Simultaneously exploiting two formulations: An exact benders decomposition approach

Lusby, Richard Martin ; Gamst, Mette; Røpke, Stefan; Spoorendonk, Simon

Published in:
Computers and Operations Research

Link to article, DOI:
10.1016/j.cor.2020.105041

Publication date:
2020

Document Version
Peer reviewed version

Simultaneously Exploiting Two Formulations: an Exact Benders Decomposition Approach

Richard Martin Lusby[a,*], Mette Gamst[b], Stefan Ropke[a], Simon Spoorendonk[c]

[a] *Department of Technology Management and Economics, Technical University of Denmark*
[b] *Energinet.dk, Tonne Kjærsvej 65, DK-7000, Fredericia, Denmark*
[c] *Flowty, Copenhagen, Denmark*

Abstract

When modelling a given problem using integer linear programming techniques several possibilities often exist, each resulting in a different mathematical formulation of the problem. Usually, advantages and disadvantages can be identified in any single formulation. In this paper we consider mixed integer linear programs and propose an approach based on Benders decomposition to exploit the advantages of two different formulations when solving a problem. We propose applying Benders decomposition to a combined formulation, comprised of two separate formulations, augmented with linking constraints to ensure consistency between the decision variables of the respective formulations. We demonstrate the applicability of the proposed methodology to situations in which one of the formulations models a relaxation of the problem and to cases where one formulation is the Dantzig-Wolfe reformulation of the other. The proposed methodology guarantees a lower bound that is as good as the tighter of the two formulations, and we show how branching can be performed on the decision variables of either formulation. Finally, we test and compare the performance of the proposed approach on publicly available instances of the Cutting Stock Problem and the Split Delivery Vehicle Routing Problem. Compared to the best approaches from the literature, the proposed method shows promising performance and appears to be an attractive alternative.

Keywords: Mixed Integer Programming, Benders Decomposition, Cutting Stock Problem, Split Delivery Vehicle Routing

1. Introduction

With their high degree of applicability to a wide range of practical problems, integer linear programming methods are some of the most extensively used techniques in the Operations Research (OR) field. An Integer Linear Program (ILP) can be succinctly stated as $\min\{c^T x : Ax \geq b, x \in \mathbb{Z}^n_{\geq 0}\}$, where x is an n-dimensional vector of integer decision variables. Matrices A, b, and c are of compatible dimension and contain known data. The aim of such problems is to minimize the value of $c^T x$ while satisfying the linear constraint system and restricting the decision variables to discrete values. If only a subset of the decision variables are restricted to integer values, the problem is termed a Mixed Integer Linear Program (MILP).

When modelling a given problem using mixed integer linear programming techniques, a variety of possibilities usually exist, and the resulting mathematical formulations can have very different performance from a computational perspective. Furthermore, such formulations can quickly become intractable as the number of constraints and variables increases. Research addressing the solvability of MILPs therefore focuses on techniques such as problem relaxation and problem reformulation to produce a simpler, perhaps more computationally attractive formulation.

*Corresponding author
Email address: rmlu@dtu.dk (Richard Martin Lusby)

In a relaxation of a problem, some information is omitted. The resulting formulation typically has fewer dimensions and/or constraints, and is hence easier to solve. As some information is lost, the relaxation may permit solutions which are not feasible for the original problem. However, any solution to the original problem must also be feasible for the relaxation, and hence statements regarding solution quality, in the form of a lower bound on the objective function value (minimization problem), can be made for the original model. Examples of relaxation approaches include Lagrangian Relaxation, see, e.g., Fisher (1973); Held and Karp (1971), and methods that aggregate multiple variables into fewer variables, see, e.g., Belenguer et al. (2000). The former is an attempt to reduce complexity by removing difficult constraints from the problem, while the latter can be effective at reducing the dimension of the problem.

Problem reformulation on the other hand, attempts to produce a MILP which provides a *tighter* description of the problem; in other words it attempts to produce a formulation for which the gap between the optimal objective values of the linear programming relaxation and that of the MILP is smaller than the original model. A reformulation is therefore not a relaxation since every feasible solution to the reformulation must also be feasible for the original problem. Complications can, however, arise in a Branch-and-Bound (BAB) context, where it may not be trivial to enforce the required branches in the reformulation. Reformulations can be produced through problem decomposition, and well known decomposition techniques include, among others, Dantzig-Wolfe (DW) decomposition, see, e.g., Dantzig and Wolfe (1960), and Benders decomposition, see, e.g., Benders (1962). These approaches naturally lend themselves to advanced algorithms, such as Branch-and-Price (BAP) and Branch-and-Cut-and-Price (BCP). These algorithms have gained much attention in state-of-the-art research as they tend to show superior performance across a wide range of applications, see e.g., Lübbecke and Desrosiers (2005).

Both relaxation and reformulation through decomposition can be effective ways at improving the solvability of difficult MILPs; however, each approach also has its disadvantages. Relaxations do not necessarily guarantee feasibility, while reformulations might, for example, complicate branching procedures. In this paper we therefore address the question of whether or not it is possible to simultaneously exploit the structures of two formulations when solving a MILP, and propose a method to achieve this. Branching in relaxations and reformulations is a well-known challenge in OR; almost all work in the area to date uses application specific branching, see, e.g., Belov (2003); Truffot and Duhamel (2008); Valério de Carvalho (1999). Some more generic branching strategies, however, have also been proposed. Branching on the original variables of DW decompositions has been considered by Villeneuve et al. (2005). The authors argue that the original formulation from which DW decomposition is obtained can always be derived from a master and a pricing problem. Branching can thus be performed in the original formulation, which is then re-decomposed into an equivalent master and pricing problem. Branching rules on the original variables will be present in the master or in the pricing problem; either way, branching may change the pricing problem through additional constraints, or through a change in the reduced cost function. A generic branching strategy for BAP algorithms can be found in Vanderbeck (2011). The strategy involves branching on original variables similar to Villeneuve et al. (2005), however, in such a way that it can be enforced through fixing variable bounds in the pricing problem. This imposes only few changes to the pricing problem variable bounds and allows for early pruning of parts of the subtree. Furthermore, it reduces symmetry in the BAB tree.

In this paper, we propose a general method for exploiting structures in two formulations using Benders decomposition. We propose applying Benders decomposition to a combined formulation, comprised of two separate formulations, augmented with linking constraints to ensure consistency between the decision variables of the respective formulations. In particular, a relaxation/reformulation is defined to be the master problem, while the original formulation and linking constraints constitute the subproblem. Given a fractional, optimal solution to the master problem, we show how branching can be performed in the subproblem. This can be particularly advantageous if the master problem is solved with column generation; by branching on the variables of the original formulation in the Benders subproblem, no modification to the solution polytope of the column generation pricing problem is necessary. The reduced costs are modified to reflect the branching strategy. This, coupled with the fact that the proposed approach also provides

a framework for guaranteeing feasibility of the original problem by solving a relaxation, means the proposed work complements that of Vanderbeck (2011) and Villeneuve et al. (2005). Neither studies consider relaxations, nor do their branching strategies leave the solution polytope of the pricing problem unchanged. Deriving cuts for one formulation using another has been considered previously, see e.g., Applegate et al. (2003); Ralphs et al. (2003). Applegate et al. (2003) generate infeasibility cuts for an edge based formulation of the Traveling Salesman Problem by transforming an Linear Programming (LP) solution of the original formulation to a solution in a lower dimensional space. This is done by clustering vertices and aggregating the values of edge variables that are connecting clusters. If the solution in the lower dimensional space is a convex combination of feasible solutions to the *graphical traveling salesman problem*, then no cut can be derived. The graphical travelling salesman problem involves determining the minimum length tour in which each customer is visited *at least* once. If, however, the solution in the lower dimensional space cannot be expressed as a convex combination of feasible solutions to the graphical travelling salesman problem, then a cut separating the transformed solution and the convex hull of feasible solutions to the graphical traveling salesman problem is found and this cut is translated back to the original formulation. Similarly, Ralphs et al. (2003) derive cuts for an edge based formulation of the Capacitated Vehicle Routing Problem using a corresponding path based formulation. In particular, Farkas Lemma is used to construct an infeasibility cut if a solution to the LP relaxed edge based model is not feasible for the path based representation.

The methodology proposed in this paper exploits the computational superiority of relaxations and reformulations, but provides a finite branching strategy based on the original variables. As the full formulation essentially consists of two separate formulations, the method provides an LP bound, which is as good as the tighter of the two. In this paper we test the approach on the Cutting-Stock Problem (CSP) and the Split Delivery Vehicle Routing Problem (SDVRP) and report promising results.

This paper is structured as follows. Section 2 introduces some required terminology and definitions. Section 3 provides a detailed description of the proposed decomposition approach, while Sections 4 and 5 focus on the computational performance of the proposed algorithm. In particular, Section 4 provides a detailed analysis of how the methodology performs on the CSP, while Section 5 looks at the SDVRP. For both problems, our generic approach is compared to efficient, problem specific methodologies from the literature. Conclusions and directions for future work are summarized in Section 6.

2. Definitions

To assist in the explanation of the approach, we begin by defining two generic ILP formulations. The first formulation, denoted $\mathcal{P}1$, also referred to as the original model, is the model from which formulation $\mathcal{P}2$, *either* a relaxation *or* a reformulation, is obtained. Formulation $\mathcal{P}2$ is assumed to be an attractive model for solving the original model from the perspective that its LP relaxation is tight or easy to compute. However, it is also assumed to have some disadvantages. For example, it might be a relaxation, or it could be that branching in $\mathcal{P}2$ is not so straightforward. In this paper we restrict our focus to ILPs; however, the proposed methodology also extends to MILPs. To make the distinction between decision variables and parameters in any formulations we state, variables will appear in boldface. Formulation $\mathcal{P}1$ is assumed to have n_1 integer decision variables, the values of which are stored in the vector $\boldsymbol{x}$. Formulation $\mathcal{P}2$, on the other hand, is assumed to have n_2 integer decision variables, and its solution is given by the vector $\boldsymbol{y}$. The formulations do not need to, and will most likely not have, the same dimension. We assume that $\mathcal{P}1$ has m_1 constraints, while $\mathcal{P}2$ has m_2 constraints. Vectors c, f, b_1 and b_2, along with matrices A_1 and A_2, are all of compatible dimensions and contain the known parameters (objective coefficients, right hand side values, and constraint coefficients) of the two models.

$$(\mathcal{P}1): \texttt{minimize } c^T\boldsymbol{x}, \quad s.t. \quad A_1\boldsymbol{x} \geq b_1, \quad \boldsymbol{x} \in \mathbb{Z}^{n_1}_{\geq 0}.$$

$$(\mathcal{P}2): \texttt{minimize } f^T\boldsymbol{y}, \quad s.t. \quad A_2\boldsymbol{y} \geq b_2, \quad \boldsymbol{y} \in \mathbb{Z}^{n_2}_{\geq 0}.$$

We denote the sets of feasible solutions to $\mathcal{P}1$ and $\mathcal{P}2$ as $\mathcal{X}$ and $\mathcal{Y}$, respectively. Furthermore, we assume an affine transformation $D\boldsymbol{x} = W\boldsymbol{y}$ (where matrices D and W are of compatible dimension) exists and maps the variables of $\mathcal{P}1$ to variables in $\mathcal{P}2$. That is, for any $\boldsymbol{x}' \in \mathcal{X}$, there exists a solution $\boldsymbol{y}' \in \mathcal{Y}$ such that $D\boldsymbol{x}' = W\boldsymbol{y}'$ with $c^T\boldsymbol{x}' = f^T\boldsymbol{y}'$. We use $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ to denote the optimal solutions to $\mathcal{P}1$ and $\mathcal{P}2$, respectively. To formalize the definitions of relaxation and reformulation, we introduce the set Q and its projection on the $\boldsymbol{y}$ space. These are defined as follows.

$$Q := \{(\boldsymbol{x}, \boldsymbol{y}\} \in \mathbb{Z}^{n_1}_{\geq 0} \times \mathbb{Z}^{n_2}_{\geq 0} : A_2\boldsymbol{y} \geq b_2, D\boldsymbol{x} = W\boldsymbol{y}\},$$
$$Proj_y(Q) := \{\boldsymbol{y} \in \mathbb{Z}^{n_2}_{\geq 0} \ : \exists \boldsymbol{x} \in \mathbb{Z}^{n_1}_{\geq 0} : (\boldsymbol{x}, \boldsymbol{y}) \in Q\}.$$

Definition 2.1. $\mathcal{P}2$ is a relaxation of $\mathcal{P}1$ if the following two conditions are satisfied:

1. $Proj_y(Q) \subseteq \mathcal{Y}$,
2. $f^T\boldsymbol{y}^* \leq c^T\boldsymbol{x}^*$.

The first point states that there are potentially more solutions to $\mathcal{P}2$ than $\mathcal{P}1$. In other words, there may exist solutions to $\mathcal{P}2$ that are not feasible for $\mathcal{P}1$. As some information is omitted when relaxing the problem, this statement is quite intuitive. The second point follows naturally from the first; with a greater number of feasible solutions, the optimal objective value of $\mathcal{P}2$ yields a lower bound on the optimal objective value of $\mathcal{P}1$.

Definition 2.2. $\mathcal{P}2$ is a reformulation of $\mathcal{P}1$ if the following three conditions are satisfied:

1. $Proj_y(Q) = \mathcal{Y}$,
2. $f^T\boldsymbol{y}^* = c^T\boldsymbol{x}^*$,
3. $f^T\boldsymbol{y}^*_{LP} \geq c^T\boldsymbol{x}^*_{LP}$.

