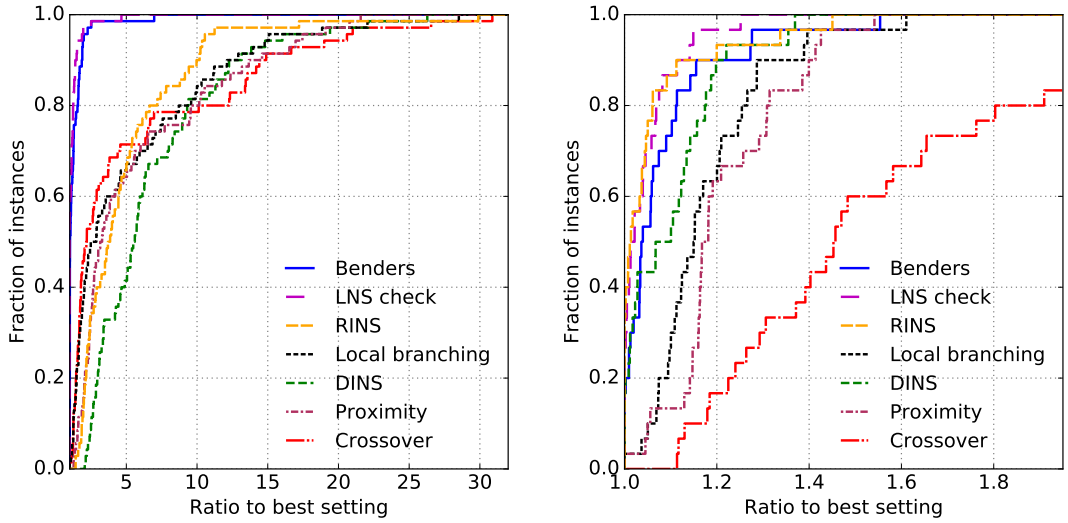
Figure 5: The performance profile of the primal integral for the SCFLP instances comparing the *LNS check* setting when using aggressive heuristics settings of *DINS*, *Crossover*, *Local Branching*, *Proximity* and *RINS*.

very few solutions are found by this heuristic for the *Benders*, *LNS check* and *Transfer cuts* settings. Over the complete instance set Crossover only finds 6, 12 and 14 solutions for *Benders*, *LNS check* and *Transfer cuts* respectively. Comparatively, RINS finds 73, 149 and 111 solutions for the same settings. Hence, it is not surprising that *RINS* performs significantly better than *Crossover* with respect to the primal integral.

The majority of the settings evaluated for these experiments show that there is value in the aggressive use of large neighbourhood search heuristics in the large neighbourhood Benders' search. Since very few modification have been made to the large neighbourhood search heuristics, this result suggests that there is further potential for the aggressive use of heuristics as an enhancement for Benders' decomposition. The performance of large neighbourhood Benders' search with aggressive heuristics verifies the results reported in previous work presented by Rei et al. [33] and Boland et al. [9] and further demonstrates the potential of this enhancement technique.

4.5.2 Stochastic network interdiction problem and stochastic binary multiple knapsack problem

The aggressive heuristic settings present very different results for the SNIP and SMKP instances compared to the SCFLP instances. The results presented in Figure 6(a) show that the aggressive use of large neighbourhood search heuristics has an overall negative effect on performance for the SNIP instances. In fact, none of the aggressive heuristic settings is able to achieve an improvement over *Benders* or *LNS check* with respect to the primal integral.



(a) SNIP (b) SMKP

Figure 6: The performance profile of the primal integral for the SNIP and SMKP instances comparing the *LNS check* setting when using aggressive heuristics settings of *DINS*, *Crossover*, *Local Branching*, *Proximity* and *RINS*.

This result can be explained by the fact that it is not difficult to find good feasible solutions for the SNIP instances. Specifically, the median times for finding the best solutions for *Benders*, *LNS check* and *Transfer cuts* are 658.26, 657.47 and 665.96 respectively. These times are relatively short, and hence, there are very few potential gains that can be achieved from the aggressive use of large neighbourhood search heuristics.

The use of aggressive heuristics settings present mixed results for the SMKP instances. Similar to the SNIP instances, finding good primal bounds is not difficult using the default settings, so it is difficult to achieve performance gains. In fact, for the SMKP instance the median time for finding the best primal bound is much less than that reported for the SNIP test set. As such, some deterioration in the performance, with respect to the primal integral, could be expected from the use of aggressive heuristic settings.

Figure 6(b) presents the performance profile for the aggressive heuristics settings compared against *Benders* and *LNS check* for the SMKP instances. The performance of the large neighbourhood Benders' search with aggressive heuristics is shown to be much better than that presented for the SNIP instances. In particular, all settings, except *Crossover*, achieve a primal integral that is within 65% of the best setting. As such, this demonstrates that there is potential for the use of aggressive heuristics for improving the performance of Benders' decomposition for the SMKP instance.

Interestingly, *RINS* achieves a similar performance to that reported by *LNS check*. This strong performance of *RINS* is similar to that displayed on the SCFLP test sets. Also, *RINS* is the best performing aggressive heuristic setting for the SNIP instances. Given the overall performance, these

results suggest a particular suitability of the RINS heuristic for the large neighbourhood Benders' search.

5 Conclusion

This paper presents a general enhancement approach for Benders' decomposition through the integration with large neighbourhood search heuristics. A key feature of the large neighbourhood Benders' search is that the auxiliary problems of large neighbourhood search heuristics are solved by Benders' decomposition in order to improve the quality of the primal solution found. This extends the traditional use of MIP solvers with Benders' decomposition, where large neighbourhood search heuristics are typically executed in the master problem only—with no consideration of the subproblem constraints. Through the integration of Benders' decomposition and large neighbourhood search heuristics, the large neighbourhood Benders' search is able to more quickly identify high quality primal solutions.

The effectiveness of the large neighbourhood Benders' search is evaluated using three different test sets—SCFLP, SNIP and SMKP. The computational experiments demonstrate the performance improvements that are gained through the use of the large neighbourhood Benders' search. Further, the results demonstrate that the large neighbourhood Benders' search is able to achieve performance improvements on many different instance types, including stochastic integer programs. Overall, the large neighbourhood Benders' search is shown to be a valuable enhancement technique for Benders' decomposition.

The large neighbourhood Benders' search interacts with Benders' decomposition primarily through the master problem structure. Within the large neighbourhood search heuristics, all variable fixings are identified based upon solutions to the master problem. A direction of further research is to investigate whether subproblem information can be used when defining the neighbourhoods. Additionally, the defined neighbourhoods could also involve variable fixings in the subproblems. Finally, this work focuses on the standard Benders' cut generation techniques, which may be inefficient when used within large neighbourhood search heuristics. The development of alternative cut generation approaches for large neighbourhood search heuristics is an area for future research

References

[1] S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, and S. Sen. SIPLIB: a stochastic integer programming test problem library. See `http://www2.isye.gatech.edu/~sahmed/siplib/`.

[2] G. Angulo, S. Ahmed, and S. S. Dey. Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28(3):483–499, 2016.

[3] J. E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164, 1990.

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

[5] T. Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614, 2013.

[6] T. Berthold. RENS. *Mathematical Programming Computation*, 6(1):33–54, 2014.

[7] J. R. Birge, M. A. Dempster, H. I. Gassmann, E. Gunn, A. J. King, and S. W. Wallace. A standard input format for multiperiod stochastic linear programs. Technical Report WP-87-118, IIASA, Laxenburg, Austria, 1987.

[8] M. Bodur, S. Dash, O. Gnlk, and J. Luedtke. Strengthened Benders cuts for stochastic integer programs with continuous recourse. *INFORMS Journal on Computing*, 29(1):77–91, 2017.

[9] N. Boland, M. Fischetti, M. Monaci, and M. Savelsbergh. Proximity Benders: a decomposition heuristic for stochastic programs. *Journal of Heuristics*, 22(2):181–198, 2016.

[10] H. Calik, M. Leitner, and M. Luipersbeck. A Benders decomposition based framework for solving cable trench problems. *Computers & Operations Research*, 81:128–140, 2017.

[11] C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464, 1998.

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

[13] E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[14] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[15] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003.

[16] M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731, 2014.

[17] B. Fortz and M. Poss. An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters*, 37(5):359–364, 2009.