The first point states that any solution $\boldsymbol{y}' \in \mathcal{Y}$ has a corresponding solution $\boldsymbol{x}' \in \mathcal{X}$ with $W\boldsymbol{y}' = D\boldsymbol{x}'$. Recall the assumption above, for any $\boldsymbol{x}' \in \mathcal{X}$, there exists a solution $\boldsymbol{y}' \in \mathcal{Y}$ such that $D\boldsymbol{x}' = W\boldsymbol{y}'$ with $c^T\boldsymbol{x}' = f^T\boldsymbol{y}'$. Together with point one this implies point two, which states that the optimal objective values of $\mathcal{P}2$ and $\mathcal{P}1$ are the same. The last point states that we are only interested in reformulations which have an LP relaxation bound that is at least as good as that of $\mathcal{P}1$; $\boldsymbol{y}^*_{LP}$ and $\boldsymbol{x}^*_{LP}$ are used to denote the optimal solutions to the LP relaxations of $\mathcal{P}2$ and $\mathcal{P}1$, respectively.

We conclude this section by identifying two examples of problems where the proposed methodology could be applied. The first is the CSP and is an example where $\mathcal{P}2$ would be a reformulation, while the second is the SDVRP and is an example where $\mathcal{P}2$ would be a relaxation. In both cases, we highlight the advantages and disadvantages associated with each of the two formulations, $\mathcal{P}1$ and $\mathcal{P}2$, that arise. An analysis of how the methodology performs on variants of these two problems is given in Sections 4 and 5, respectively.

The CSP was first introduced in Kantorovich (1960) and involves determining the minimum number of rolls (or stocks) of a given length that must be cut in order to meet the demand for a set of items I, of shorter, specified lengths. It is often modelled using the Gilmore-Gomory formulation, given in Gilmore and Gomory (1963). This formulation has a set-cover like form and is characterized by integer pattern variables, $\boldsymbol{y}_p$. A pattern corresponds to a possible cutting of the stock into a subset of the items, while the value of a variable states the number of patterns of that type to cut. Due to the large number of patterns possible, column generation is used to dynamically generate good patterns. The generation of a pattern for a given stock length involves solving an

integer knapsack problem. Eliminating fractionality typically involves branching on items and this can either be enforced directly in the pattern formulation through additional constraints on the $\boldsymbol{y}_p$ variables, or implicitly in the pattern construction process. The latter induces modifications in the subproblem, which may impact the performance of the pattern generation step, see, e.g., Alves and De Carvalho (2008). In a column generation setting, branching constraints on the $\boldsymbol{y}_p$ variables also induce changes. The branching constraints introduce additional dual variables which must be accounted for in the pattern generation step. Alternatively, the CSP can be modelled using an arc-flow formulation, see Valério de Carvalho (1999). The number of vertices in the underlying network is equal to the stock length plus one, where each vertex represents a discrete position along the stock. Two types of arcs are used to represent cutting items and wasted stock length. As an example, arc (i, j) indicates the cutting of an item of length $j - i$ where the cut starts at a length i units from the start of the stock. Integer decision variables $\boldsymbol{x}_{ij} \in \mathbb{Z}_{\geq 0}$ state the number of items of length $j - i$ starting at position i. Such a modelling approach for this problem results in significant amounts of symmetry and is much less tractable than a reformulation using the pattern based formulation; however, it is perhaps preferable from a branching perspective. We postpone an overview of the full formulation until Section 4. Our intention here is to highlight from a motivational perspective how the two formulations can be connected. We demonstrate in Section 3, that it is possible to explicitly consider both formulations. The following affine transformation is used:

$$\sum_{p \in P} a_{ip} \boldsymbol{y}_p = \sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i} \quad \forall i \in I, \tag{1}$$

where a_{ip} denotes the number of times item $i \in I$ occurs in pattern $p \in P$, w_i is the length of item i, and A is the arc set.

The SDVRP was first introduced in Dror and Trudeau (1990) and is a variant of the well known vehicle routing problem. The problem involves finding the minimum cost set of routes for a set of vehicles K through a set of customers, where the demand of a customer can be satisfied through multiple visits. A common approach is to define binary variables $\boldsymbol{x}_{ij}^k$ that indicate whether or not vehicle $k \in K$ drives from customer $i \in V$ to customer $j \in V$ along edge (i, j), see, e.g., Jin et al. (2007); Lee et al. (2006). Such formulations are, however, difficult to solve to optimality due to symmetry. A relaxation can be achieved through aggregating the binary $\boldsymbol{x}_{ij}^k$ variables into integer variables $\boldsymbol{y}_{ij}$, which count the number of vehicles travelling between customers i and j (in both directions), see Belenguer et al. (2000); Archetti et al. (2014). The latter are more appealing from a computational perspective, but may not necessarily be feasible when trying to produce individual vehicle routes from the aggregated variables. To exploit the simpler relaxation and the feasibility provided by the $\boldsymbol{x}_{ij}^k$ variables we therefore suggest to simultaneously consider both formulations. The decision variables of the original problem are mapped to the decision variables of the relaxation with the following affine transformation:

$$\boldsymbol{y}_{ij} = \sum_{k \in k} (\boldsymbol{x}_{ij}^k + \boldsymbol{x}_{ji}^k) \quad \forall (i, j) \in E, \tag{2}$$

where $E = \{(i, j) : i < j, i \in V, j \in V\}$.

3. Solution Approach

This section focuses on the proposed solution approach. The main motivation behind this research is the desire to be able to exploit the benefits of two different formulations of a MILP. For example, formulation $\mathcal{P}2$ might be nice from a tractability perspective; relaxations are typically smaller in size than the original formulation, while reformulations lend themselves to decomposition methodologies. Formulation $\mathcal{P}1$ may, however, allow for easier branching implementations. We therefore, suggest to combine formulations $\mathcal{P}1$ and $\mathcal{P}2$ into one formulation, denoted $\mathcal{P}3$, and augment this with the linking constraints $D\boldsymbol{x} = W\boldsymbol{y}$ to force consistency between the two different sets of decision variables.

$$
\begin{aligned}
(\mathcal{P}3): \texttt{minimize}\ & f^T \boldsymbol{y}, && (3)\\
s.t.\quad & A_1 \boldsymbol{x} \geq b_1, && (4)\\
& A_2 \boldsymbol{y} \geq b_2, && (5)\\
& D\boldsymbol{x} = W\boldsymbol{y}, && (6)\\
& \boldsymbol{x} \in \mathbb{Z}^{n_1}_{\geq 0}, && (7)\\
& \boldsymbol{y} \in \mathbb{Z}^{n_2}_{\geq 0}. && (8)
\end{aligned}
$$

All of the components of $\mathcal{P}3$ have already been introduced. The formulation includes both sets of decision variables $\boldsymbol{x}$ and $\boldsymbol{y}$. The objective function, (3) and Constraints (5) and Constraints (8) come from $\mathcal{P}2$, while Constraints (4) and Constraints (7) come from $\mathcal{P}1$. Constraints (6) link the two formulations together. In what follows, we demonstrate that $\mathcal{P}3$ naturally lends itself to a Benders decomposition approach, where the aim is to solve formulation $\mathcal{P}2$ while restricting the search to solutions $\boldsymbol{x} \in \mathcal{X}$ that can be mapped, using Constraints (6) to a solution, in $\mathcal{Y}$. We now prove some characteristics of $\mathcal{P}3$.

Proposition 3.1. *Consider the formulations $\mathcal{P}1$, $\mathcal{P}2$ and $\mathcal{P}3$. Then:*

$$z^{\mathcal{P}3}_{LP} \geq \max\{z^{\mathcal{P}1}_{LP}, z^{\mathcal{P}2}_{LP}\},$$

where $z^{\mathcal{P}1}_{LP}, z^{\mathcal{P}2}_{LP}$, and $z^{\mathcal{P}3}_{LP}$ are the optimal objective values to the three formulations, respectively.

Proof. Formulation $\mathcal{P}3$ includes all constraints and variables of formulations $\mathcal{P}1$ and $\mathcal{P}2$. In other words, $\mathcal{P}1$ and $\mathcal{P}2$ are relaxations of $\mathcal{P}3$. The optimal objective value can thus not be smaller than $z^{\mathcal{P}1}_{LP}$ or $z^{\mathcal{P}2}_{LP}$. □

Proposition 3.2. *Consider the formulations $\mathcal{P}1, \mathcal{P}2$, and $\mathcal{P}3$. Then:*

$$z^{\mathcal{P}1} = z^{\mathcal{P}3},$$

where $z^{\mathcal{P}1}$ and $z^{\mathcal{P}3}$ are the optimal objective values to formulations $\mathcal{P}1$ and $\mathcal{P}3$.

Proof. Any solution $\boldsymbol{x}' \in \mathcal{X}$ has a corresponding solution $\boldsymbol{y}' \in \mathcal{Y}$ such that $D\boldsymbol{x}' = W\boldsymbol{y}'$. By assumption, we have that $c^T\boldsymbol{x}' = f^T\boldsymbol{y}'$. Thus if $\boldsymbol{x}'$ is an optimal solution to $\mathcal{P}1$, $(\boldsymbol{x}', \boldsymbol{y}')$ is an optimal solution for $\mathcal{P}3$.

□

Benders decomposition, see Benders (1962), is a solution method that is designed to exploit MILPs having a so-called *Dual Block Angular Structure*. Problems with this structure decompose into independent problems once a set of "complicating" variables is fixed. One can observe that this structure is, by construction, inherent in $\mathcal{P}3$. If the values of $\boldsymbol{y}$ are known, then all that remains is to solve a feasibility problem over $\boldsymbol{x}$ with the values of $\boldsymbol{y}$ fixed. Essentially $\mathcal{P}3$ can then be seen as two problems. One is an optimization problem determining the $\boldsymbol{y}$ values, while the other is a feasibility problem that assesses the feasibility of the $\boldsymbol{y}$ solution with respect to the $\boldsymbol{x}$ variables. Benders decomposition therefore reformulates the MILP as a *master problem* and one or more independent *subproblems*. The role of a subproblem is to assess the optimality, not to mention the feasibility, of a given solution to the master problem. If the master solution results in an infeasible subproblem, then a *feasibility cut* for the master problem can be generated. On the other hand, if the solution to the master problem results in a feasible subproblem, an *optimality cut* for the master problem can be generated if the master problem solution is suboptimal for the original MILP. In both cases, each cut provokes a new solution when added to the master problem. The classical Benders decomposition algorithm is therefore an iterative procedure between the master problem and the subproblems and continues as long as cuts are returned from the subproblems. In the Benders reformulation of a MILP integer variables are placed in the master problem, while

continuous variables are placed in the subproblems (a prerequisite to obtain a dual solution). One can therefore view classical Benders as a sequence of ILPs, where the ILPs typically get larger (through the addition of optimality and feasibility cuts) and hence take longer to solve as the Benders iterations continue.

Branch-and-Benders-cut (BBC) is an alternative implementation of Benders decomposition that preserves optimality, see e.g, Fortz and Poss (2009). The main difference to the iterative procedure is that this approach views Benders decomposition as more of a traditional Branch-and-Cut (BAC) algorithm. As such, there is only a single BAB search tree for the master problem, and feasibility and optimality cuts are generated from any solution encountered in this tree. In other words, cuts are generated from both fractional and integer solutions. This implementation of Benders decomposition has been shown to be computationally superior to that of the iterative approach, see Naoum-Sawaya and Elhedhli (2013).

There are several points to be aware of when directly applying Benders decomposition to formulation $\mathcal{P}3$. The first is that all $\boldsymbol{x}$ variables could have integrality restrictions. To address this, we propose an extension of the BBC approach, whereby branching on a subproblem variable simply creates two child nodes. In each branch the master problem is inherited in its entirety and one additional constraint is included in the subproblem (reflecting the new bound on the chosen variable). Secondly, if $\mathcal{P}2$ is a reformulation, then it is not necessary to solve the Benders subproblem on finding an integer feasible solution to $\mathcal{P}2$. Any integer solution to the reformulation is also feasible for the original problem. One of the main aims of coupling $\mathcal{P}1$ and $\mathcal{P}2$ is to allow the possibility to branch in the $\boldsymbol{x}$ variable space. The BBC approach preserves this possibility. In other words, given a fractional solution to the master problem at a node of the BAB tree, instead of branching on $\boldsymbol{y}$ variables, one can branch on $\boldsymbol{x}$ variables in the subproblem. Finally, observe that we only separate feasibility cuts as the cost is only associated with master problem variables.

We now try to formalize our approach. Accepting for the moment integer subproblems are possible within a Benders Decomposition framework, we identify the following master problem $\mathcal{MP}$ and subproblem $\mathcal{SP}$ from formulation $\mathcal{P}3$:

$$(\mathcal{MP}) : \texttt{minimize} \quad f^T \boldsymbol{y}, \tag{3}$$
$$s.t. \quad A_2 \boldsymbol{y} \geq b_2, \tag{5}$$
$$\boldsymbol{y} \in \mathbb{Z}^{n_2}_{\geq 0}. \tag{8}$$

$$(\mathcal{SP}) : \texttt{minimize} \quad 0^T \boldsymbol{x}, \tag{9}$$
$$s.t. \quad A_1 \boldsymbol{x} \geq b_1, \tag{4}$$
$$D\boldsymbol{x} = W\bar{y}, \tag{6}$$
$$\boldsymbol{x} \in \mathbb{Z}^{n_1}_{\geq 0}. \tag{7}$$

Observe that $\mathcal{MP}$ is just formulation $\mathcal{P}2$. This is because we would like to exploit this formulation as much as possible. Given a feasible solution $\bar{y}$ to $\mathcal{MP}$, the role of the subproblem is to check if $\bar{y}$ is feasible for the original problem.

To solve $\mathcal{P}3$, we begin by relaxing the integrality restrictions of both problems, obtaining $\mathcal{MP}_{LP}$ and $\mathcal{SP}_{LP}$. This involves changing the requirement $\boldsymbol{y} \in \mathbb{Z}^{n_2}_{\geq 0}$ to $\boldsymbol{y} \in \mathbb{R}^{n_2}_{\geq 0}$ in $\mathcal{MP}$ and replacing $\boldsymbol{x} \in \mathbb{Z}^{n_1}_{\geq 0}$ with $\boldsymbol{x} \in \mathbb{R}^{n_1}_{\geq 0}$ in $\mathcal{SP}_{LP}$. We then apply a modified version of the BBC method. We start by solving the root node of $\mathcal{P}2$; this is an iterative procedure that utilizes $\mathcal{SP}_{LP}$ to see if the solution to $\mathcal{MP}_{LP}$ is feasible. If it is not feasible, using the theory of Benders decomposition, the following feasibility cut can be added to $\mathcal{MP}_{LP}$:

$$u_1^T b_1 + u_2^T W \boldsymbol{y} \leq 0, \tag{10}$$

where $u_1 \in \mathbb{R}_{m_1}$ and $u_2 \in \mathbb{R}_{m_2}$ are the dual variables associated with Constraints (4) and Con-

straints (6) of $\mathcal{SP}_{LP}$ and collectively define an extreme dual ray. Ultimately, this iterative procedure will yield the optimal solution to the LP relaxation of $\mathcal{P}3$, $(\boldsymbol{y}^*_{lp}, \boldsymbol{x}^*_{lp})$. If this solution is fractional, we can now branch on disjunctions in the $\boldsymbol{y}$ variable space or the $\boldsymbol{x}$ variable space to reach integer feasibility. We propose to branch on $\boldsymbol{x}$ variables as an optimal integer solution for $\mathcal{P}1$ is needed to solve the problem in the case $\mathcal{P}2$ is a relaxation, and in the case of reformulations it is generally easier to enforce. The latter is true for the CSP we consider in Section 4. Given a variable $\boldsymbol{x}_i$ with fractional solution value v, the disjunctions are enforced in $\mathcal{SP}_{LP}$ by adding constraints

$$\boldsymbol{x}_i \leq \lfloor v \rfloor \vee \boldsymbol{x}_i \geq \lceil v \rceil.$$

Such a branching strategy generates two new nodes for $\mathcal{MP}_{LP}$. Note further that this approach *partitions* the solution space into two disjoint subsets and thus preserves optimality. In the first node $\mathcal{SP}_{LP}$ is modified to include $\boldsymbol{x}_i \leq \lfloor v \rfloor$ while in the second, $\mathcal{SP}_{LP}$ is modified to include $\boldsymbol{x}_i \geq \lceil v \rceil$. Such additions force changes in $\boldsymbol{x}^*_{lp}$ and may prompt changes in $\boldsymbol{y}_{lp}$ through new feasibility cuts. When $\mathcal{P}2$ is a reformulation of $\mathcal{P}1$, feasibility cuts of the form (10) are generated when branching in the $\boldsymbol{x}$ variable space leads to infeasibility for the Benders subproblem. This is also true when $\mathcal{P}2$ is a relaxation of $\mathcal{P}1$; however, for this case feasibility cuts can also be generated when a solution to $\mathcal{P}2$ cannot be transformed to a feasible solution to $\mathcal{P}1$. The BBC procedure then continues as long as there are unsolved nodes in the BAB tree.

From a computational perspective, the dual formulation of $\mathcal{SP}_{LP}$ is usually solved in the implementation of Benders Decomposition. This leaves the solution polyhedron for $\mathcal{SP}_{LP}$ unchanged. Any new changes in variable bounds appear as modifications to the objective function only. In fact, the resulting variable bounds only affect the constant term of Constraints (10). Branching in the $\boldsymbol{x}$ variable space therefore implicitly enforces the disjunctions in the $\boldsymbol{y}$ variable space. As such, integrality of the $\boldsymbol{y}$ is strictly not necessary as it is imposed by the $\boldsymbol{x}$ variables. Depending on the situation, however, it might be preferable to branch on the y variables, or even a hybrid strategy. The latter is useful when, for example, $\mathcal{P}2$ is a relaxation of $\mathcal{P}1$. Other branching strategies can also be applied, see e.g., Mak (2007). A complete overview of the procedure is provided in Algorithm 1.

Essentially the algorithm maintains an upper bound on solution value, ub, and processes a sequence of nodes in the BAB tree. We denote the objective value of a solution to $\mathcal{MP}_{LP}$ as z. Feasibility cuts are used to cut away $\mathcal{MP}_{LP}$ solutions that when fixed in $\mathcal{SP}_{LP}$ result in infeasibility (see line 14). Branching is performed on disjunctions in the $\boldsymbol{x}$ variable space (see lines 23-24), and these are implemented as variable bounds in $\mathcal{SP}_{LP}$. When solving a given node, $\mathcal{SP}_{LP}$ must be updated to reflect the branching history. The interesting parts of the proposed Benders decomposition approach are the combined formulation $\mathcal{P}3$ and the suggested branching procedure. All other steps of the algorithm are textbook procedures. It is important to observe that, although problem $\mathcal{SP}$ contains integer variables and is hence not a conventional Benders subproblem, these integrality restrictions are relaxed and gradually introduced through upper and lower bound constraints on specific variable values. The subproblems that we solve are hence only ever linear programs. Thus, standard duality theory can be applied when generating the feasibility cuts.

4. The Cutting Stock Problem

In this section we apply the proposed methodology to the CSP. Recall that the problem involves minimizing the number of rolls (stocks) of given length that must be cut in order to meet the demand for a set of items I of shorter lengths. A demand of b_i is specified for each item $i \in I$. The CSP has been extensively studied in the literature. Results using exact approaches include the BCP algorithm in Alves and De Carvalho (2008), which solves instances with multiple length stocks, and boosts the performance of Ben Amor et al. (2006) through the addition of dual-optimal inequalities. We refer the reader to Ben Amor and Valério de Carvalho (2005) for an overview of solution methods for the CSP. In what follows we show how the proposed methodology can be

Algorithm 1 Benders Decomposition Solution Approach

```
Construct MP_LP and SP_LP
ub ← ∞
while unexplored BAB nodes exist do
    retrieve node to process
    solved ← false
    update x variable bounds in SP_LP to reflect branching history
    while solved is false do
        (z, y) ← Solve MP_LP
        if z ≥ ub then
            fathom node
        end if
        x ← Solve SP_LP with y fixed
        if SP_LP is infeasible then
            generate infeasibility cut (10) and add to MP_LP
        else
            solved ← true
            if x is integer feasible then
                if z < ub then
                    ub ← z
                    save solution
                end if
            else
                Identify a fractional variable x_i with fractional value v
                Create two new nodes by enforcing x_i ≤ ⌊v⌋ ∨ x_i ≥ ⌈v⌉
            end if
        end if
    end while
end while
```

used to exploit the structure of two different formulations of the problem, one of which is the DW reformulation of the other. This first is the arc-flow formulation of Valério de Carvalho (1999), while the second is the pattern formulation of Gilmore and Gomory (1963). We present each in turn, as well as the combined formulation to which Benders decomposition is applied.

Given a stock length of width W, and unique item widths w_i for each item $i \in I$, the arc-flow formulation of Valério de Carvalho (1999) utilizes a graph $G = (N, A)$, where $N = 0, 1, 2, \ldots, W$. Each node $i \in N$ represents a discrete position along the stock. The arc set contains two types of arcs. The first is used to indicate "cutting" items from the stock. Such an arc exists between any two nodes $i \in N$ and $j \in N$ if the length $j - i$ corresponds to an item width. The arc $(i, j) \in A$ therefore indicates cutting the item with length $j - i$, where the cut starts at position i. The second type of arc $(i, i + 1)$, where $i = 1, 2, \ldots, W - 1$, is used to connect the network and indicates loss or waste. Paths starting at node 0 and terminating at node W can hence be used to describe the possible ways in which a stock can be cut. The authors define integer variables $x_{ij} \in \mathbb{Z}_{\geq 0}$ to count the number of items of length $j - i$, which are placed at position i, together with an integer variable $z \in \mathbb{Z}_{\geq 0}$ to count the number of stocks used. The full ILP formulation is given below.

$$(\mathcal{BP}1) \quad \texttt{minimize } z, \tag{11}$$

$$s.t. \quad \sum_{(j,i)\in A} \boldsymbol{x}_{j,i} - \sum_{(i,j)\in A} \boldsymbol{x}_{i,j} = \begin{cases} z & j = 0 \\ 0 & j = 1, \ldots, W-1 \\ -z & j = W \end{cases} \qquad \forall j \in N, \tag{12}$$

$$\sum_{(h,h+w_i)\in A} \boldsymbol{x}_{h,h+w_i} \geq b_i \quad \forall i \in I, \tag{13}$$

$$\boldsymbol{x}_{ij} \in \mathbb{Z}_{\geq 0} \quad \forall (i,j) \in A, \tag{14}$$

$$z \in \mathbb{Z}_{\geq 0}. \tag{15}$$

The objective function (11) minimizes the number of used stocks. Constraints (12) count the number of used stocks and preserve flow conservation. Constraints (13) ensure that all demand is met, and bounds (14)–(15) force variables to assume integer values. A BAP approach is suggested in Valério de Carvalho (1999) to solve this model.

Rather than explicitly determining the number of items of length $j-i$ using integer x_{ij} variables, the formulation provided in Gilmore and Gomory (1963) determines the minimum number of patterns to cut to produce the required item-wise demand. A pattern describes one way in which the stock can be cut into a set of items. Assuming the full set of possible patterns is denoted by P, integer decision variables $\boldsymbol{\lambda}_p \in \mathbb{Z}_{\geq 0}$, where $p \in P$, are introduced and count the number of times pattern p is used. In addition, the parameter $a_{ip} \in \mathbb{Z}_{\geq 0}$ is also introduced and indicates the number of times item $i \in I$ appears in pattern $p \in P$. This gives rise to the following ILP formulation

$$(\mathcal{BP}2) \quad \texttt{minimize} \sum_{p\in P} \boldsymbol{\lambda}_p, \tag{16}$$

$$s.t. \quad \sum_{p\in P} a_{ip}\boldsymbol{\lambda}_p \geq b_i \quad \forall i \in I, \tag{17}$$

$$\boldsymbol{\lambda}_p \in \mathbb{Z}_{\geq 0} \quad \forall p \in P. \tag{18}$$

The objective function (16) minimizes the number of used stocks. Constraints (17) ensure that the item demand is met, and the bounds given by Constraints (18) force variables to assume integer values. For large problems there are likely to be many different possible ways to cut items from the stock. This results in a large number of variables. Consequently, a BAP procedure is usually used to solve this model. To implement column generation, the integrality restriction on the $\boldsymbol{\lambda}_p$ variables is replaced with $\boldsymbol{\lambda}_p \geq 0$. Model (16) – (17), $\boldsymbol{\lambda}_p \geq 0$ is then solved using only a subset of the pattern variables P'. Its optimal dual solution π can then be used to identify potential entering variables. The reduced cost of a pattern $p \in P$ is:

$$1 - \sum_{i\in I} a_{ip}\pi_i \tag{19}$$

The pricing problem can be identified as a knapsack problem; the profit of packing an item $i \in I$ is π_i. To obtain an integer solution to the problem, branching is performed on the $\boldsymbol{\lambda}_p$ variables; however, this complicates the structure of the pricing problem. Instead of the described knapsack problem, it becomes a more difficult constrained shortest path problem.

DW Decomposition is applied to $\mathcal{BP}1$ in Alves and De Carvalho (2008). The resulting master problem is identical in structure to $\mathcal{BP}2$, while the corresponding pricing problem at the root node of the BAP is the integer knapsack problem. If the variable z is fractional, then this variable is branched on. There are situations, however, when z is integer valued in a fractional solution. In such cases, branching must be performed on arcs associated to items. This branching strategy changes the structure of the pricing problem; when solving the integer knapsack problem with dynamic programming, restricting the use of an arc introduces an extra dimension to the state space which in practice slows the solution algorithm, see, e.g., Kellerer et al. (2004). We demonstrate how the proposed Benders decomposition approach can exploit the computational superiority of $\mathcal{BP}2$ while also preserving the integer knapsack problem structure of the pricing problem. The Benders master problem will hence be similar to $\mathcal{BP}2$, while $\mathcal{BP}1$ effectively constitutes the subproblem, where all necessary branches can be enforced. In order to formulate the combined formulation to

which Benders decomposition will be applied, we first restate $\mathcal{BP}2$ slightly differently. We define $x_i \in \mathbb{Z}_{\geq 0}$ to be the number of times item $i \in I$ is cut. The restated version is as follows:

$$\begin{aligned} \texttt{minimize} \quad & \sum_{p \in P} \boldsymbol{\lambda}_p, \\ s.t. \quad & \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p = \boldsymbol{x}_i \quad \forall i \in I, \\ & \boldsymbol{x}_i \geq b_i \quad \forall i \in I, \\ & \boldsymbol{x}_i \in \mathbb{Z}_{\geq 0} \quad \forall i \in I, \\ & \boldsymbol{\lambda}_p \in \mathbb{Z}_{\geq 0} \quad \forall p \in P. \end{aligned}$$

This formulation is clearly equivalent to $\mathcal{BP}2$. The number of times item $i \in I$ is cut in $\mathcal{BP}1$ is defined as: $\sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i}$, see Constraints (13). Now we are able to link the $\boldsymbol{\lambda}_p$ variables of formulation $\mathcal{BP}2$ with the $\boldsymbol{x}_{ij}$ variables of $\mathcal{BP}1$.

$$\begin{aligned} & \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p = \boldsymbol{x}_i = \sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i} && \forall i \in I \\ \Leftrightarrow \quad & \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p = \sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i} && \forall i \in I \end{aligned} \tag{20}$$

Furthermore, the variable z in $\mathcal{BP}1$ indicates the solution value and is thus equal the objective value of $\mathcal{BP}2$. When combining formulations $\mathcal{BP}1$ and $\mathcal{BP}2$, the right hand side of Constraints (12) is thus replaced by $\sum_{p \in P} \boldsymbol{\lambda}_p$, giving

$$\sum_{(j,i) \in A} \boldsymbol{x}_{ji} - \sum_{(i,j) \in A} \boldsymbol{x}_{i,j} = \begin{cases} \sum_{p \in P} \boldsymbol{\lambda}_p & j = 0 \\ 0 & j = 1, \ldots, W-1 \\ -\sum_{p \in P} \boldsymbol{\lambda}_p & j = W \end{cases} \quad \forall j \in N. \tag{21}$$