[18] G. Froyland, S. J. Maher, and C.-L. Wu. The recoverable robust tail assignment problem. *Transportation Science*, 48(3):351–372, 2014.

[19] A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by Benders' decomposition. *Management Science*, 20(5):822–844, 1974.

[20] S. Ghosh. *DINS, a MIP Improvement Heuristic*, pages 310–323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[21] ILOG, Inc. ILOG CPLEX: High-performance software for mathematical programming and optimization. See `http://www.ilog.com/products/cplex/`.

[22] G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[23] F. V. Louveaux. Discrete stochastic location models. *Annals of Operations Research*, 6(2):21–34, Feb 1986.

[24] T. Magnanti and R. Wong. Accelerating Benders' decomposition: algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.

[25] S. J. Maher, G. Desaulniers, and F. Soumis. Recoverable robust single day aircraft maintenance routing problem. *Computers & Operations Research*, 51:130–145, 2014.

[26] S. J. Maher, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 4.0. Technical Report 17-12, ZIB, Takustr.7, 14195 Berlin, 2017.

[27] A. Mercier, J. Cordeau, and F. Soumis. A computational study of Benders' decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476, 2005.

[28] F. Pan and D. P. Morton. Minimizing a stochastic maximum-reliability path. *Networks*, 52(3):111–119, 2008.

[29] N. Papadakos. Practical enhancements to the Magnanti-Wong method. *Operations Research Letters*, 36(4):444–449, 2008.

[30] N. Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 36(1):176–195, 2009.

[31] J. Petersen. *Large-scale mixed integer optimization approaches for scheduling airline operations under irregularity*. PhD thesis, Georgia Institute of Technology, May 2012.

[32] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: a literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[33] W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders' decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345, 2009.

[34] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.

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

[36] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115, 2005.

[37] E. Thorsteinsson. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *Principles and Practice of Constraint Programming - CP 2001*, pages 16–30. Springer, 2001.

A Detailed computational results

Table 4: The primal integral, final gap and run time for the SCFLP instances with 250 scenarios

Instance	Primal integral			Final gap		
	Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
cap81	2424.13	3266.9	1924.25	6.59	7.03	6.6
cap82	805.52	980.61	3701.43	7.19	7.1	8
cap83	2192.23	1846.4	2629.32	7.45	7.54	7.33
cap84	2482.59	2414.11	1437.17	6.77	7.04	6.2
cap91	7476.32	7241.45	7240.81	16.43	16.43	16.32
cap92	8333.15	6996.17	7107.92	19.91	20.42	19.92
cap93	3557.27	8229.47	1750.34	19.96	21.26	19.24
cap94	2735.43	3876.34	3040.13	20.05	20.16	20.06
cap101	6772.57	6873.29	6114.66	18.34	18.65	17.58
cap102	6391.21	2141.39	9375.88	24.84	24.34	25.19
cap103	8932.06	2734.87	2895.34	28.51	27.64	27.21
cap104	4534.18	4420	1819.3	27.4	28.91	27.63
cap111	10689.62	10651.4	10631.19	14.72	14.73	14.71
cap112	1553.78	2561.33	6149.09	13.14	12.89	14.41
cap113	11465.21	4738.23	6319.53	15.3	12.86	12.67
cap114	6642.49	10444.58	11608.49	11.77	12.76	12.94
cap121	15753.87	9792.73	11484.56	26.21	23.81	24.02
cap122	21536.21	7549.15	11096.27	29.4	26.43	27.7
cap123	10046.42	4424.12	3359.85	30.36	26.9	28.76
cap124	3720.15	3394.52	3883.39	28.43	27.12	26.8
cap131	24061.12	13316.23	15867.9	33.08	27.62	29.61
cap132	21266.95	10657.23	14595.07	40.11	35.36	36.03
cap133	16719.91	8133.26	4996.43	41.65	37.31	36.63
cap134	24099.04	10127.4	13067.09	43.16	40.35	39.07

Table 5: The primal integral, final gap and run time for the SCFLP instances with 500 scenarios

Instance	Primal integral			Final gap		
	Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
cap81	3409.24	1969.61	2812.91	6.82	6.57	6.7
cap82	5772.14	3587.87	3650.48	8.35	7.84	7.74
cap83	3492.01	3672.8	4412.97	8.02	8.16	7.99

Table 5: (continued)

cap84	1589.34	1580.45	1545.15	7.28	7.25	7.22
cap91	6443.1	3109	3088.71	16.9	15.66	14.99
cap92	6927.34	5929.02	6008.86	21.53	20.91	21.21
cap93	9848.35	6886.64	4954.06	22.36	21.52	21.22
cap94	8705.69	3719.46	6138.36	22.43	21.22	22.32
cap101	6463.69	5251.37	6350.11	20.01	18.69	19.71
cap102	8208.88	5822.81	2162.27	27.05	26.08	24.75
cap103	7621.63	2248.44	4440.87	29.44	27.72	27.93
cap104	7771.13	4695.24	4364.58	29.53	28.53	29.29
cap111	4122.13	4877.4	5551.03	12.45	12.67	12.94
cap112	7128.3	6649.55	2453.1	13.36	13.19	11.66
cap113	5606.03	5581.92	5595.17	12.95	12.94	12.95
cap114	4025.65	4151.24	2403.03	11.62	11.61	10.97
cap121	16629.38	11673.86	11690.07	27.15	24.98	25.23
cap122	15875.43	7123.25	8526.37	31.74	28.09	29.04
cap123	11961.01	5377.7	5271.21	31.32	27.98	28.56
cap124	5692.79	9072.7	12600.8	26.87	29	30.63
cap131	18308.97	11724.31	14387.65	32.65	28.49	30.95
cap132	27481.86	15065.5	15003.69	41.5	35.48	36.03
cap133	15366.03	17814.05	21399.59	39.67	42.19	44.53
cap134	18226.64	6716.02	8367.73	41.26	40.8	41.73

Table 6: The primal integral when using the aggressive heuristic settings for the SCFLP instances with 250 scenarios

Instance	*Benders*	*LNS check*	*Crossover*	*DINS*	*Local Branching*	*Proximity*	*RINS*
cap81	2424.13	3266.9	1027.47	1522.7	2458.92	2512.11	1017.08
cap82	805.52	980.61	4127.96	2039.31	2240.83	1218.54	816.53
cap83	2192.23	1846.4	1955.9	2963.33	3114.68	3430.41	2334.76
cap84	2482.59	2414.11	3459.2	3433.32	2714.43	2108.68	2943.98
cap91	7476.32	7241.45	7556.4	2085.74	4625.69	8563.29	1372.24
cap92	8333.15	6996.17	4395.92	7001.02	6225.67	2494.53	2420.13
cap93	3557.27	8229.47	4934.91	5685.72	2539.07	5637.85	3481.29
cap94	2735.43	3876.34	2523.55	4808.46	3669.4	1758.58	2621.81
cap101	6772.57	6873.29	5355.4	6156.53	9981.95	10060.59	1882.83
cap102	6391.21	2141.39	9964	6955.34	6765.24	10402.36	4659.11
cap103	8932.06	2734.87	4471.04	8401.82	874.97	7885.41	2900.32
cap104	4534.18	4420	5449.93	3475.11	3032.35	3757.95	3066.9
cap111	10689.62	10651.4	5447.2	8305.97	6837.53	8843.91	5406.9
cap112	1553.78	2561.33	4954.21	3111.06	2157.15	6505.07	3595.57
cap113	11465.21	4738.23	9095.52	8474.12	8118.86	2990.97	6097.69
cap114	6642.49	10444.58	9687.74	11875.26	12287.88	14461.39	10129.98
cap121	15753.87	9792.73	18334.29	7946.45	11732.08	17068.33	6269.39
cap122	21536.21	7549.15	17681.24	12447.81	13374.3	19845	9437.14
cap123	10046.42	4424.12	12267.05	9153.74	7332.29	4264	6678.54
cap124	3720.15	3394.52	15388.4	10884.39	4911.18	8578.74	11450.15
cap131	24061.12	13316.23	17864	17026.39	11682.66	26431.61	15240.58
cap132	21266.95	10657.23	17377.71	15525.5	5486.56	16575.93	9443.49
cap133	16719.91	8133.26	20566.47	13117.93	8132.37	16556.19	15504.47
cap134	24099.04	10127.4	11677.86	11737.8	7037.57	7370.86	11081.69

Table 7: The primal integral when using the aggressive heuristic settings for the SCFLP instances with 500 scenarios

Instance	*Benders*	*LNS check*	*Crossover*	*DINS*	*Local Branching*	*Proximity*	*RINS*
cap81	3409.24	1969.61	3764.9	3986.95	3205.71	3353.49	1931.06
cap82	5772.14	3587.87	3352.19	2509.8	3327.14	3464.44	2600.76
cap83	3492.01	3672.8	3837.82	3729.63	3409.09	3470.21	2905.53
cap84	1589.34	1580.45	3953.7	2347.4	2142.92	2541.26	3036.62
cap91	6443.1	3109	5515.8	4308.38	7908.05	8466.53	1191.81
cap92	6927.34	5929.02	5084.83	4137.77	2764.37	7849.92	3912.86
cap93	9848.35	6886.64	8739.78	7366.98	7676.92	5048.67	5690.83
cap94	8705.69	3719.46	7005.67	6793.22	2902.97	6510.79	3687.82
cap101	6463.69	5251.37	8209.53	8907.74	7042.05	8092.04	2921.51
cap102	8208.88	5822.81	8831.41	3604.15	7574.76	13564.74	3298.04
cap103	7621.63	2248.44	8470.08	7354.25	1711.44	4094.59	2744.15
cap104	7771.13	4695.24	6303.95	7007.53	3752.56	4723.49	4773.48

Table 7: (continued)

cap111	4122.13	4877.4	7189.56	3458.95	3221.15	3545.54	3989.73
cap112	7128.3	6649.55	9370.12	10717.7	11336.72	8360.76	6143.25
cap113	5606.03	5581.92	12113.89	2924.83	6154.85	7103.46	2701.07
cap114	4025.65	4151.24	11876.64	6843.95	7531.85	8326.57	6090.26
cap121	16629.38	11673.86	15808.67	5148.66	11727.18	8518.25	5455.4
cap122	15875.43	7123.25	13344.41	8089.75	11772.88	16169.19	7243.12
cap123	11961.01	5377.7	8334.73	6699.38	13210.44	11873.37	7028.54
cap124	5692.79	9072.7	12447.73	11009.39	13993.47	5711.09	7623.33
cap131	18308.97	11724.31	20986.19	11304.66	6983.65	19014.15	10187
cap132	27481.86	15065.5	27530.65	19015.08	3429.62	2518.12	19043.84
cap133	15366.03	17814.05	12788.34	12695.8	3670.64	13727.39	13464.45
cap134	18226.64	6716.02	11054.08	4610.5	2934.99	15159.48	10224.51

Table 8: The primal integral, final gap and run time for the SMKP instances

Instance	Primal integral			Final gap			Run time		
	Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
snip3-inst0-budget30	1824.97	1998.92	2036.5	0	0	0	1042.3	668.4	924.81
snip3-inst0-budget40	2355.4	2602.84	2349.05	9.24	7.21	7.47	3600	3600	3600
snip3-inst0-budget50	4408.84	3498.45	4259.46	9.37	11.69	8.76	3600	3600	3600
snip3-inst0-budget60	3866.55	17993.75	10144.73	12.39	22.25	15.23	3600	3600.01	3600
snip3-inst0-budget70	4671.19	5315.56	7075.43	19.29	19.17	20.39	3600	3600	3600
snip3-inst0-budget80	4975.51	6493.46	3334.6	27.35	17.96	18.42	3600.01	3600	3600
snip3-inst0-budget90	9889.87	6399.42	3653.05	29.14	22.18	25.14	3600	3600	3600
snip3-inst1-budget30	2402.86	1896.08	1854.88	0	0	0	2224.31	1092.43	2840.33
snip3-inst1-budget40	2370.61	2393.7	2246.35	9.29	10.92	9.37	3600	3600	3600.01
snip3-inst1-budget50	6098.04	5101.11	3145.6	15.03	14.08	14.66	3600	3600	3600
snip3-inst1-budget60	6320.82	3174.64	7032.39	15.8	14.89	19.04	3600	3600	3600
snip3-inst1-budget70	4352.64	2755.26	5497.35	21.14	17.53	18.28	3600	3600	3600
snip3-inst1-budget80	8425.88	5024.06	5186.08	23.73	29.42	23.54	3600	3600	3600
snip3-inst1-budget90	6929.65	8466.99	11093.77	24.46	24	29.55	3600	3600	3600
snip3-inst2-budget30	1997.69	1764.1	2173.39	0	0	0	1829.87	2873.85	3130.91
snip3-inst2-budget40	4233.19	3281.1	4511.37	13.28	9.53	12.4	3600	3600	3600
snip3-inst2-budget50	5114.35	6395.15	6340.58	10.58	11.62	11.73	3600	3600	3600.01
snip3-inst2-budget60	3835.96	5238.04	3010.21	19.7	21.14	27.51	3600	3600	3600.01
snip3-inst2-budget70	4143.35	5463.61	4609.17	19.55	23.75	22.19	3600	3600.01	3600
snip3-inst2-budget80	4790.96	3724.95	4067.07	19.72	22.66	21.38	3600	3600	3600
snip3-inst2-budget90	8187.74	12169.05	15187.96	21.86	23.42	30.22	3600	3600	3600
snip3-inst3-budget30	1783.86	1753.54	1862.21	0	0	0	1685.25	1582.89	1657.56
snip3-inst3-budget40	4754.29	3752.06	3903.63	12.27	10.86	11.81	3600	3600	3600
snip3-inst3-budget50	28579.35	4100.1	25870.69	29.04	14.66	30.4	3600.01	3600	3600.01
snip3-inst3-budget60	13302.88	5271.15	13664.58	14.96	15.08	17.64	3600	3600	3600
snip3-inst3-budget70	6335.68	5664.94	2597.43	18.93	19.84	16.83	3600	3600.01	3600
snip3-inst3-budget80	5187.04	2803.92	3983.18	23.26	22.69	24	3600	3600	3600
snip3-inst3-budget90	17049.26	7524.5	7916.16	37.87	24.87	25.02	3600.01	3600	3600
snip3-inst4-budget30	2126.27	2089.25	2396.72	0	0	0	1126.54	1294.04	911.22
snip3-inst4-budget40	2652.54	2730.47	2299.62	8.94	10.75	8.92	3600	3600	3600
snip3-inst4-budget50	3121.8	4764.58	4462.93	13.96	16.25	14.08	3600	3600	3600
snip3-inst4-budget60	7017.5	7022.59	25845.93	20.44	17.94	30.15	3600	3600	3600
snip3-inst4-budget70	11953.45	6331.62	2750.44	19.16	19.88	20.58	3600	3600	3600
snip3-inst4-budget80	8369.82	5172.75	3984.28	22.21	23.67	21.88	3600	3600	3600
snip3-inst4-budget90	14073.82	15016.39	16852.46	30.34	27.31	25.49	3600	3600	3600
snip4-inst0-budget30	2032.55	2013.7	2070.56	0	0	0	590	501.54	1010.53
snip4-inst0-budget40	2073.64	2240.14	2545.89	0	0	0	1277.14	1262.7	1246.32
snip4-inst0-budget50	4365	2311.74	2503.95	10.36	5.43	8.38	3600	3600	3600
snip4-inst0-budget60	2575.38	2908.62	3715.19	8.63	8.39	9.34	3600	3600	3600
snip4-inst0-budget70	2849.8	3513.36	3411.59	8.76	10.56	8.02	3600	3600	3600.22
snip4-inst0-budget80	7657.21	3979.14	2929.28	13.9	11.61	8.14	3600	3600	3600
snip4-inst0-budget90	5404.28	5111.85	3933.71	13.09	12.38	6.57	3600	3600	3600
snip4-inst1-budget30	2013.06	1970.31	2070.73	0	0	0	1038.66	1131.32	2521.01
snip4-inst1-budget40	2012.28	2009.98	2155.13	0	4.7	4.57	3162.95	3600	3600
snip4-inst1-budget50	2434.24	2720.51	3527.56	11.92	10.53	9.76	3600	3600	3600
snip4-inst1-budget60	3208.97	3954.06	4928.8	12.35	12.93	12.55	3600	3600	3600
snip4-inst1-budget70	4304.14	2534.99	3928.67	16.38	13.39	11.83	3600	3600	3600
snip4-inst1-budget80	4688.77	4462.16	4017.84	14.38	16.85	10.85	3600	3600	3600
snip4-inst1-budget90	5179.76	4135.85	6098.13	13.22	18.83	13.05	3600	3600	3600
snip4-inst2-budget30	2064.88	2121.2	2339.64	0	0	0	2238.12	2200.36	2967.76
snip4-inst2-budget40	2336.73	2118.74	2783.15	7.11	5.91	7.44	3600	3600	3600
snip4-inst2-budget50	2281.14	5722.38	2414.21	13.15	14.63	14.06	3600	3600	3600.01