A combined formulation of $\mathcal{BP}1$ and $\mathcal{BP}2$, where consistency between the values of the respective sets of decision variables is achieved using the linking constraints (20) and (21), can now be stated as:

$$\begin{aligned} (\mathcal{BP}3) \quad \texttt{minimize} \quad & \sum_{p \in P} \boldsymbol{\lambda}_p, && && (16) \\ s.t. \quad & \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p \geq b_i && \forall i \in I, && (17) \\ & \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p = \sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i} && \forall i \in I, && (20) \\ & \sum_{(j,i) \in A} \boldsymbol{x}_{ji} - \sum_{(i,j) \in A} \boldsymbol{x}_{i,j} = \begin{cases} \sum_{p \in P} \boldsymbol{\lambda}_p & j = 0 \\ 0 & j = 1, \ldots, W-1 \\ -\sum_{p \in P} \boldsymbol{\lambda}_p & j = W \end{cases} && \forall j \in N, && (21) \\ & \boldsymbol{x}_{ij} \in \mathbb{Z}_{\geq 0} && \forall (i,j) \in A, && (14) \\ & \boldsymbol{\lambda}_p \in \mathbb{Z}_{\geq 0} && \forall p \in P. && (18) \end{aligned}$$

Note that Constraints (13) are not part of $\mathcal{BP}3$; they are redundant due to Constraints (17) and (20). Also note that $\mathcal{BP}3$ can be see as an *explicit master* in the sense used by Fukasawa et al. (2006). Applying Benders decomposition to the LP relaxation of $\mathcal{BP}3$ yields the following master $\mathcal{BPMP}_{LP}$:

$$(\mathcal{BPMP}_{LP}) \quad \texttt{minimize} \sum_{p \in P} \boldsymbol{\lambda}_p \tag{16}$$

$$s.t. \quad \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p \geq b_i \qquad \forall i \in I, \tag{17}$$

$$\boldsymbol{\lambda}_p \geq 0 \qquad \forall p \in P \tag{22}$$

Given a solution $\overline{\lambda}$ to the master problem, the Benders subproblem $\mathcal{BPSP}_{LP}$ can then be formulated as follows.

$$(\mathcal{BPSP}_{LP}) \quad \texttt{minimize } 0 \tag{23}$$

$$\sum_{(h,h+w_i) \in A} \boldsymbol{x}_{h,h+w_i} = \sum_{p \in P} a_{ip} \overline{\lambda_p} \qquad \forall i \in I, \tag{24}$$

$$\sum_{(j,i) \in A} \boldsymbol{x}_{ji} - \sum_{(i,j) \in A} \boldsymbol{x}_{i,j} = \begin{cases} \sum_{p \in P} \overline{\lambda_p} & j = 0 \\ 0 & j = 1, \ldots, W-1 \\ -\sum_{p \in P} \overline{\lambda_p} & j = W \end{cases} \qquad \forall j \in N \tag{25}$$

$$\boldsymbol{x}_{ij} \geq f_{ij}^l \qquad \forall (i,j) \in A_l \tag{26}$$

$$\boldsymbol{x}_{ij} \leq f_{ij}^u \qquad \forall (i,j) \in A_u \tag{27}$$

$$\boldsymbol{x}_{ij} \geq 0 \qquad \forall (i,j) \in A \setminus \{A_l \cup A_u\} \tag{28}$$

To ensure an integer solution is ultimately obtained we branch on fractional variables in the subproblem using the procedure outlined in Section 3, enforcing new lower and upper bounds on the $\boldsymbol{x}_{ij}$ variables. We indicate these by constraints (26) and (27). Sets A_l and A_u denote arcs with lower and upper bound restrictions. The values of the specific bounds are given by f_{ij}^l and f_{ij}^u, respectively. Branching on a fractional variable $\boldsymbol{x}_{ij}$ produces two child nodes; if infeasibility is encountered, feasibility cuts for $\mathcal{BPMP}_{LP}$ can be constructed using an extreme dual ray associated with the infeasible instance of $\mathcal{BPSP}_{LP}$. Given an extreme dual ray defined by the vectors of dual values $u_i^{(24)}, u_j^{(25)}, u_{ij}^{(26)}$ and $u_{ij}^{(27)}$ for Constraints (24), (25), (26) and (27) the resulting feasibility cut has the following form:

$$\sum_{i \in I} \left(u_i^{(24)} \sum_{p \in P} a_{ip} \boldsymbol{\lambda}_p \right) + u_0^{(25)} \sum_{p \in P} \boldsymbol{\lambda}_p - u_W^{(25)} \sum_{p \in P} \boldsymbol{\lambda}_p + \sum_{(i,j) \in A_l} f_{ij}^l u_{ij}^{(26)} + \sum_{(i,j) \in A_u} f_{ij}^u u_{ij}^{(27)} \leq 0 \tag{29}$$

We solve $\mathcal{BPMP}_{LP}$ with a BAP approach. Consequently, after having branched, the pricing problem must take any added feasibility cuts into account. Assuming a set of feasibility cuts of the form (29), R, is available, where each feasibility cut $r \in R$ has an associated dual value δ_r, the reduced cost of pattern $p \in P$ can be stated as:

$$1 - \sum_{i \in I} a_{ip} \pi_i - \sum_{r \in R} \delta_r \left(\sum_{i \in I} a_{ip} u_i^{(24)} + u_0^{(25)} - u_W^{(25)} \right) \tag{30}$$

where π is the dual vector associated with Constraints (17). The pricing problem therefore remains an integer knapsack problem. The profit of including an item i in the packing is just

$$\pi_i + \sum_{r \in R} \delta_r u_i^{(24)} \tag{31}$$

If the value of (31) results in a negative profit for an item, then the item will never be packed and can hence be removed from the knapsack problem. Thus, the proposed branching strategy within

our Benders decomposition algorithm can be applied to the BAP procedure for $\mathcal{BP}2$ without changing the structure of the pricing problem.

4.1. Initial Solution

To initialize either BAP procedure an initial solution is needed. To do this we adopt a simple greedy algorithm. The method considers the first item with unsatisfied demand and cuts as many copies of the item as it can from the available stock length. Either the demand for the item is obtained or we exhaust the stock length. If, after this, any remaining length exists, the next item with unsatisfied demand is considered. When no additional items can be cut from the pattern, the method starts over with a new stock length and considers the first item with unsatisfied demand and this is repeated until demand has been met. The running time of this method is $\mathcal{O}(|I|^2)$ since the method, in the worst case, needs a pattern for each item and investigates every item for every pattern.

4.2. Upper bound heuristic

Given a fractional solution in the BAP tree, heuristics can be applied to obtain feasible integer solutions to the problem. We implement a simple procedure which floors all fractional values and stores the result. Any remaining unsatisfied demand is then considered. The items with unmet demand are sorted in decreasing order of item length and are then greedily assigned to patterns in the same way as the initial solution is obtained. This heuristic has a running time of $\mathcal{O}(|I|^2)$; at most one pattern per item is needed, hence at most $\mathcal{O}(|I|)$ variables are in the basis. Sorting the items with unmet demand takes $\mathcal{O}(|I|\log(|I|))$ time, and the greedy assignment takes $\mathcal{O}(|I|^2)$ time as argued above.

4.3. Computational Study

To assess the performance of the proposed methodology we benchmark it against the non Benders decomposed formulation $\mathcal{BP}3$, and implementations of the BAP approach described in Alves and De Carvalho (2008) on the 160 CSP instances from Falkenauer (1996)[1]. The instances are divided into eight classes of 20 instances, having a different number of items and one of two possible stock lengths. An overview of the eight different classes is given in Table 1.

Category	$\|I\|$	W
`cutting stock 1`	120	150
`cutting stock 2`	250	150
`cutting stock 3`	500	150
`cutting stock 4`	1000	150
`cutting stock 5`	60	100
`cutting stock 6`	120	100
`cutting stock 7`	249	100
`cutting stock 8`	501	100

Table 1: Summary of the instances

All methods essentially differ in their respective branching routines. The non Benders decomposed model $\mathcal{BP}3$ is solved using column generation, where the $\boldsymbol{\lambda}$ pattern variables are dynamically generated. Branching is, however, performed on the $\boldsymbol{x}$ arc flow variables. This, like the proposed Benders approach, guarantees no modifications to the subproblem are necessary when enforcing branches. This comparison has been included to primarily show the benefits of decomposition in this case. The $\boldsymbol{x}$ arc flow variables are not explicitly included in the BAP approach of Alves and De Carvalho (2008). Any branching decisions must also therefore be accounted for in the subproblem to prevent the generation of patterns that violate enforced branches.

[1] Available online at `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`, last accessed 21-03-2020

All methods are implemented in COIN's open source BCP framework, where the commercial solver Cplex 12.6 is used to solve the respective master problems. For the Benders approach, we also make use of Gurobi 8.0 to solve the subproblem. It is important to stress that the respective implementations are basic and do not include any sophisticated methods such as cutting planes or advanced primal heuristics to find integer feasible solutions. The purpose of this computational study is to solely demonstrate the applicability of the proposed approach in practice. The algorithms are coded in the C++ programming language and all calculations were performed on a Intel Xeon E5-2680 v2 @ 2.80GHz machine with 128GB RAM running Scientific Linux 7.3. We permit a maximum of four threads and allow a maximum running time of three hours.

	$\mathcal{BP}3$ - No Benders			Benders				
Class	Nodes	t_{tot} (s)	Branch	Nodes	t_{tot} (s)	Cuts	Cuts$_{max}$	Branch
`cutting stock 1`	16.40	1.37	2	13.00	0.48	0.00	0	6
`cutting stock 2`	45.40	4.69	7	14.80	1.24	0.00	0	7
`cutting stock 3`	41.00	5.47	4	17.90	2.18	0.00	0	7
`cutting stock 4`	11.57	11.57	6	30.20	4.15	0.00	0	9
`cutting stock 5`	7.20	2.04	6	7.10	0.57	0.00	0	5
`cutting stock 6`	128.70	63.09	20	333.70	45.33	20.45	186	20
`cutting stock 7`	329.90	497.74	20	329.90	87.32	12.14	79	20
`cutting stock 8`	663.80	1462.91	20	861.30	276.29	4.20	21	20

Table 2: Non-Decomposition vs Decomposition

Table 2 provides a summary of the results obtained for the non Benders decomposed approach and the proposed Benders approach on the eight different problem classes. The columns report averaged statistics for the number of nodes in the BAB tree, the wall clock time (in seconds) needed (t_{tot}), and the number of instances for which branching was necessary (Branch). For the Benders approach we also report the average number of cuts (Cuts) and the maximum number of cuts (Cuts$_{max}$) for a single instance. For both approaches we branch on the first fractional value and include the simple upper bound heuristic from Section 4.2. The master problem branching parameters in the BCP framework are set to their default values. For the Benders approach, no branching is performed in the master problem. Consequently, no information is available for node selection when branching. Assuming the first fractional variable in the subproblem is x_i and that its value is f, we instruct the master problem to evaluate the node corresponding to the branch $x_i \geq \lceil f \rceil$ when $f - \lfloor f \rfloor \geq 0.70$, and to evaluate the node with $x_i \leq \lfloor f \rfloor$ otherwise. The results indicate that applying Benders decomposition to $\mathcal{BP}3$ leads to much shorter solution times on average. This is despite the fact that, for the more difficult instances, the BAB trees are larger. The results also show that branching is necessary on all instances from the classes `cutting stock` 6, 7, and 8, and that feasibility cuts are only needed on the same three problem classes. As an indication, on the these three problem classes, the upper bound heuristic provided solutions with an average optimality gap of 7.63%, 5.54%, and 3.5% when using the non Benders decomposed model, while the same values for the Benders approach were 12.38%, 8.98%, and 5.72%, respectively.

Table 3 provides a comparison of the proposed Benders approach with two implementations of the BAP procedure of Alves and De Carvalho (2008). For each approach we report the average number of BAB nodes evaluated, the average time spent in the pricing problem (t_p), and the average total time. The two BAP methods differ in their respective branching strategies. The first, denoted BAP-FF, branches on the first fractional variable, while the second, termed BAP-SB, uses strong branching on a set of candidate variables. Here, the number of candidates is set to half the number of items, i.e., the first $\frac{|I|}{2}$ fractional variables define the candidate set. To determine the variable on which to branch, each candidate is presolved. This step involves performing a certain number of dual simplex iterations on each of the problems defining the candidate's child nodes to get estimates of the resulting optimal objective values. The candidate that has the highest lowest objective value is then selected (the default strategy for COIN BCP). The results show that the Benders approach performs very well and is significantly faster than both BAP methods. The

Class	Benders Nodes	Benders $t_p(s)$	Benders $t_{tot}(s)$	BAP-FF Nodes	BAP-FF $t_p(s)$	BAP-FF $t_{tot}(s)$	BAP-SB Nodes	BAP-SB $t_p(s)$	BAP-SB $t_{tot}(s)$
`cutting stock 1`	13.00	0.17	0.48	49.20	0.80	1.54	31.20	0.37	2.13
`cutting stock 2`	14.80	0.52	1.24	123.80	2.44	4.82	82.90	1.14	7.26
`cutting stock 3`	17.90	1.06	2.18	359.40	13.47	24.78	130.40	1.32	15.45
`cutting stock 4`	30.20	2.16	4.15	359.60	13.56	25.84	230.10	8.78	68.97
`cutting stock 5`	7.10	0.25	0.57	7028.70	531.11	576.31*	54.00	2.75	4.11
`cutting stock 6`	333.70	2.12	45.33	399.80	50.11	64.99	186.10	21.20	50.29
`cutting stock 7`	329.90	7.21	87.32	1825.40	1210.15	1494.04	464.00	210.09	689.40
`cutting stock 8`	861.30	45.88	276.29	2485.10	3651.04	4253.10*	861.10	860.28	3166.73

Table 3: Benders vs BAP

comparison with BAP-FF is perhaps the fairest comparison; only one variable is considered when branching in the Benders approach. An asterisks denotes that there were unsolved instances within a problem class given the 3-hour time limit. For BAP-FF the BAB tree is typically larger and a greater percentage of the total run time is spent in the subproblem. This percentage noticeably increases with more BAB nodes. BAP-SB is much better than BAP-FF, but with exception of perhaps the `cutting stock` 6 class, is noticeably slower than the Benders approach. Looking at the `cutting stock 6` class, this is the one where infeasibility was encountered the most for the Benders approach. The effect of the strong branching is that there are in general fewer nodes evaluated than the BAP-FF approach. Furthermore, a smaller percentage of the total time is spent in the subproblem. An explanation for this is that more time is needed to evaluate the possible branching candidates when strong branching is used.