Table 8: (continued)

snip4-inst2-budget60	4603.7	2817.42	6507.82	14.3	16.89	19.19	3600	3600	3600
snip4-inst2-budget70	5437.29	3755.6	2900.7	17.48	14.28	17.6	3600	3600	3600
snip4-inst2-budget80	4111.73	3794.45	4861.84	16.82	18.86	16.42	3600	3600	3600
snip4-inst2-budget90	4217.72	5056.51	4196.25	17.5	20.12	13.66	3600	3600	3600
snip4-inst3-budget30	2005.59	2262.54	2753.95	0	0	0	1153.7	1182.64	1599.2
snip4-inst3-budget40	2160.85	1832.37	3072.98	6.19	7.56	6.42	3600	3600	3600
snip4-inst3-budget50	4704.42	3835.66	3804.13	9.7	8.02	4.25	3600	3600	3600
snip4-inst3-budget60	2793.25	3869.51	3619.86	10.5	11.73	12.99	3600	3600	3600
snip4-inst3-budget70	4284.65	5367.27	4729.04	13.66	13.97	14.7	3600	3600	3600
snip4-inst3-budget80	3193.19	3763.21	3539.67	11.74	11.55	9.45	3600	3600	3600
snip4-inst3-budget90	10521.86	14426.92	9434.28	20.66	23.69	20.66	3600	3600	3600
snip4-inst4-budget30	1745.61	2145.78	1898.22	0	0	0	841.58	705.36	1253
snip4-inst4-budget40	1960.8	3653.42	2117.84	8.46	9.56	5.31	3600	3600	3600
snip4-inst4-budget50	2912.41	2745.52	2229.89	9.41	9.53	12.7	3600	3600	3600
snip4-inst4-budget60	2901.75	2662.3	7090.1	13.83	16.18	18.46	3600	3600	3600
snip4-inst4-budget70	4794.62	5142.97	3283.08	20.15	19.93	16.4	3600	3600	3600
snip4-inst4-budget80	7212.71	4481.61	5732.08	16.76	19.52	11.46	3600	3600	3600
snip4-inst4-budget90	4287.11	8374.04	5222.62	18.32	23.61	20.61	3600	3600	3600

Table 9: The primal integral when using the aggressive heuristic settings for the SMKP instances

Instance	*Benders*	*LNS check*	*Crossover*	*DINS*	*Local Branching*	*Proximity*	*RINS*
snip3-inst0-budget30	1824.97	1998.92	2738.99	4862.65	2988.8	3376.47	3359.82
snip3-inst0-budget40	2355.4	2602.84	4079.59	10039.39	3982.01	5584.13	4256.92
snip3-inst0-budget50	4408.84	3498.45	12992.48	20433.43	24724.95	41139.07	18795.86
snip3-inst0-budget60	3866.55	17993.75	16946.95	24255.19	39884.89	25201.99	17178.05
snip3-inst0-budget70	4671.19	5315.56	11411.27	43892.34	21543.04	28062.46	43994.62
snip3-inst0-budget80	4975.51	6493.46	50344.86	65457.58	73544.14	86829.44	45488.06
snip3-inst0-budget90	9889.87	6399.42	86802.38	88516.35	63351.6	49360.42	56225.42
snip3-inst1-budget30	2402.86	1896.08	2643.55	4607.9	2672.05	6527.74	4126.45
snip3-inst1-budget40	2370.61	2393.7	4447.54	7707.88	7800.61	11341.37	5329.48
snip3-inst1-budget50	6098.04	5101.11	15547.6	15787.74	10486.35	19544.73	11369.77
snip3-inst1-budget60	6320.82	3174.64	60297.87	38785.07	20649.56	60594.68	20312.76
snip3-inst1-budget70	4352.64	2755.26	73159.98	60927.18	60749.67	57254.5	18303.91
snip3-inst1-budget80	8425.88	5024.06	105545.41	80558.99	94666.9	108381.07	86483.26
snip3-inst1-budget90	6929.65	8466.99	98269.67	78293.34	89490.82	73060.97	49825.3
snip3-inst2-budget30	1997.69	1764.1	2387.81	4660.36	2849.63	3485.16	3826.06
snip3-inst2-budget40	4233.19	3281.1	5509.8	6984.79	6346.54	9350.02	9198.41
snip3-inst2-budget50	5114.35	6395.15	8920.89	11285.48	44579.06	25812.79	11989.4
snip3-inst2-budget60	3835.96	5238.04	55245.15	34322.39	57846.39	55997.77	40484.67
snip3-inst2-budget70	4143.35	5463.61	61694.85	50809.33	50197.22	42603.88	23588.17
snip3-inst2-budget80	4790.96	3724.95	76737.15	72274.8	52013.28	61559.44	27656.74
snip3-inst2-budget90	8187.74	12169.05	100516.83	73353.46	91333.4	83151.29	28551.99
snip3-inst3-budget30	1783.86	1753.54	2513.61	5476.59	2199.96	3524.9	3239.97
snip3-inst3-budget40	4754.29	3752.06	7533.97	9531.15	6118.24	7613.77	7500.43
snip3-inst3-budget50	28579.35	4100.1	26028.96	33915.85	31045.43	39202.69	23765.47
snip3-inst3-budget60	13302.88	5271.15	35233.14	60032.44	50513.4	28259.15	41203.53
snip3-inst3-budget70	6335.68	5664.94	39422.23	67885.77	38818.56	54006.49	57674.77
snip3-inst3-budget80	5187.04	2803.92	86590.91	73688.38	79946.92	47573.11	83797.13
snip3-inst3-budget90	17049.26	7524.5	92752.08	68970.11	55867.13	71077.09	38137
snip3-inst4-budget30	2126.27	2089.25	2665.16	4304.59	3092.98	5689.01	2961.71
snip3-inst4-budget40	2652.54	2730.47	17371.35	6076.95	3789.35	8849.81	6515.8
snip3-inst4-budget50	3121.8	4764.58	41958.6	29036.82	15584.17	42638.14	11169.14
snip3-inst4-budget60	7017.5	7022.59	26752.04	44187.25	59656.25	23069.45	32235.03
snip3-inst4-budget70	11953.45	6331.62	84806.83	68871.29	63280.65	83412.26	64845.9
snip3-inst4-budget80	8369.82	5172.75	86177.22	76645.52	57990.53	63954.66	58148.09
snip3-inst4-budget90	14073.82	15016.39	91258.51	84014.89	80848.81	78102.07	62419.83
snip4-inst0-budget30	2032.55	2013.7	2274.78	6825.29	2517.13	2561.8	3392.49
snip4-inst0-budget40	2073.64	2240.14	2496.13	7140.77	2774.29	4257.38	4492.17
snip4-inst0-budget50	4365	2311.74	6066.78	6925.11	3427.99	6126.7	4900.23
snip4-inst0-budget60	2575.38	2908.62	3587.4	14922.14	5889.15	6738.59	9925.13
snip4-inst0-budget70	2849.8	3513.36	10633.31	15637.72	13033.02	12965.84	14257.81
snip4-inst0-budget80	7657.21	3979.14	7133.79	21097.91	8097.86	10201.08	12537.6
snip4-inst0-budget90	5404.28	5111.85	6463.97	32559.93	8453.13	12748.54	12408.97
snip4-inst1-budget30	2013.06	1970.31	2805.67	4733.01	2461.09	2523.94	4034.85
snip4-inst1-budget40	2012.28	2009.98	3600.93	5912.94	2595.88	3115.14	4819.08
snip4-inst1-budget50	2434.24	2720.51	7049.96	10810.98	5367.17	7323.59	7088.63
snip4-inst1-budget60	3208.97	3954.06	4966.14	10728.43	4674.97	7580.28	7782.2
snip4-inst1-budget70	4304.14	2534.99	6724.79	18209.27	15047.86	16601.96	15559.12

Table 9: (continued)

snip4-inst1-budget80	4688.77	4462.16	7171.97	24991.34	9627.24	10683.29	17558.28
snip4-inst1-budget90	5179.76	4135.85	11211.27	32786.27	9530.53	15662.43	13433.48
snip4-inst2-budget30	2064.88	2121.2	3003.07	6255.99	2243.92	2977.34	3288.7
snip4-inst2-budget40	2336.73	2118.74	3338.21	9857.03	3936.69	4023.9	5985.93
snip4-inst2-budget50	2281.14	5722.38	10433.96	10432.04	6608.68	7192.79	7966.49
snip4-inst2-budget60	4603.7	2817.42	5389.65	14359.68	9982.47	6852.26	13121.97
snip4-inst2-budget70	5437.29	3755.6	6722.27	22042.39	6702.69	21021.53	37417.4
snip4-inst2-budget80	4111.73	3794.45	4659.17	19160.08	5717.47	9241.78	15116.27
snip4-inst2-budget90	4217.72	5056.51	7699.48	27671.83	17673.24	12722.93	17648.7
snip4-inst3-budget30	2005.59	2262.54	3361.98	5414.71	3489.75	3208.13	3921.03
snip4-inst3-budget40	2160.85	1832.37	3051.51	9693.73	4458.87	4962.74	6616.83
snip4-inst3-budget50	4704.42	3835.66	5975.74	16101.46	4490.85	9922.36	10318.02
snip4-inst3-budget60	2793.25	3869.51	5717.74	16015.79	5578.73	7724.45	10329.85
snip4-inst3-budget70	4284.65	5367.27	14373.17	23128.62	8109.26	10026.15	17328.39
snip4-inst3-budget80	3193.19	3763.21	6936.56	25573.15	10982.63	12493.4	33082.51
snip4-inst3-budget90	10521.86	14426.92	13021.81	25921.78	10357.52	14077.67	14641.95
snip4-inst4-budget30	1745.61	2145.78	2205.6	4981.78	2022.52	2684.86	3592.03
snip4-inst4-budget40	1960.8	3653.42	2410.2	6640.55	4874.98	4085.07	3723.99
snip4-inst4-budget50	2912.41	2745.52	4602.04	7825.72	3474.03	4421.28	5430.85
snip4-inst4-budget60	2901.75	2662.3	5890.93	13018.45	8034.69	5740.56	13970.57
snip4-inst4-budget70	4794.62	5142.97	3401.04	20890.97	15181.25	12192.54	18420.77
snip4-inst4-budget80	7212.71	4481.61	13018.46	25657.46	12910.04	10213.55	20095.91
snip4-inst4-budget90	4287.11	8374.04	9445.53	36076.11	12475.88	18045.52	20779.31

Table 10: The primal integral, final gap and run time for the SMKP instances

Instance	Primal integral			Final gap			Run time		
	Benders	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*	*Benders*	*LNS check*	*Transfer cuts*
smkp_01	410.38	417.64	412.62	0	0	0	272.83	252.46	326.89
smkp_02	435.55	421.81	410.64	0	0	0	517.52	348.17	369.6
smkp_03	509.13	498.64	308.19	0	0	0	1020.9	981.88	689.54
smkp_04	457.95	488.54	508.71	0	0	0	2186.09	1943.62	2513.64
smkp_05	783.19	787.18	837.26	0.04	0.08	0.08	3600	3600	3600
smkp_06	698.93	685.49	706.08	0	0	0	2414.53	2597.12	2526.73
smkp_07	581.92	574.82	601.28	0.09	0.09	0.09	3600	3600	3600
smkp_08	551.48	538.98	518.03	0.16	0.15	0.16	3600	3600	3600
smkp_09	600.49	617.16	603.58	0.14	0.14	0.12	3600	3600	3600
smkp_10	468.07	483.78	485.54	0.22	0.22	0.23	3600	3600	3600
smkp_11	395.02	362.33	402.18	0.12	0.09	0.14	3600	3600	3600
smkp_12	702.94	694.61	695.89	0.22	0.22	0.22	3600	3600	3600
smkp_13	442.79	441.69	466.51	0.18	0.18	0.18	3600	3600	3600
smkp_14	993.97	994.08	1022.27	0.26	0.27	0.27	3600	3600	3600
smkp_15	374.92	372.24	366.04	0.18	0.18	0.18	3600	3600	3600
smkp_16	574.35	600.5	601.75	0.2	0.21	0.2	3600	3600	3600
smkp_17	446.28	446.05	465.98	0.21	0.21	0.21	3600	3600	3600
smkp_18	442.55	432.86	462.29	0.2	0.2	0.2	3600	3600	3600
smkp_19	726.53	729.52	469.33	0.23	0.24	0.23	3600	3600	3600
smkp_20	656.06	656.54	680.59	0.23	0.22	0.22	3600	3600	3600
smkp_21	529.29	500.71	557.56	0.28	0.27	0.3	3600	3600	3600
smkp_22	896.98	732.6	822.79	0.35	0.27	0.32	3600	3600	3600
smkp_23	480.52	480.6	462.98	0.26	0.26	0.26	3600	3600	3600
smkp_24	771.77	718.36	781.41	0.22	0.23	0.24	3600	3600	3600
smkp_25	455.74	409.97	431.69	0.27	0.26	0.27	3600	3600	3600
smkp_26	672.44	664.51	652.42	0.23	0.22	0.23	3600	3600	3600
smkp_27	701.88	705.9	707.75	0.25	0.26	0.25	3600	3600	3600
smkp_28	461.74	453.15	458.95	0.26	0.27	0.27	3600	3600	3600
smkp_29	698.24	449.37	658.68	0.28	0.26	0.26	3600	3600	3600
smkp_30	637.89	625.55	521.62	0.3	0.3	0.29	3600	3600	3600

Table 11: The primal integral when using the aggressive heuristic settings for the SMKP instances

Instance	*Benders*	*LNS check*	*Crossover*	*DINS*	*Local Branching*	*Proximity*	*RINS*
smkp_01	410.38	417.64	576.38	388.54	446.69	445.86	405.59
smkp_02	435.55	421.81	578.53	422.58	474.29	492.33	425.03
smkp_03	509.13	498.64	565.5	484.95	514.47	501.18	479.47
smkp_04	457.95	488.54	542.4	470.77	501.18	480.83	459.06

Table 11: (continued)

smkp_05	783.19	787.18	1151.2	888.49	889.72	932.87	784.55
smkp_06	698.93	685.49	987.64	612.16	660.61	716.07	547.71
smkp_07	581.92	574.82	701.12	591.52	609.35	568.97	503.76
smkp_08	551.48	538.98	764.4	541.36	591.93	532.06	551.69
smkp_09	600.49	617.16	652.68	577.56	665.6	669.88	599.33
smkp_10	468.07	483.78	682	473.71	502.78	534.36	483.33
smkp_11	395.02	362.33	599.45	408.97	503.26	476.25	403.05
smkp_12	702.94	694.61	859.07	679.77	704.79	789.04	687.5
smkp_13	442.79	441.69	698.73	451.83	464.01	513.62	449.46
smkp_14	993.97	994.08	2354.2	1175.91	1148.57	1233.83	981.4
smkp_15	374.92	372.24	460.53	410.32	396.97	432.07	371.32
smkp_16	574.35	600.5	834.36	675.76	727.22	751.64	578.85
smkp_17	446.28	446.05	544.8	433.26	462.49	544.27	421.42
smkp_18	442.55	432.86	601.57	428.83	471.93	507.14	434.76
smkp_19	726.53	729.52	948.96	775.45	805.54	833.46	760.36
smkp_20	656.06	656.54	1240.64	715.16	755.93	767.85	649.76
smkp_21	529.29	500.71	2495.88	611.2	699.05	700.71	669.63
smkp_22	896.98	732.6	784.88	804.98	704.71	744.05	845.5
smkp_23	480.52	480.6	708.99	460.46	537.6	502.87	431.51
smkp_24	771.77	718.36	1511.14	983.72	924.34	1015.6	784.2
smkp_25	455.74	409.97	722.48	474.38	527.45	567.91	435.11
smkp_26	672.44	664.51	1704.87	772.27	815.6	759.4	650.27
smkp_27	701.88	705.9	686.21	619.88	707.8	725.81	614.34
smkp_28	461.74	453.15	513	418.82	470.16	506.44	439.86
smkp_29	698.24	449.37	704.34	608.85	723.76	692.7	651.79
smkp_30	637.89	625.55	1611.65	702.36	750.89	891.78	663.32

B Non-default parameter settings

The following table present the parameter settings that are different from the default SCIP settings in the presented computational experiments. In addition, a parameter is available to control the maximum depth at which the large neighbourhood Benders' search is employed. For the SMKP and SNIP instances, this parameter is set to 10. This means that at every node that has a depth greater than 10, the auxiliary problem of the large neighbourhood search heuristics is solved a (19). For the SCFLP instances, no maximum depth is imposed. This parameter is not shown in the following table.