As can be seen from Tables 2 and 3, the proposed approach outperforms the other tested methods on all instance classes. Dramatic improvements can be seen on the more difficult problem classes (i.e., `cutting stock` 6, 7, and 8). The comparison also shows that solving $\mathcal{BP}3$ directly, without Benders decomposition, outperforms the column generation of Alves and De Carvalho (2008). Looking at Table 2, we can see that not a single feasibility cut is generated using the Benders approach for the first five problem classes. This indicates that branching in the subproblem never yielded an infeasible solution with respect to the master problem. Branching then only creates two new child nodes for the master problem with further constrained subproblem variable domains. The results collectively indicate that the proposed Benders approach has potential for the CSP.

5. The Split Delivery Vehicle Routing Problem

To demonstrate the applicability of the method to the case where one of the two MILP formulations is a relaxation of the other, we examine its performance when applied to the SDVRP. Recall that this variant of the well known vehicle routing problem involves finding the minimum cost set of routes for a set of vehicles through a set of customers, where the demand of a customer can be satisfied through multiple visits, and each vehicle has a known capacity. The structure of this section is therefore similar to Section 4. We begin by introducing the two separate formulations and then show how they can be coupled and solved using BBC. We also explain in detail various aspects of the solution procedure. In particular, in Section 5.1 we outline a heuristic method for obtaining a feasible solution to the problem, in Section 5.2 we describe separation routines for families of valid inequalities for the resulting master problem, and in Section 5.3 we discuss branching possibilities. Computational results on publicly available instances are presented in Section 5.4. For readability, all notation is presented independently of other sections.

The first formulation we consider is a common three-index model (see e.g., Lee et al. (2006); Jin et al. (2007)). This model naturally arises when an instance of the SDVRP with n customers is formulated as a complete digraph $G_1 = (V, E)$, with vertex set $V = \{0, 1, 2, \ldots, n\}$ and edge set $E = \{(i, j) : i \in V, j \in V, i \neq j\}$. The depot is located at vertex 0, and the set of customers is denoted as $V_0 := V \setminus \{0\}$. Each customer $i \in V_0$ is assumed to have a known, positive demand

d_i, and a non-negative cost c_{ij} is associated with the traversal of any edge $(i,j) \in E$. We further assume that $c_{ij} = c_{ji}$ for any two customers i and j and that the costs satisfy the triangle inequality. Given a set of K vehicles, each with capacity Q, the SDVRP involves determining a minimum travel cost set of routes to service the demand of all customers. Routes must start and end at the depot and use no more than $|K|$ vehicles. A customer's demand may be split across multiple vehicles.

Viewing the problem in this way typically leads to three sets of decision variables. The three index binary variable $\boldsymbol{x}_{ij}^k$ is used to indicate whether or not vehicle $k \in K$ traverses edge $(i,j) \in E$. Continuous variables $0 \leq \boldsymbol{\lambda}_{ik} \leq 1$ are then needed to identify the fraction of demand for customer $i \in V_0$ serviced by vehicle $k \in K$. The third set, $\boldsymbol{t}_{ik} \geq 0, \forall i \in V_0, k \in K$, contains continuous variables that are used to eliminate subtours. This leads to the following formulation:

$$(\mathcal{SDVRP}1) \quad \texttt{minimize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} \boldsymbol{x}_{ij}^k \tag{32}$$

$$s.t. \quad \sum_{j \in V} \boldsymbol{x}_{0j}^k \leq 1 \qquad \forall k \in K \tag{33}$$

$$\sum_{j \in V} \boldsymbol{x}_{ji}^k - \sum_{j \in V} \boldsymbol{x}_{ij}^k = 0 \qquad \forall i \in V_0, \forall k \in K \tag{34}$$

$$\boldsymbol{t}_{ik} - \boldsymbol{t}_{jk} + (|V|+1)\boldsymbol{x}_{ij}^k \leq |V| \qquad \forall i,j \in V_0, \forall k \in K \tag{35}$$

$$\sum_{k \in K} \boldsymbol{\lambda}_{ik} = 1 \qquad \forall i \in V_0 \tag{36}$$

$$\sum_{i \in V_0} d_i \boldsymbol{\lambda}_{ik} \leq Q \qquad \forall k \in K \tag{37}$$

$$\boldsymbol{\lambda}_{ik} - \sum_{j \in V} \boldsymbol{x}_{ji}^k \leq 0 \qquad \forall i \in V_0, \forall k \in K \tag{38}$$

$$0 \leq \boldsymbol{\lambda}_{ik} \leq 1 \qquad \forall i \in V, \forall k \in K \tag{39}$$

$$\boldsymbol{t}_{ik} \geq 0 \qquad \forall i \in V, \forall k \in K \tag{40}$$

$$\boldsymbol{x}_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K \tag{41}$$

The objective function (32) minimizes the total cost of the solution. Constraints (33) guarantee each vehicle leaves the depot at most once, while Constraints (34) ensure vehicle flow is conserved. Subtours are eliminated using Constraints (35). Constraints (36) ensure the demand for each customer is serviced, and Constraints (37) make sure that a vehicle is not allocated more demand than its available capacity. Constraints (38) connect the $\boldsymbol{\lambda}$ and $\boldsymbol{x}$ variables. Variable domains are enforced through Constraints (40)–(41).

Formulation $\mathcal{SDVRP}1$ suffers from symmetry issues and it is therefore difficult to solve for large instances. We add the following constraints, as described in Dror et al. (1994), to partially remedy this; however, even with the addition of these constraints, the symmetry issues remain significant.

$$\sum_{j \in V} x_{i^f j}^0 = 1 \tag{42}$$

$$\sum_{k \in K} x_{i^* j^*}^k = 0 \tag{43}$$

The first constraint, (42), arbitrarily assigns the first vehicle to the farthest customer, $i^f \in V_0$, from the depot. Due to the symmetric cost structure, we can also arbitrarily select a single arc in the network and ensure no vehicle uses it. This is what Constraint (43) enforces. Arc (i^*, j^*) is determined as the one with lowest cost. Ensuring no vehicle uses this arc is legal, since the arc in

the other direction, namely (j^*, i^*) is still present.

The second formulation we consider is a two-index formulation that models a relaxation of the SDVRP. This was first proposed by Belenguer et al. (2000) to provide lower bounds for the SDVRP. A slightly modified version of the formulation, which we state here, was proposed by Archetti et al. (2014) and is solved with a BAC procedure to provide what is to date the best known exact algorithm for the SDVRP. In the two-index relaxation the vehicle index is dropped. The aim of the model is to instead determine aggregated vehicle flows between customers. This relaxation of the SDVRP can be formulated using an undirected complete graph $G_2 = (V, E_2)$, where $E_2 = \{(i,j) \in E, i < j, i \in V, j \in V\}$, since edge direction is no longer relevant. In this case, we also define the set $E_0 = \{(i,j) \in E_2 : i \in V_0, j \in V_0)\}$ to be the set of all edges joining pairs of customers.

Only two sets of decision variables are required to model the relaxation. The first set determines edge flows in the graph. More specifically, the variable $\boldsymbol{y}_{ij}$ is used to count the number of times edge (i,j) is traversed by a vehicle. The second type is used to count and restrict vertex visits. In particular, the variable $\boldsymbol{z}_i$ counts the number of times vertex $i \in V$ is visited. The full formulation is given below. To assist in the explanation of the model, we define $k_S = \lceil \sum_{i \in S} \frac{d_i}{Q} \rceil$ to be the minimum number of vehicles needed to serve all customers in $S \subseteq V_0$. For example, k_{V_0} is the minimum number of vehicles needed to satisfy all of the demand. We also let $\delta(S) = \{(i,j) : (i \in S, j \notin S)$ or $(i \notin S, j \in S)\}$ to be the set of edges linking a vertex in S with a vertex in $V \setminus S$. For readability, we write $\delta(i)$ when $S = \{i\}$. Finally, we let $y(S) = \sum_{(i,j) \in S} y_{ij}$.

$$(\mathcal{SDVRP}2) \quad \texttt{minimize} \sum_{i \in V} \sum_{j \in V} c_{ij} \boldsymbol{y}_{ij} \tag{44}$$

$$s.t. \sum_{(i,j) \in \delta(i)} \boldsymbol{y}_{ij} = 2 \cdot \boldsymbol{z}_i \qquad \forall i \in V \tag{45}$$

$$\sum_{(i,j) \in \delta(S)} \boldsymbol{y}_{ij} \geq 2 \cdot k_S \qquad \forall S \subset V_0 \tag{46}$$

$$\sum_{i \in V_0} \boldsymbol{z}_i \leq n + |K| - 1 \tag{47}$$

$$0 \leq \boldsymbol{y}_{ij} \leq 2 \cdot |K| \text{ and integer} \qquad \forall (i,j) \in E \setminus E_0 \tag{48}$$

$$\boldsymbol{y}_{ij} \in \{0,1\} \qquad \forall (i,j) \in E_0 \tag{49}$$

$$k_{V_0} \leq \boldsymbol{z}_0 \leq |K| \text{ and integer} \tag{50}$$

$$0 \leq \boldsymbol{z}_i \leq |K| \qquad \forall i \in V_0 \tag{51}$$

The objective function (44) minimizes the total cost of the solution. Constraints (45) ensure vehicle flow is conserved; the number of times a vertex is visited must be exactly half of the value of the flow on the edges incident to the vertex. Constraints (46) enforce the vehicle capacity requirements and prevent the generation of subtours. Assuming the costs satisfy the triangle inequality, Constraint (47) enforces an upper bound on the total number of visits to vertices. Finally, variable domains are enforced by constraints (48)–(50). Again, if we assume that the costs satisfy the triangle inequality, then any edge that connects two customers will not be traversed more than once. Constraints (49) enforces this. This formulation is a relaxation of the SDVRP as it may not be possible to disaggregate the aggregated vehicle flows into individual vehicle routes. An example of this is provided in Archetti et al. (2014).

Archetti et al. (2014) propose a BAC approach applied to formulation $\mathcal{SDVRP}2$ to find the optimal solution to the $\mathcal{SDVRP}1$. Constraints (46) are removed and dynamically separated, along with *connectivity cuts*. The connectivity cuts are defined as:

$$\sum_{(i,j) \in \delta(S)} y_{ij} \geq 2 z_i \quad \forall S \subseteq V_0, |S| \geq 2, i \in S$$

The connectivity cuts can only be used when the costs c_{ij} satisfy the triangle inequality. Given an integer solution, the expression $\frac{1}{2}\sum_{(i,j)\in\delta(S)} y_{ij}$ is an upper bound on the number of vehicles serving the set S (upper bound, since vehicles may enter and leave the set several times). If $\tilde{k}_S$ vehicles are serving a set S in a certain solution then we can enforce that each vertex in S is visited at most $\tilde{k}_S$ times: If a vertex $i \in S$ was visited more than $\tilde{k}_S$ times it means that it is visited more than once by the same vehicle. In this case we can simplify the route by eliminating extra visits on the route and short cutting the route every time we remove a visit to customer i (i.e. a segment $\ldots \to k \to i \to j \to \ldots$ is reduced to $\ldots \to k \to j \to \ldots$). Since the c_{ij} matrix is assumed to satisfy the triangle inequality we are not making the solution longer by applying these short cutting operations. Technically speaking the constraints are not valid inequalities since they can cut away feasible integer solutions but since they always leave at least one optimal integer solution they are safe to use. In practice we observed that violated connectivity cuts can be detected even after separating the capacity constraints (46) so they are useful in reducing the solution space.

Whether or not a feasible solution to $\mathcal{SDVRP}2$ is also feasible for the $\mathcal{SDVRP}1$ is assessed using a separate route based procedure, which is formulated using a MILP. If the solution is feasible, it is taken into consideration, otherwise it is excluded through the addition of a constraint which cuts away the solution. To our knowledge, this is the best known exact method for the SDVRP, and the approach with which we compare the performance of our algorithm in Section 5.4.