Table 12: Non-default parameter settings

Benders	*LNS check*
presolving/maxrounds = 0	presolving/maxrounds = 0
presolving/maxrestarts = 0	presolving/maxrestarts = 0
propagating/maxrounds = 0	propagating/maxrounds = 0
propagating/maxroundsroot = 0	propagating/maxroundsroot = 0
constraints/nonlinear/maxprerounds = 0	constraints/nonlinear/maxprerounds = 0
constraints/quadratic/maxprerounds = 0	constraints/quadratic/maxprerounds = 0
constraints/linear/maxprerounds = 0	constraints/linear/maxprerounds = 0
constraints/abspower/maxprerounds = 0	constraints/abspower/maxprerounds = 0
constraints/and/maxprerounds = 0	constraints/and/maxprerounds = 0
constraints/bivariate/maxprerounds = 0	constraints/bivariate/maxprerounds = 0
constraints/bounddisjunction/maxprerounds = 0	constraints/bounddisjunction/maxprerounds = 0
constraints/cardinality/maxprerounds = 0	constraints/cardinality/maxprerounds = 0
constraints/conjunction/maxprerounds = 0	constraints/conjunction/maxprerounds = 0
constraints/cumulative/maxprerounds = 0	constraints/cumulative/maxprerounds = 0
constraints/disjunction/maxprerounds = 0	constraints/disjunction/maxprerounds = 0
constraints/indicator/maxprerounds = 0	constraints/indicator/maxprerounds = 0
constraints/knapsack/maxprerounds = 0	constraints/knapsack/maxprerounds = 0
constraints/linking/maxprerounds = 0	constraints/linking/maxprerounds = 0
constraints/logicor/maxprerounds = 0	constraints/logicor/maxprerounds = 0
constraints/or/maxprerounds = 0	constraints/or/maxprerounds = 0
constraints/orbitope/maxprerounds = 0	constraints/orbitope/maxprerounds = 0
constraints/pseudoboolean/maxprerounds = 0	constraints/pseudoboolean/maxprerounds = 0
constraints/setppc/maxprerounds = 0	constraints/setppc/maxprerounds = 0

Table 12: (continued)

constraints/soc/maxprerounds = 0	constraints/soc/maxprerounds = 0
constraints/SOS1/maxprerounds = 0	constraints/SOS1/maxprerounds = 0
constraints/SOS2/maxprerounds = 0	constraints/SOS2/maxprerounds = 0
constraints/superindicator/maxprerounds = 0	constraints/superindicator/maxprerounds = 0
constraints/varbound/maxprerounds = 0	constraints/varbound/maxprerounds = 0
constraints/xor/maxprerounds = 0	constraints/xor/maxprerounds = 0
constraints/components/maxprerounds = 0	constraints/components/maxprerounds = 0
reading/storeader/usebenders = TRUE	reading/storeader/usebenders = TRUE
presolving/domcol/maxrounds = 0	presolving/domcol/maxrounds = 0
presolving/dualcomp/maxrounds = 0	presolving/dualcomp/maxrounds = 0
presolving/gateextraction/maxrounds = 0	presolving/gateextraction/maxrounds = 0
presolving/implics/maxrounds = 0	presolving/implics/maxrounds = 0
presolving/inttobinary/maxrounds = 0	presolving/inttobinary/maxrounds = 0
presolving/qpkktref/maxrounds = 0	presolving/qpkktref/maxrounds = 0
presolving/trivial/maxrounds = 0	presolving/trivial/maxrounds = 0
propagating/dualfix/maxprerounds = 0	propagating/dualfix/maxprerounds = 0
propagating/genvbounds/maxprerounds = 0	propagating/genvbounds/maxprerounds = 0
propagating/obbt/maxprerounds = 0	propagating/obbt/maxprerounds = 0
propagating/nlobbt/maxprerounds = 0	propagating/nlobbt/maxprerounds = 0
propagating/probing/maxprerounds = 0	propagating/probing/maxprerounds = 0
propagating/pseudoobj/maxprerounds = 0	propagating/pseudoobj/maxprerounds = 0
propagating/redcost/maxprerounds = 0	propagating/redcost/maxprerounds = 0
propagating/rootredcost/maxprerounds = 0	propagating/rootredcost/maxprerounds = 0
propagating/vbounds/maxprerounds = 0	propagating/vbounds/maxprerounds = 0
benders/default/lnscheck = FALSE	benders/default/transfercuts = FALSE
benders/default/lnsmaxdepth = 10	benders/default/lnsmaxdepth = 10
constraints/benders/maxprerounds = 0	constraints/benders/maxprerounds = 0
constraints/benderslp/maxprerounds = 0	constraints/benderslp/maxprerounds = 0
Transfer cuts	*Crossover*
presolving/maxrounds = 0	presolving/maxrounds = 0
presolving/maxrestarts = 0	presolving/maxrestarts = 0
propagating/maxrounds = 0	propagating/maxrounds = 0
propagating/maxroundsroot = 0	propagating/maxroundsroot = 0
constraints/nonlinear/maxprerounds = 0	constraints/nonlinear/maxprerounds = 0
constraints/quadratic/maxprerounds = 0	constraints/quadratic/maxprerounds = 0
constraints/linear/maxprerounds = 0	constraints/linear/maxprerounds = 0
constraints/abspower/maxprerounds = 0	constraints/abspower/maxprerounds = 0
constraints/and/maxprerounds = 0	constraints/and/maxprerounds = 0
constraints/bivariate/maxprerounds = 0	constraints/bivariate/maxprerounds = 0
constraints/bounddisjunction/maxprerounds = 0	constraints/bounddisjunction/maxprerounds = 0
constraints/cardinality/maxprerounds = 0	constraints/cardinality/maxprerounds = 0
constraints/conjunction/maxprerounds = 0	constraints/conjunction/maxprerounds = 0
constraints/cumulative/maxprerounds = 0	constraints/cumulative/maxprerounds = 0
constraints/disjunction/maxprerounds = 0	constraints/disjunction/maxprerounds = 0
constraints/indicator/maxprerounds = 0	constraints/indicator/maxprerounds = 0
constraints/knapsack/maxprerounds = 0	constraints/knapsack/maxprerounds = 0
constraints/linking/maxprerounds = 0	constraints/linking/maxprerounds = 0
constraints/logicor/maxprerounds = 0	constraints/logicor/maxprerounds = 0
constraints/or/maxprerounds = 0	constraints/or/maxprerounds = 0
constraints/orbitope/maxprerounds = 0	constraints/orbitope/maxprerounds = 0
constraints/pseudoboolean/maxprerounds = 0	constraints/pseudoboolean/maxprerounds = 0
constraints/setppc/maxprerounds = 0	constraints/setppc/maxprerounds = 0
constraints/soc/maxprerounds = 0	constraints/soc/maxprerounds = 0
constraints/SOS1/maxprerounds = 0	constraints/SOS1/maxprerounds = 0
constraints/SOS2/maxprerounds = 0	constraints/SOS2/maxprerounds = 0
constraints/superindicator/maxprerounds = 0	constraints/superindicator/maxprerounds = 0
constraints/varbound/maxprerounds = 0	constraints/varbound/maxprerounds = 0
constraints/xor/maxprerounds = 0	constraints/xor/maxprerounds = 0
constraints/components/maxprerounds = 0	constraints/components/maxprerounds = 0
reading/storeader/usebenders = TRUE	reading/storeader/usebenders = TRUE
presolving/domcol/maxrounds = 0	presolving/domcol/maxrounds = 0
presolving/dualcomp/maxrounds = 0	presolving/dualcomp/maxrounds = 0
presolving/gateextraction/maxrounds = 0	presolving/gateextraction/maxrounds = 0
presolving/implics/maxrounds = 0	presolving/implics/maxrounds = 0
presolving/inttobinary/maxrounds = 0	presolving/inttobinary/maxrounds = 0
presolving/qpkktref/maxrounds = 0	presolving/qpkktref/maxrounds = 0
presolving/trivial/maxrounds = 0	presolving/trivial/maxrounds = 0
propagating/dualfix/maxprerounds = 0	heuristics/coefdiving/freq = -1
propagating/genvbounds/maxprerounds = 0	heuristics/crossover/freq = -1
propagating/obbt/maxprerounds = 0	heuristics/crossover-benders/freq = 10