The two formulations outlined above both have advantages and disadvantages. Formulation $\mathcal{SDVRP}1$ guarantees feasible solutions, but is computationally difficult due to its large number of variables and symmetry, while formulation $\mathcal{SDVRP}2$ is computationally appealing, but may result in infeasible solutions. The methodology we propose tries to exploit this. We propose applying BBC to a formulation comprised of formulations $\mathcal{SDVRP}1$ and $\mathcal{SDVRP}2$, coupled together with the following linking constraints

$$\boldsymbol{y}_{ij} = \sum_{k\in K} (\boldsymbol{x}^k_{ij} + \boldsymbol{x}^k_{ji}) \quad \forall (i,j) \in E_2 \tag{52}$$

This gives the following formulation:

$$\begin{aligned} (\mathcal{SDVRP}3) \quad & \texttt{minimize} && \sum_{i\in V}\sum_{j\in V} c_{ij}\boldsymbol{y}_{ij} && (44)\\ & s.t. && (33)-(43) \\ & && (45)-(47) \\ & && (48)-(52) \end{aligned}$$

Formulation $\mathcal{SDVRP}2$ is more computationally appealing than formulation $\mathcal{SDVRP}1$. Hence, it becomes the master problem when applying the proposed decomposition approach. To be able to apply BBC, we also relax the integrality restrictions on the decision variables. The master problem is therefore:

$$\begin{aligned} (\mathcal{SDVRPMP}_{LP}) \quad & \texttt{minimize} && \sum_{i\in V}\sum_{j\in V} c_{ij}\boldsymbol{y}_{ij} && && (44)\\ & s.t. && (45)-(46) \\ & && 0 \le \boldsymbol{y}_{ij} \le 2\cdot|K| && \forall (i,j) \in E\setminus E_0 && (53)\\ & && 0 \le \boldsymbol{y}_{ij} \le 1 && \forall (i,j) \in E_0 && (54)\\ & && k_{V_0} \le \boldsymbol{z}_0 \le |K| && && (55)\\ & && 0 \le \boldsymbol{z}_i \le |K| && \forall i \in V_0 && (56) \end{aligned}$$

Formulation $\mathcal{SDVRP}1$ is essentially used to assess the feasibility of solutions found by formulation $\mathcal{SDVRPMP}_{LP}$, and thus its LP-relaxation forms the subproblem in the proposed BBC approach. It can be stated as follows:

$$(\mathcal{SDVRPSP}_{LP}) \quad \texttt{minimize} \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} 0\cdot \boldsymbol{x}_{ij}^k \tag{57}$$

$$s.t. \qquad (33)-(41)$$

$$(42)-(43)$$

$$\sum_{k\in K}(\boldsymbol{x}_{ij}^k + \boldsymbol{x}_{ji}^k) = \hat{y}_{ij} \qquad \forall (i,j)\in E_2 \tag{58}$$

$$0 \le \boldsymbol{x}_{ij}^k \le 1 \qquad \forall (i,j)\in E, \forall k \in K \tag{59}$$

Again, we are interested in feasibility of the subproblem only, and this explains the nature of objective function (57). The subproblem is also defined for a fixed solution $\hat{y}_{ij}$ to formulation $\mathcal{SDVRPMP}_{LP}$. Given an infeasible solution to formulation $\mathcal{SDVRPSP}_{LP}$, if we let $u_k^{(33)}$, $u_{ijk}^{(35)}$, $\mu_i^{(36)}$, $\mu_k^{(37)}$, $\gamma ij^{(58)}$, and $\eta^{(42)}$ denote the components of the corresponding extreme dual ray associated with constraints (33), (35), (36), (37), (58), and (42), then, assuming no branching constraints on the subproblem variables, the following feasibility cut can be added to $\mathcal{SDVRPMP}_{LP}$:

$$\sum_{k\in K} u_k^{(33)} + \sum_{(i,j)\in E}\sum_{k\in K} |V|\cdot u_{ijk}^{(35)} + \sum_{i\in V_0}\mu_i^{(36)} + \sum_{k\in K} Q\cdot \mu_k^{(37)} + \sum_{(i,j)\in E_2}\gamma_{ij}^{(58)}\cdot \boldsymbol{y}_{ij} + \eta^{(42)} \le 0 \tag{60}$$

When applying the proposed BBC approach, the integrality restrictions present in formulation $\mathcal{SDVRP3}$ are enforced through the use of a BAB tree. Typically restrictions appear as bounds on subproblem variables (or groups of subproblem variables). If such constraints appear, then their respective dual objective contribution must be included when constructing the feasibility cut (see Constraint (29) for an example of this). In this work, both formulations $\mathcal{SDVRPMP}_{LP}$ and $\mathcal{SDVRPSP}_{LP}$ will be solved with a commercial LP solver. How successful the proposed BBC approach is at finding the optimal solution depends on three main factors: obtaining a good upper bound to limit the size of the tree, the separation of valid inequalities, and the branching strategies used to eliminate fractionality. We discuss each of these aspects in the following three subsections.

5.1. Upper bound heuristic

In order to find good upper bounds for the SDVRP a simple Large Neighbourhood Search (LNS) heuristic has been implemented. LNS search was first proposed by Shaw (1998) and has since been widely used. The implementation in this paper is inspired by the LNS heuristic described by Ropke and Pisinger (2006) and Pisinger and Ropke (2007).

Pseudocode for the algorithm is shown in Algorithm 2. The algorithm maintains three solutions: the current solution x, a temporary solution x', and the best known solution x^*. The variable i is an iteration counter. The main loop in lines 4–13 repeatedly destroys and repairs the current solution using functions destroy(...) and repair(...) (line 5) that are explained below. In line 6 the accept(...) function decides if the temporary solution should be accepted and made the new current solution, and in lines 9–11 it is checked if a new best solution has been found ($f(x)$ evaluates the objective of x). The accept$(x', x^*, i, i_{max}, T_0)$ function accepts the solution x' if

$$\frac{f(x') - f(x^*)}{f(x^*)} < T_0\left(1 - \frac{i}{i_{max}}\right)$$

that is, solutions that are better than the best known solution are always accepted. Solutions that are worse than the best known solution are accepted if the deviation from the best known objective is within a certain threshold. This threshold starts at T_0 and is linearly decreased to zero. This acceptance criterion is known as record-to-record travel and was introduced by Dueck (1993). Experiments in Santini et al. (2018) indicate that this acceptance criterion and way of controlling the acceptance threshold is simple to tune and works well with LNS heuristics.

Algorithm 2 Large neighborhood search

input: Initial solution: x, Start threshold: T_0, Iteration limit: i_{max}
$x^* = x$
$i = 0$
while $i \leq i_{max}$ **do**
    $x' = \text{repair}(\text{destroy}(x))$
    if $\text{accept}(x', x^*, i, i_{max}, T_0)$ **then**
        $x = x'$;
    end if
    if $f(x') < f(x^*)$ **then**
        $x^* = x'$
    end if
    $i = i + 1$
end while
return x^*

5.1.1. Destroy method

The destroy method selects k customers at random and removes them from the solution. If a chosen customer is split between several routes all "copies" of the customer are removed. The number of customers to remove is found by drawing a random integer in the interval $[k_{min}; k_{max}]$ where k_{min} and k_{max} are determined by

$$k_{min} = \max(10, 0.1n) \text{ and } k_{max} = \min(50, 0.4n),$$

The approach for finding number of customers to remove and parameter values are based on Pisinger and Ropke (2007).

5.1.2. Repair method

The repair method consists of two phases. In the first phase customers are inserted without splitting demand. Since the number of vehicles is limited, the first phase may end with some customers that do not fit into any route. In the second phase these customers are inserted by splitting their demand and serving them by two or more vehicles.

The first phase (non-splitting phase) is based on a regret heuristic (see Potvin and Rousseau (1993), Ropke and Pisinger (2006)). The heuristic is given a set S of non-planned customers. For each customer $i \in S$, let δ_i^k express the cost of inserting customer i in its k'th cheapest route. The heuristic iteratively inserts the customer i from S that maximizes the *regret* $\delta_i^2 - \delta_i^1$. This customer is inserted at its cheapest possible position in the solution and removed from S. If a customer i only fits into one route then δ_i^2 is set to ∞.

In the second phase, the heuristic inserts customers that are left over after the first phase. The heuristic goes through the routes one by one, in order of increasing route id. If there is any capacity left on the route under consideration, the customer, that can be inserted at the cheapest possible cost, is inserted. If there is not enough capacity on the route to insert the customer completely, as much demand as possible is served and the rest is left for another route. If there is capacity left on the route after inserting a customer (this can happen once one starts to insert "remainders") more customers are inserted. The heuristic goes on to process the next route once all capacity of a route is used.

5.2. Separation procedures for the SDVRP

At any node of the BAB tree we consider separating four types of valid inequalities: capacity cuts, connectivity cuts, *k-route inequalities*, and Benders feasibility cuts. The k-route cuts inequalities are only considered when the solution to a node is integer feasible. For Benders feasibility cuts, with the exception of the root node, we also only call separation routine for integer solutions. A short explanation on how each of the first three types of cuts are separated is given

below. Benders feasibility cuts can be separated by solving $\mathcal{SDVRPSP}_{LP}$. We therefore do not include any further discussion on this.

5.2.1. Capacity cuts

Capacity cuts are separated by a randomized greedy heuristic, very similar to the *greedy rounded capacity heuristic* described in Naddef and Rinaldi (2002). Based on a seed node $i \in V_0$ the heuristic constructs a set $S = \{i\}$ that is iteratively enlarged. In the basic version, at each step, a node j is added to S such that $x(\delta(S \cup \{j\}))$ is minimized. After updating S we check if

$$x(\delta(S)) \geq 2k_S$$

if this is not the case, a violated inequality has been detected. After detecting a violated inequality the enlarging of S continues. If several sets S defining violated inequalities are found during the expansion of a set, the one maximizing $2k_S - x(\delta(S))$ is kept. In order to randomize the heuristic a small random noise term is added to the evaluation of $x(\delta(S \cup \{j\}))$ while growing the set S. The heuristic is run with every node $i \in V_0$ as the seed node and the heuristic is repeated several times for each seed node. The motivation for repeating the heuristic from the same seed node is that the noise component may lead to a different selection of sets in each repetition.

5.2.2. Connectivity cuts

Connectivity cuts are separated by solving a max-flow problem from each node $i \in V_0$ to the depot (see Archetti et al. (2014)). Only nodes $i \in V_0$ with $z_i > 1$ are considered. For nodes with $z_i = 1$ the constraints would only be violated if the classic TSP subtour elimination constraint is violated and the capacity cuts contain that class of inequalities as a special case.

5.2.3. k-route inequalities

The k-route inequality was first proposed by Pereira do Valle (2012) and specifically focuses on assessing whether or not an integer feasible solution to formulation $\mathcal{SDVRP}2$ can be successfully disaggregated into individual vehicle routes, as required by formulation $\mathcal{SDVRP}3$. In particular, the analysis is centered around the "split customer nodes" (i.e., customers visited by more than one vehicle) in a solution to $\mathcal{SDVRP}2$ and the customer disjoint path segments connected to them. A path segment for a particular split node is a sequence of nodes that is used in the solution and which connects the split node to another split node or the depot. If individual vehicle routes can be obtained from a solution to $\mathcal{SDVRP}2$, then for any split node it must be possible to partition the path segments connected to the node into pairs that do not violate the allowed vehicle capacity. If this partitioning is not possible for a given split node $i \in V_0$, Pereira do Valle (2012) show that the following cut can be used to remove the solution from consideration:

$$x(\delta(i)) \geq 2 \cdot \beta + 2 \cdot \sum_{j=1}^{p} x(S_{j|i}) - x(\delta(S_j)) - x(S_j) + |S_j|,$$

where $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ is the set of p path segments associated with node i. Path segments are subsets of non-split nodes, i.e, $S_j \subset V_0 \; \forall j = 1, 2, \ldots, p$, where any two path segments $S_l, S_{l'} \in \mathcal{S}$ satisfy $S_l \cap S_{l'} = \emptyset$. Each $S_j \in \mathcal{S}$ has an associated demand value $\hat{d}_{S_j} = \sum_{l \in S_j} d_l$. β states the minimum number of vehicles required to service the total demand of all path segments in $\mathcal{S}$. The set $S_{j|i} = \{(k, l) \in E : (l \in S_j, k = i) \text{ or } (k \in S_j, l = i)\}$. As Pereira do Valle (2012) show, β can be found from the solution to a small bin packing problem. There is one item in the bin packing problem for each path segment, and the weight of an item is the demand associated with the corresponding path segment. Each bin has a capacity of Q, i.e, the vehicle capacity, and each bin is limited to at most two items. Two items j and j' are compatible if the combined weight of the items is less-than-or-equal-to Q, i.e., if $\hat{d}_{S_j} + \hat{d}_{S_{j'}} \leq Q$. To generate all path segments connected to a particular split node i a depth first search to connected nodes can be used. We refer the reader to Pereira do Valle (2012) for more details.

5.3. Branching

For this particular implementation of the BBC approach we prefer a hybrid branching strategy in which we branch on master problem variables as well as subproblem variables. Unlike the CSP, in this case no complications arise when branching on the master problem variables, and we prefer to remove this fractionality first. Given a fractional solution (y', z') to $\mathcal{SDVRPMP}_{LP}$, we rank the variables in increasing order of their deviation in value from 0.5. More specifically, the fractionality of variable $\boldsymbol{y}_{ij}$ is computed as $|y'_{ij} - 0.5|$. Variables with a low deviation are considered to be more fractional than variables with a higher deviation. From this ranked list of variables we identify a candidate set of fractional variables to be used by a strong branching routine. The set size is determined as a fixed percentage of the number of fractional variables. To determine the variable on which to branch, each candidate is presolved. The one that results in the highest lowest objective value is selected, and a variable branch on this variable is then enforced in $\mathcal{SDVRPMP}_{LP}$. As an example, if variable $\boldsymbol{y}_{ij}$ is selected to branch on, the respective branches are $\boldsymbol{y}_{ij} \leq \lfloor y'_{ij} \rfloor$ and $\boldsymbol{y}_{ij} \geq \lceil y'_{ij} \rceil$ (analogous for the $\boldsymbol{z}$ variables).

If the solution to a particular node is integer feasible for the master problem, three situations can arise: the subproblem is integer feasible, the subproblem is LP-infeasible, the subproblem is only LP-feasible (but no feasible solution to the integer program exists). In the first case, we have a feasible solution to $\mathcal{SDVRP}1$ and we can prune the node. In the second case, we can generate a feasibility cut of the form (60), which can be added to $\mathcal{SDVRPMP}_{LP}$ to remove the current master solution from further consideration. For the last case, we can branch on fractional variables in the solution to the associated $\mathcal{SDVRPSP}_{LP}$. For this we propose a two level strategy. We first look at arcs that are not connected to the depot and which have a fractional coverage less than the value of one. In particular we look at arcs (i, j) satisfying the following:

$$0 < \sum_{k \in K} x_{ij}^k < 1.0$$

and create the two branches $\sum_{k \in K} x_{ij}^k = 1$ and $\sum_{k \in K} x_{ij}^k = 0$. Like the branching strategy in the master problem, we select the most fractional arc (i.e, the one with a fractional coverage closest to 0.5) to branch on. If such an arc does not exist, we resort to branching on individual vehicles and create the two branches $x_{ij}^k = 1$ and $x_{ij}^k = 0$. Again, we select the most fractional variable to branch on. This implementation is carried out as per line 24 of Algorithm 1. With the exception of the optimal solution to the LP relaxation of the master problem, Benders feasibility cuts are only separated on integer master solutions. Given an integer solution, we first attempt to solve formulation $\mathcal{SDVRPSP}_{LP}$ with integer restrictions on the $\boldsymbol{x}$ variables for a limited time (i.e, 10 seconds). If a feasible solution cannot be found in this time, we start to assess LP feasibility.

The proposed solution approach resembles that of Archetti et al. (2014); we also solve a relaxation that is coupled with a separation routine to assess feasibility of the original problem. However, there are noticeable differences: 1) A feasibility cut has the potential to cut away multiple master solutions and 2) the proposed approach permits branching in the subproblem space and any infeasibility encountered can be conveyed to the master problem via feasibility cuts.

5.4. Computational Study

We use a benchmark set of 43 SDVRP instances to test the proposed approach. Again, the commercial solvers CPLEX 12.6 and Gurobi 8.0 are used to solve the master and subproblem linear programs, respectively. We intend to highlight the general applicability of the approach and as such use COIN's BCP framework for the implementation. The proposed Benders Decomposition algorithm is coded in the C++ programming language, and all calculations were performed on a Intel Xeon E5-2680 v2 @ 2.80GHz machine with 128GB RAM running Scientific Linux 7.3, using a maximum of four cores.

The 43 instances are well known and come from three different sets. The first set (Set 1) was proposed by Belenguer et al. (2000) and is comprised of instances from the TSPLIB. The second set (Set 2) was also proposed by Belenguer et al. (2000). These particular instances are generated based on the instances $eil51$, $eil76$, and $eil101$ from the first set. The vehicle capacity is

assumed to be 160, customer demands are generated based on a set of scenarios and are randomly generated from uniform distributions. The third set (Set 3) was put forward by Archetti et al. (2006). Customer demands are generated in the same way as Set 2, but the vehicle capacity lies between 140 and 200. Customer coordinates are taken from the Capacitated Vehicle Routing Problem (Christofides et al. (1979)). In this paper we restrict our attention to instances which have integer edge costs

From an implementation perspective, when branching in the master problem, the strong branching candidate list contains 35% of all fractional variables. Master problem cuts are separated in the order: capacity, connectivity, k-route, and feasibility. If a cut is found from one particular separation routine, then the master problem is immediately updated, and the remaining separation routines are not called. The deterministic upper bound heuristic is run just once prior to running the Benders framework and is given 150,000 iterations. In all experiments we observe a time limit of three hours.

We compare our results directly with those reported in Archetti et al. (2014) and, with the best known lower and upper bound for each instance. Furthermore, like we did for the CSP, we also compare the proposed approach with a non Benders decomposed version of $\mathcal{SDVRP}3$ to highlight the need for decomposition. The best known lower and upper bounds have been collected from the following sources: Belenguer et al. (2000) (a), Moreno et al. (2010) (b), Archetti et al. (2014) (c), Silva et al. (2015) (d), and Ozbaygin et al. (2018) (e). We credit the earliest research giving the bound. To identify the source of a bound in any results table, we attach the letter given in the parentheses as superscript. Silva et al. (2015) propose a multi-start iterated local search procedure to solve the SDVRP. The method includes a novel perturbation method to improve diversification. This metaheuristic provides many of the best known solutions to benchmark SDVRP instances. Its performance on Set 3 is, however, unknown. Ozbaygin et al. (2018) develop new exact methods for the SDVRP. The authors propose solving a relaxation of an arc flow formulation in which the decision variables are aggregated over all vehicles. This results in a formulation that is similar to the two index formulation of Archetti et al. (2014) that we compare to. Two methods are proposed to eliminate solutions of the relaxation that are not feasible for the SDVRP. The first is a *patching* method which locally extends the relaxation with vehicle indexed variables, while the second is a *node-splitting* algorithm. Node splitting is an alternative way of making vehicle distinction in the relaxation. The two methods are not competitive with that of Archetti et al. (2014), but improve the upper bound on four of the instances. On sets 1 and 2, the column and cut generation algorithm of Moreno et al. (2010) generally provides the tightest lower bounds.

Tables 4, 6 and 8 provide an overview of the objective values obtained using the proposed algorithm on each of the test sets. For each instance we report the best known lower and upper bounds, the best lower and upper bounds obtained stated in Archetti et al. (2014) for the author's BAC method (configuration c), and the root lower bound, the best lower bound, the best upper bound, and the optimality gap provided by the proposed Benders algorithm. A dash for a particular method indicates that no solution was obtained for the respective instance. In Archetti et al. (2014) a two hour time limit was observed. For comparison purposes, we report percentage improvements for best known results and those provided by the BAC approach. The percentage improvement in the upper bound is computed as $(UB_c - UB_b)/UB_c \times 100\%$, where UB_b is the bound provided by the Benders approach and UB_c is the bound to which we compare. As such, improvements in the upper bound value have positive values. The lower bound improvement is computed similarly, $(LB_b - LB_c)/LB_c \times 100\%$. In addition to this, we provide Tables 5, 7,and 9. Each table reports summary statistics for the respective test sets. In particular we state the number of nodes in the instance, the total wall clock time needed; the time used by the heuristic (Heur.); the time used by each of the Benders (Ben.) Capacity (Cap.), Connectivity (Con.) and k-route (k) separation routines; the number of cuts generated in total by each of the separation routines; the number of times the Benders subproblem was solved (Sub); and the total number of nodes evaluated in the BAB tree.

The results indicate that the proposed methodology is promising. For instance sets 1, 2, and 3 we report average optimality gaps of 2.36%, 4.24%, and 5.83% within the three hour time limit. The method is currently not competitive, from a solution time perspective on the larger instances

with the approach of Archetti et al. (2014), which solves an additional three instances (S101D1, p01_1030, and p02_110) to optimality in a much shorter time frame. However, it is important to note that our approach utilizes an open source decomposition framework, which is unlikely to be as effective as a commercial solver. Furthermore, all separated cuts are only locally applied. This explains the large number of capacity cuts generated. With this in mind, it is impressive that a fairly generic methodology performs well in comparison to a state-of-the-art, exact approach.

Looking at the results more closely, we can see that for several of the instances (e.g., eilB76, eilC76, and eilD76) no integer solutions, other than the one found by the heuristic, are encountered within the time limit. One reason for this is the high quality of the solution found by the heuristic. For such instances, the Benders subproblem is only solved once. For any instance, we always solve at least one subproblem because we check if we can add infeasibility cuts to the root node based on its optimal LP solution. Recall that the solution found by the heuristic is used as an initial upper bound in the branching process. The better this solution is, the fewer the number of integer solutions that will be found in the B&B tree. If the heuristic provides the optimal solution, the B&B procedure is essentially used to prove optimality. For instances like eilA76 and eilB76, the solution provided by the heuristic is some distance from the best known bound solution ($> 5\%$). In such cases, the weakness of the LP relaxation prevents the branching procedure from finding integer solutions within the time limit.

A second observation is that for several instances (e.g, S51D4, S51D5, S51D6), the Benders separation routine is responsible for a significant amount of the total runtime. For such instances there are usually a significant number of subproblem solved (> 1100) but relatively few feasibility cuts (around 200). In such cases, the solution to $\mathcal{SDVRP}2$ is feasible for the $\mathcal{SDVRPSP}_{LP}$, but not $\mathcal{SDVRP}3$. In other words, these instances are characterized by significant amounts of subproblem branching. This can naturally result in large BAB trees. Without a certificate of infeasibility for $\mathcal{SDVRP}3$ it can be difficult to dismiss a feasible solution to $\mathcal{SDVRP}2$. From a cutting perspective, the k-route cuts have, with the exception of eil30, relatively little impact.

Finally, Table 10 gives a summary of the results obtained when trying to directly solve the combined formulation $\mathcal{SDVRP}3$. For this, we use CPLEX 12.6 and dynamically separate capacity and connectivity cuts via call back functions. Due to the poor performance of the combined approach we limit the focus to nine instances; these were the only instances for which an upper bound was provided by CPLEX. In some cases, the root node could not even be solved within three hours. Table 10 clearly highlights the need for decomposition. The results are much worse than the decomposed counterpart. The lower number of capacity and connectivity cuts can be attributed to the fact that these are applied globally with the CPLEX implementation.

The computational results clearly demonstrate the applicability and potential of the proposed approach on the SDVRP. Looking forward, being able to more quickly detect integer infeasibility in formulation $\mathcal{SDVRP}2$ for a feasible solution to $\mathcal{SDVRP}3$ is critical to the success of the method. This, along with the inclusion of more heuristics to generate feasible solutions to $\mathcal{SDVRP}2$ will further improve the algorithm's performance. From an implementation perspective it is not obvious whether the proposed approach can be implemented directly in a commercial solver as a BAC algorithm. A key component of the approach is the ability to branch on subproblem variables. This results in "empty" master problem branches where the only difference between one node of the BAB tree and its children are the bounds on subproblem variables.

6. Conclusion

When solving ILPs usually several possible formulations exist, and each has its own advantages and disadvantages. Either a DW reformulation or a relaxation of the original problem might be useful in creating a tractable formulation, but this can often lead to situations in which it can be difficult to enforce branching decisions.

In this paper we propose a generic, Benders decomposition based approach for simultaneously exploiting the structure in two formulations, one of which is either a DW reformulation or a relaxation of the other. The method involves combining the two formulations into one formulation

	Best		BAC		Benders				Δ LB (%)		Δ UB (%)	
Instance	LB	UB	LB	UB	Root LB	Final LB	UB	Opt (%)	BAC	Best	BAC	Best
eil22	375.00[b]	375[a]	375.00	375	375.00	375.00	375	0.00	0.00	0.00	0.00	0.00
eil23	569.00[a]	569[a]	569.00	569	569.00	569.00	569	0.00	0.00	0.00	0.00	0.00
eil30	510.00[c]	510[a]	510.00	510	508.83	510.00	510	0.00	0.00	0.00	0.00	0.00
eil33	835.00[c]	835[a]	835.00	835	833.50	835.00	835	0.00	0.00	0.00	0.00	0.00
eil51	521.00[c]	521[a]	521.00	521	513.40	521.00	521	0.00	0.00	0.00	0.00	0.00
eilA76	807.60[b]	818[d]	782.00	876	781.02	792.68	832	4.73	1.37	-1.85	5.02	-1.17
eilB76	981.40[b]	1002[d]	932.30	1083	939.14	950.86	1014	6.23	1.99	-3.11	6.37	-1.20
eilC76	717.80[b]	733[d]	711.54	738	704.10	713.27	736	3.09	0.24	-0.63	0.27	-0.41
eilD76	666.10[b]	681[d]	665.43	698	657.47	664.94	690	3.63	-0.07	-0.17	1.15	-1.32
eilA101	799.80[b]	815[d]	795.58	836	790.05	799.23	818	2.29	0.46	-0.07	2.15	-0.37
eilB101	1040.60[b]	1061[d]	999.84	-	999.51	1006.37	1071	6.03	0.65	-3.29	-	-0.94
Avg.								2.36				

Table 4: Objectives - Set 1

		Time (s)						Cuts					
Instance	$\|V\|$	Total	Heur.	Ben.	Cap.	Con.	k	Ben.	Cap.	Con.	k	Sub	Nodes
eil22	21	1.73	1.54	0.07	0.00	0.00	0.00	0	53	0	0	1	1
eil23	22	1.26	1.22	0.03	0.00	0.00	0.00	0	20	1	0	1	1
eil30	29	6.73	1.62	4.08	0.25	0.01	0.02	30	212	12	33	61	163
eil33	32	2.09	1.94	0.03	0.03	0.00	0.00	0	162	0	0	1	3
eil51	50	7.44	3.29	0.18	1.58	0.01	0.00	0	1785	12	0	1	43
eilA76	75	10801.43	5.93	174.84	2858.13	18.15	0.00	2	1228108	15389	0	18	13883
eilB76	75	10800.94	6.54	3.92	2688.26	17.92	0.00	0	1373985	14253	0	1	11293
eilC76	75	10800.21	5.51	1.33	3565.42	17.66	0.00	0	1415078	20703	0	1	20687
eilD76	75	10800.26	5.36	1.17	4066.95	21.76	0.09	0	1379215	27028	1	1	29194
eilA101	100	10800.38	8.08	1.80	3776.21	9.88	0.00	0	697438	6766	0	1	11020
eilB101	100	10800.46	9.06	2.59	2845.67	13.51	0.00	0	674564	12231	0	1	4011