Table 12: (continued)

propagating/nlobbt/maxprerounds = 0	heuristics/crossover-benders/nodesofs = 10000
propagating/probing/maxprerounds = 0	heuristics/crossover-benders/maxnodes = 20
propagating/pseudoobj/maxprerounds = 0	heuristics/crossover-benders/minnodes = 0
propagating/redcost/maxprerounds = 0	heuristics/crossover-benders/nwaitingnodes = 0
propagating/rootredcost/maxprerounds = 0	heuristics/crossover-benders/nodesquot = 1
propagating/vbounds/maxprerounds = 0	heuristics/crossover-benders/minfixingrate = 0
benders/default/lnsmaxdepth = 10	heuristics/crossover-benders/lplimfac = 1000
constraints/benders/maxprerounds = 0	heuristics/distributiondiving/freq = -1
constraints/benderslp/maxprerounds = 0	heuristics/feaspump/freq = -1
	heuristics/fracdiving/freq = -1
	heuristics/gins/freq = -1
	heuristics/guideddiving/freq = -1
	heuristics/linesearchdiving/freq = -1
	heuristics/lpface/freq = -1
	heuristics/nlpdiving/freq = -1
	heuristics/objpscostdiving/freq = -1
	heuristics/pscostdiving/freq = -1
	heuristics/rens/freq = -1
	heuristics/rins/freq = -1
	heuristics/rootsoldiving/freq = -1
	heuristics/subnlp/freq = -1
	heuristics/undercover/freq = -1
	heuristics/veclendiving/freq = -1
	propagating/dualfix/maxprerounds = 0
	propagating/genvbounds/maxprerounds = 0
	propagating/obbt/maxprerounds = 0
	propagating/nlobbt/maxprerounds = 0
	propagating/probing/maxprerounds = 0
	propagating/pseudoobj/maxprerounds = 0
	propagating/redcost/maxprerounds = 0
	propagating/rootredcost/maxprerounds = 0
	propagating/vbounds/maxprerounds = 0
	benders/default/transfercuts = FALSE
	benders/default/lnsmaxdepth = 10
	constraints/benders/maxprerounds = 0
	constraints/benderslp/maxprerounds = 0

DINS	*Local Branching*
presolving/maxrounds = 0	presolving/maxrounds = 0
presolving/maxrestarts = 0	presolving/maxrestarts = 0
propagating/maxrounds = 0	propagating/maxrounds = 0
propagating/maxroundsroot = 0	propagating/maxroundsroot = 0
constraints/nonlinear/maxprerounds = 0	constraints/nonlinear/maxprerounds = 0
constraints/quadratic/maxprerounds = 0	constraints/quadratic/maxprerounds = 0
constraints/linear/maxprerounds = 0	constraints/linear/maxprerounds = 0
constraints/abspower/maxprerounds = 0	constraints/abspower/maxprerounds = 0
constraints/and/maxprerounds = 0	constraints/and/maxprerounds = 0
constraints/bivariate/maxprerounds = 0	constraints/bivariate/maxprerounds = 0
constraints/bounddisjunction/maxprerounds = 0	constraints/bounddisjunction/maxprerounds = 0
constraints/cardinality/maxprerounds = 0	constraints/cardinality/maxprerounds = 0
constraints/conjunction/maxprerounds = 0	constraints/conjunction/maxprerounds = 0
constraints/cumulative/maxprerounds = 0	constraints/cumulative/maxprerounds = 0
constraints/disjunction/maxprerounds = 0	constraints/disjunction/maxprerounds = 0
constraints/indicator/maxprerounds = 0	constraints/indicator/maxprerounds = 0
constraints/knapsack/maxprerounds = 0	constraints/knapsack/maxprerounds = 0
constraints/linking/maxprerounds = 0	constraints/linking/maxprerounds = 0
constraints/logicor/maxprerounds = 0	constraints/logicor/maxprerounds = 0
constraints/or/maxprerounds = 0	constraints/or/maxprerounds = 0
constraints/orbitope/maxprerounds = 0	constraints/orbitope/maxprerounds = 0
constraints/pseudoboolean/maxprerounds = 0	constraints/pseudoboolean/maxprerounds = 0
constraints/setppc/maxprerounds = 0	constraints/setppc/maxprerounds = 0
constraints/soc/maxprerounds = 0	constraints/soc/maxprerounds = 0
constraints/SOS1/maxprerounds = 0	constraints/SOS1/maxprerounds = 0
constraints/SOS2/maxprerounds = 0	constraints/SOS2/maxprerounds = 0
constraints/superindicator/maxprerounds = 0	constraints/superindicator/maxprerounds = 0
constraints/varbound/maxprerounds = 0	constraints/varbound/maxprerounds = 0
constraints/xor/maxprerounds = 0	constraints/xor/maxprerounds = 0
constraints/components/maxprerounds = 0	constraints/components/maxprerounds = 0
reading/storeader/usebenders = TRUE	reading/storeader/usebenders = TRUE
presolving/domcol/maxrounds = 0	presolving/domcol/maxrounds = 0
presolving/dualcomp/maxrounds = 0	presolving/dualcomp/maxrounds = 0

Table 12: (continued)

presolving/gateextraction/maxrounds = 0	presolving/gateextraction/maxrounds = 0
presolving/implics/maxrounds = 0	presolving/implics/maxrounds = 0
presolving/inttobinary/maxrounds = 0	presolving/inttobinary/maxrounds = 0
presolving/qpkktref/maxrounds = 0	presolving/qpkktref/maxrounds = 0
presolving/trivial/maxrounds = 0	presolving/trivial/maxrounds = 0
heuristics/coefdiving/freq = -1	heuristics/coefdiving/freq = -1
heuristics/crossover/freq = -1	heuristics/crossover/freq = -1
heuristics/dins-benders/freq = 10	heuristics/distributiondiving/freq = -1
heuristics/dins-benders/nodesofs = 10000	heuristics/feaspump/freq = -1
heuristics/dins-benders/nodesquot = 1	heuristics/fracdiving/freq = -1
heuristics/dins-benders/minnodes = 0	heuristics/gins/freq = -1
heuristics/dins-benders/maxnodes = 20	heuristics/guideddiving/freq = -1
heuristics/dins-benders/nwaitingnodes = 0	heuristics/linesearchdiving/freq = -1
heuristics/dins-benders/lplimfac = 1	heuristics/localbranching-benders/freq = 20
heuristics/dins-benders/minfixingrate = 0	heuristics/localbranching-benders/nodesofs = 10000
heuristics/distributiondiving/freq = -1	heuristics/localbranching-benders/neighborhoodsize = 1
heuristics/feaspump/freq = -1	heuristics/localbranching-benders/nodesquot = 1
heuristics/fracdiving/freq = -1	heuristics/localbranching-benders/minnodes = 0
heuristics/gins/freq = -1	heuristics/localbranching-benders/maxnodes = 20
heuristics/guideddiving/freq = -1	heuristics/localbranching-benders/nwaitingnodes = 0
heuristics/linesearchdiving/freq = -1	heuristics/lpface/freq = -1
heuristics/lpface/freq = -1	heuristics/nlpdiving/freq = -1
heuristics/nlpdiving/freq = -1	heuristics/objpscostdiving/freq = -1
heuristics/objpscostdiving/freq = -1	heuristics/pscostdiving/freq = -1
heuristics/pscostdiving/freq = -1	heuristics/rens/freq = -1
heuristics/rens/freq = -1	heuristics/rins/freq = -1
heuristics/rins/freq = -1	heuristics/rootsoldiving/freq = -1
heuristics/rootsoldiving/freq = -1	heuristics/subnlp/freq = -1
heuristics/subnlp/freq = -1	heuristics/undercover/freq = -1
heuristics/undercover/freq = -1	heuristics/veclendiving/freq = -1
heuristics/veclendiving/freq = -1	propagating/dualfix/maxprerounds = 0
propagating/dualfix/maxprerounds = 0	propagating/genvbounds/maxprerounds = 0
propagating/genvbounds/maxprerounds = 0	propagating/obbt/maxprerounds = 0
propagating/obbt/maxprerounds = 0	propagating/nlobbt/maxprerounds = 0
propagating/nlobbt/maxprerounds = 0	propagating/probing/maxprerounds = 0
propagating/probing/maxprerounds = 0	propagating/pseudoobj/maxprerounds = 0
propagating/pseudoobj/maxprerounds = 0	propagating/redcost/maxprerounds = 0
propagating/redcost/maxprerounds = 0	propagating/rootredcost/maxprerounds = 0
propagating/rootredcost/maxprerounds = 0	propagating/vbounds/maxprerounds = 0
propagating/vbounds/maxprerounds = 0	benders/default/transfercuts = FALSE
benders/default/transfercuts = FALSE	benders/default/lnsmaxdepth = 10
benders/default/lnsmaxdepth = 10	constraints/benders/maxprerounds = 0
constraints/benders/maxprerounds = 0	constraints/benderslp/maxprerounds = 0
constraints/benderslp/maxprerounds = 0	
Proximity	*RINS*
presolving/maxrounds = 0	presolving/maxrounds = 0
presolving/maxrestarts = 0	presolving/maxrestarts = 0
propagating/maxrounds = 0	propagating/maxrounds = 0
propagating/maxroundsroot = 0	propagating/maxroundsroot = 0
constraints/nonlinear/maxprerounds = 0	constraints/nonlinear/maxprerounds = 0
constraints/quadratic/maxprerounds = 0	constraints/quadratic/maxprerounds = 0
constraints/linear/maxprerounds = 0	constraints/linear/maxprerounds = 0
constraints/abspower/maxprerounds = 0	constraints/abspower/maxprerounds = 0
constraints/and/maxprerounds = 0	constraints/and/maxprerounds = 0
constraints/bivariate/maxprerounds = 0	constraints/bivariate/maxprerounds = 0
constraints/bounddisjunction/maxprerounds = 0	constraints/bounddisjunction/maxprerounds = 0
constraints/cardinality/maxprerounds = 0	constraints/cardinality/maxprerounds = 0
constraints/conjunction/maxprerounds = 0	constraints/conjunction/maxprerounds = 0
constraints/cumulative/maxprerounds = 0	constraints/cumulative/maxprerounds = 0
constraints/disjunction/maxprerounds = 0	constraints/disjunction/maxprerounds = 0
constraints/indicator/maxprerounds = 0	constraints/indicator/maxprerounds = 0
constraints/knapsack/maxprerounds = 0	constraints/knapsack/maxprerounds = 0
constraints/linking/maxprerounds = 0	constraints/linking/maxprerounds = 0
constraints/logicor/maxprerounds = 0	constraints/logicor/maxprerounds = 0
constraints/or/maxprerounds = 0	constraints/or/maxprerounds = 0
constraints/orbitope/maxprerounds = 0	constraints/orbitope/maxprerounds = 0
constraints/pseudoboolean/maxprerounds = 0	constraints/pseudoboolean/maxprerounds = 0
constraints/setppc/maxprerounds = 0	constraints/setppc/maxprerounds = 0
constraints/soc/maxprerounds = 0	constraints/soc/maxprerounds = 0
constraints/SOS1/maxprerounds = 0	constraints/SOS1/maxprerounds = 0

Table 12: (continued)

constraints/SOS2/maxprerounds = 0	constraints/SOS2/maxprerounds = 0
constraints/superindicator/maxprerounds = 0	constraints/superindicator/maxprerounds = 0
constraints/varbound/maxprerounds = 0	constraints/varbound/maxprerounds = 0
constraints/xor/maxprerounds = 0	constraints/xor/maxprerounds = 0
constraints/components/maxprerounds = 0	constraints/components/maxprerounds = 0
reading/storeader/usebenders = TRUE	reading/storeader/usebenders = TRUE
presolving/domcol/maxrounds = 0	presolving/domcol/maxrounds = 0
presolving/dualcomp/maxrounds = 0	presolving/dualcomp/maxrounds = 0
presolving/gateextraction/maxrounds = 0	presolving/gateextraction/maxrounds = 0
presolving/implics/maxrounds = 0	presolving/implics/maxrounds = 0
presolving/inttobinary/maxrounds = 0	presolving/inttobinary/maxrounds = 0
presolving/qpkktref/maxrounds = 0	presolving/qpkktref/maxrounds = 0
presolving/trivial/maxrounds = 0	presolving/trivial/maxrounds = 0
heuristics/coefdiving/freq = -1	heuristics/coefdiving/freq = -1
heuristics/crossover/freq = -1	heuristics/crossover/freq = -1
heuristics/distributiondiving/freq = -1	heuristics/distributiondiving/freq = -1
heuristics/feaspump/freq = -1	heuristics/feaspump/freq = -1
heuristics/fracdiving/freq = -1	heuristics/fracdiving/freq = -1
heuristics/gins/freq = -1	heuristics/gins/freq = -1
heuristics/guideddiving/freq = -1	heuristics/guideddiving/freq = -1
heuristics/linesearchdiving/freq = -1	heuristics/linesearchdiving/freq = -1
heuristics/lpface/freq = -1	heuristics/lpface/freq = -1
heuristics/nlpdiving/freq = -1	heuristics/nlpdiving/freq = -1
heuristics/objpscostdiving/freq = -1	heuristics/objpscostdiving/freq = -1
heuristics/proximity-benders/freq = 5	heuristics/pscostdiving/freq = -1
heuristics/proximity-benders/maxnodes = 20	heuristics/rens/freq = -1
heuristics/proximity-benders/nodesofs = 10000	heuristics/rins-benders/freq = 10
heuristics/proximity-benders/minlpiters = 10000	heuristics/rins-benders/nodesofs = 10000
heuristics/proximity-benders/waitingnodes = 0	heuristics/rins-benders/maxnodes = 20
heuristics/proximity-benders/minimprove = 0.01	heuristics/rins-benders/minnodes = 0
heuristics/proximity-benders/nodesquot = 10000	heuristics/rins-benders/nodesquot = 1
heuristics/proximity-benders/binvarquot = 0	heuristics/rins-benders/nwaitingnodes = 0
heuristics/proximity-benders/lpitersquot = 1	heuristics/rins-benders/minfixingrate = 0
heuristics/pscostdiving/freq = -1	heuristics/rins-benders/lplimfac = 1
heuristics/rens/freq = -1	heuristics/rins/freq = -1
heuristics/rins/freq = -1	heuristics/rootsoldiving/freq = -1
heuristics/rootsoldiving/freq = -1	heuristics/subnlp/freq = -1
heuristics/subnlp/freq = -1	heuristics/undercover/freq = -1
heuristics/undercover/freq = -1	heuristics/veclendiving/freq = -1
heuristics/veclendiving/freq = -1	propagating/dualfix/maxprerounds = 0
propagating/dualfix/maxprerounds = 0	propagating/genvbounds/maxprerounds = 0
propagating/genvbounds/maxprerounds = 0	propagating/obbt/maxprerounds = 0
propagating/obbt/maxprerounds = 0	propagating/nlobbt/maxprerounds = 0
propagating/nlobbt/maxprerounds = 0	propagating/probing/maxprerounds = 0
propagating/probing/maxprerounds = 0	propagating/pseudoobj/maxprerounds = 0
propagating/pseudoobj/maxprerounds = 0	propagating/redcost/maxprerounds = 0
propagating/redcost/maxprerounds = 0	propagating/rootredcost/maxprerounds = 0
propagating/rootredcost/maxprerounds = 0	propagating/vbounds/maxprerounds = 0
propagating/vbounds/maxprerounds = 0	benders/default/transfercuts = FALSE
benders/default/transfercuts = FALSE	benders/default/lnsmaxdepth = 10
benders/default/lnsmaxdepth = 10	constraints/benders/maxprerounds = 0
constraints/benders/maxprerounds = 0	constraints/benderslp/maxprerounds = 0
constraints/benderslp/maxprerounds = 0	