Table 5: Statistics - Set 1

	Best		BAC		Benders				Δ LB (%)		Δ UB (%)	
Instance	LB	UB	LB	UB	Root LB	Final LB	UB	Opt (%)	BAC	Best	BAC	Best
S51D1	458.00[c]	458[a]	458.00	458	454.33	458.00	458	0.00	0.00	0.00	0.00	0.00
S51D2	703.00[c]	703[c]	703.00	703	682.67	703.00	703	0.00	0.00	0.00	0.00	0.00
S51D3	930.70[b]	943[e]	925.94	944	910.45	924.54	960	3.69	-0.15	-0.66	-1.69	-1.80
S51D4	1547.04[b]	1551[c]	1547.04	1551	1529.58	1535.93	1625	5.48	-0.72	-0.72	-4.77	-4.77
S51D5	1313.40[b]	1328[e]	1300.64	1328	1285.07	1298.60	1341	3.16	-0.16	-1.13	-0.98	-0.98
S51D6	2141.70[b]	2153[f]	2101.78	2301	2131.72	2135.06	2235	4.47	1.58	-0.31	2.87	-3.81
S76D1	592.00[c]	592[c]	592.00	592	585.98	592.00	592	0.00	0.00	0.00	0.00	0.00
S76D2	1061.10[b]	1081[e]	1007.73	-	1022.86	1035.62	1094	5.34	2.77	-2.40	-	-1.20
S76D3	1395.90[b]	1419[e]	1345.47	-	1345.94	1359.78	1447	6.03	1.06	-2.59	-	-1.97
S76D4	2046.10[b]	2071[e]	1980.71	-	2002.33	2011.26	2158	6.80	1.54	-1.70	-	-4.20
S101D1	716.00[c]	716[c]	716.00	716	703.25	710.54	718	1.04	-0.76	-0.76	-0.28	-0.28
S101D2	1337.10[b]	1364[e]	1260.59	-	1271.99	1280.15	1384	7.50	1.55	-4.26	-	-1.47
S101D3	1832.10[b]	1859[e]	1693.47	-	1754.56	1761.39	1922	8.36	4.01	-3.86	-	-3.39
S101D5	2737.10[b]	2772[e]	2522.71	-	2668.44	2673.83	2891	7.51	5.99	-2.31	-	4.29
Avg.								4.24				

Table 6: Objectives - Set 2

and augmenting the resulting formulation with linking constraints to force consistency between the decision variables of the two otherwise separate formulations. The structure of the resulting formulation is such that it naturally lends itself to Benders decomposition. The LP relaxation of the reformulation (or that of a relaxed formulation) can be identified as the Benders master

		Time (s)						Cuts					
Instance	$\|V\|$	Total	Heur.	Ben.	Cap.	Con.	k	Ben.	Cap.	Con.	k	Sub	Nodes
S51D1	50	3.91	2.99	0.44	0.16	0.00	0.00	0	139	0	0	2	17
S51D2	50	973.68	3.85	144.31	218.79	2.85	0.00	11	254489	4961	0	77	5443
S51D3	50	10800.38	4.28	657.32	1996.66	35.46	0.09	12	2753961	26696	1	205	46868
S51D4	50	10800.60	6.40	9639.31	96.67	1.87	0.22	209	266942	537	2	1616	5101
S51D5	50	10800.19	5.89	8506.93	270.10	4.32	0.00	171	642191	1529	0	1814	10514
S51D6	50	10802.43	8.43	9881.74	49.48	1.27	0.00	116	254371	99	0	1123	3363
S76D1	75	70.64	4.76	2.54	30.38	0.07	0.00	0	11150	76	0	2	393
S76D2	75	10801.47	6.97	3.95	2374.64	13.02	0.00	0	1345812	9964	0	1	7663
S76D3	75	10800.19	7.59	6.96	2183.65	17.35	0.00	0	1355562	11509	0	1	8407
S76D4	75	10802.30	10.30	7454.53	420.18	3.58	0.00	30	482648	1580	0	446	3390
S101D1	100	10800.33	7.43	337.20	4191.53	6.36	2.48	4	717420	3803	29	33	15814
S101D2	100	10803.39	9.99	3.71	2349.97	9.00	0.00	0	627692	7627	0	1	2673
S101D3	100	10801.30	12.50	5.06	2192.47	10.66	0.00	0	798817	4903	0	1	2422
S101D5	100	10801.19	16.79	5751.97	687.74	2.36	0.00	2	563396	698	0	123	1153

Table 7: Statistics - Set 2

	Best		BAC		Benders				Δ LB (%)		Δ UB (%)	
Instance	LB	UB	LB	UB	Root LB	Final LB	UB	Opt (%)	BAC	Best	BAC	Best
p01_110	458.00^{c}	458^{c}	458.00	458	454.29	458.00	458	0.00	0.00	0.00	0.00	0.00
p01_1030	753.00^{c}	753^{c}	753.00	753	727.18	742.71	769	3.42	-1.37	-1.37	-2.12	-2.12
p01_1050	978.63^{c}	998^{c}	978.63	998	955.89	973.72	1010	3.59	-0.50	-0.50	-1.20	-1.20
p01_1090	1459.58^{c}	1480^{f}	1459.58	1481	1448.01	1455.43	1552	6.22	-0.28	-0.28	-4.79	-4.86
p01_3070	1438.13^{c}	1473^{c}	1438.13	1473	1430.86	1437.83	1548	7.12	-0.02	-0.02	-5.09	-5.09
p01_7090	2075.84^{f}	2142^{f}	2067.56	2212	2115.69	2118.43	2195	3.49	2.46	2.05	0.77	-2.47
p02_110	612.00^{c}	612^{c}	612.00	612	597.10	605.71	613	1.19	-1.03	-1.03	-0.16	-0.16
p02_1030	1044.54^{f}	1157^{c}	1042.65	1157	1038.13	1049.30	1133	7.39	0.64	0.46	2.07	2.07
p02_1050	1433.98^{f}	-	1412.11	-	1425.56	1439.43	1532	6.04	1.93	0.38	-	-
p02_1090	2212.47^{f}	-	2211.37	-	2234.27	2237.36	2413	7.28	1.18	1.12	-	-
p02_3070	2133.99^{f}	-	2114.04	-	2139.89	2143.92	2388	10.22	1.41	0.47	-	-
p02_7090	3103.35^{f}	3205^{f}	3030.72	-	3148.43	3151.48	3333	5.45	3.98	1.55	-	-3.99
p03_110	739.07^{c}	760^{c}	739.07	760	732.14	738.01	757	2.51	-0.14	-0.14	0.39	0.39
p03_1030	1295.46^{c}	-	1295.46	-	1342.92	1349.55	1472	8.32	4.18	4.18	-	-
p03_1050	1826.56^{c}	-	1826.56	-	1874.31	1881.45	2076	9.37	3.01	3.01	-	-
p03_1090	2849.55^{c}	-	2849.55	-	2952.91	2956.68	3236	8.63	3.76	3.76	-	-
p03_3070	2818.38^{c}	-	2818.38	-	2847.89	2851.91	3067	7.01	1.19	1.19	-	-
p03_7090	4062.61^{c}	-	4062.61	-	4243.18	4243.43	4629	8.33	4.45	4.45	-	-
Avg.								5.83				

Table 8: Objectives - Set 3

problem, while the LP relaxation of the original formulation plus the linking constraints constitutes the Benders subproblem. The subproblem can be solved given an optimal solution to the master problem, and any resulting fractionality can be eliminated by introducing upper and lower bounds in the subproblem. The coordination of the master and subproblem is embedded in a BAB tree, where the branching decisions made in the subproblem create two new nodes and information regarding infeasible subproblems is conveyed through feasibility cuts in the master problem. We demonstrate that such an approach allows one to exploit the branching capabilities of the original problem, while at the same time exploiting the computational superiority of the reformulation (or that of a relaxed formulation).

In addition, this paper identifies several applications of the proposed methodology and proves the correctness of the developed solution approach. Our approach guarantees an LP relaxation bound at least as good as the tighter of the two combined formulations. We also consider how to include cutting planes in each of the two separate formulations, and discuss how to extend the method with advanced problem specific solution approaches such as specialized branching

		Time (s)						Cuts					
Instance	$\|V\|$	Total	Heur.	Ben.	Cap.	Con.	k	Ben.	Cap.	Con.	k	Sub	Nodes
p01_110	50	3.93	2.96	0.54	0.13	0.00	0.00	0	125	0	0	2	13
p01_1030	50	10800.11	4.01	367.15	2389.08	39.74	0.23	28	2750684	42672	4	221	58751
p01_1050	50	10800.21	4.51	2232.72	1668.04	24.35	0.97	60	2405515	19736	12	659	39032
p01_1090	50	10800.61	6.31	9724.45	90.88	1.87	0.65	245	228598	465	5	1937	5940
p01_3070	50	10801.00	6.20	9689.72	115.35	2.38	0.41	144	293431	817	3	1706	6044
p01_7090	50	10800.75	8.35	9822.31	61.81	1.40	0.00	76	287413	97	0	1051	3476
p02_110	75	10800.58	5.18	9.20	5063.48	14.42	0.02	0	1789165	15729	1	4	45335
p02_1030	75	10800.77	6.57	4.15	2165.75	14.16	0.30	0	1121430	12054	2	1	9966
p02_1050	75	10801.30	8.10	60.03	1975.74	11.15	0.00	0	1331373	9429	0	2	6410
p02_1090	75	10802.70	11.50	9101.10	186.25	1.06	0.00	29	255310	310	0	397	1478
p02_3070	75	10803.25	11.25	8492.79	276.87	1.84	1.36	34	359912	820	2	482	2131
p02_7090	75	10804.40	15.40	8784.98	99.84	0.99	0.00	30	289432	54	0	296	1313
p03_110	100	10800.02	7.72	23.07	4208.22	7.20	0.08	0	718146	5327	1	3	14532
p03_1030	100	10801.67	10.27	2.97	2362.39	7.68	0.00	0	729553	6696	0	1	2632
p03_1050	100	10809.02	13.42	1880.33	1711.15	5.13	0.00	1	687827	3153	0	44	2332
p03_1090	100	10807.19	18.59	8330.99	339.79	1.43	0.00	12	329344	586	0	148	872
p03_3070	100	10814.66	17.96	7356.82	472.31	1.75	0.00	14	384400	702	0	112	921
p03_7090	100	10800.62	24.32	7855.28	200.78	1.03	0.00	21	365912	54	0	116	710

Table 9: Statistics - Set 3

Instance	LB	UB	Gap (%)	Nodes	Cap.	Con	Time (s)
eil22	375.00	375	0.00	0	43	0	1
eil23	569.00	569	0.00	0	22	1	1
eil30	510.00	510	0.00	1248	182	148	33
eil33	835.00	835	0.00	580	163	38	51
eil51	521.00	521	0.00	524	658	15	523
S51D1	458.00	458	0.00	67	133	3	47
S76D1	592.00	592	0.00	4784	3677	480	10305
p01_110	458.00	458	0.00	10	66	0	19
p02_110	601.24	806	25.40	4021	3539	245	10691

Table 10: Results $\mathcal{SDVRP}3$

strategies and column generation.

The proposed solution method is applied to the CSP and the SDVRP and demonstrates potential in both cases. We show for the CSP that the approach preserves the structure of the pricing problem, if the master problem is solved using a BAP approach. A comparison with an alternative, exact BAP procedure on a set of publicly available instances indicates that the proposed method is a competitive alternative, finding solutions much faster in some cases. Furthermore, the results also reveal that the branching strategy based on feasibility cuts in Benders decomposition can cause large BAB trees. Future work should thus focus on how to strengthen the cuts. For the SDVRP the approach also shows promise, producing solutions of good quality in general. The SDVRP case study does, however, emphasize the importance of being able to quickly detect infeasibility of a feasible solution to the master problem. Feasible solutions to relaxations that are feasible with respect to the LP-relaxation of the original formulation have the potential to induce large BAB trees. When using the approach with a relaxation it is therefore important to be able to obtain a tight description of the convex hull of integer solutions to the subproblem. Future work will therefore be dedicated to improvements of this general approach as well as an assessment of the algorithm's performance on other classes of problems. Candidates for this include the Capacitated Arc Routing Problem and the k-splittable flow problem.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable feedback. Addressing the points that were raised has significantly improved the quality of the paper.

References

Alves, C., De Carvalho, J.M.V., 2008. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. Computers and Operations Research 35, 1315–1328.

Applegate, D., Bixby, R., Chvátal, V., Cook, W., 2003. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems. Mathematical Programming 97, 91–153.

Archetti, C., Bianchessi, N., Speranza, M.G., 2014. Branch-and-cut algorithms for the split delivery vehicle routing problem. European Journal of Operational Research 238, 685 – 698.

Archetti, C., Speranza, M.G., Hertz, A., 2006. A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science 40, 64–73.

Belenguer, J.M., Martinez, M.C., Mota, E., 2000. A lower bound for the split delivery vehicle routing problem. Operations Research 48, 801–810.

Belov, G., 2003. Problems, Models and Algorithms in One- and Two-Dimensional Cutting. Ph.D. thesis. Technischen Universitaet Dresden, Germany.

Ben Amor, H., Desrosiers, J., Valerio de Carvalho, J.M., 2006. Dual-optimal inequalities for stabilized column generation. Operations Research 54, 454–463.

Ben Amor, H., Valério de Carvalho, J.M., 2005. Cutting stock problems, in: Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), Column Generation, Springer US. pp. 131–161.

Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238–252.

Valério de Carvalho, J., 1999. Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operations Research 86, 629–659.

Christofides, N., Mingozzi, A., Toth, P., 1979. The vehicle routing problem. Combinatorial Optimization , 315–38, 315–338.

Dantzig, G., Wolfe, P., 1960. Decomposition principle for linear-programs. Operations Research 8, 101–111.

Dror, M., Laporte, G., Trudeau, P., 1994. Vehicle routing with split deliveries. Discrete Applied Mathematics 50, 239 – 254.

Dror, M., Trudeau, P., 1990. Split delivery routing. Naval Research Logistics 37, 383–402.

Dueck, G., 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104, 86–92.

Falkenauer, E., 1996. A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics 2, 5–30. doi:10.1007/bf00226291.

Fisher, M.L., 1973. Optimal solution of scheduling problems using lagrange multipliers: Part ii, in: Elmaghraby, S.E. (Ed.), Symposium on the Theory of Scheduling and Its Applications, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 294–318.

Fortz, B., Poss, M., 2009. An improved benders decomposition applied to a multi-layer network design problem. Operations Research Letters 37, 359–364.

Fukasawa, R., Longo, H., Lysgaard, J., De Aragão, M.P., Reis, M., Uchoa, E., Werneck, R.F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical programming 106, 491–511.

Gilmore, P.C., Gomory, R.E., 1963. A linear programming approach to the cutting stock problem - part ii. Operations Research 11, 863–888.

Held, M., Karp, R.M., 1971. The traveling-salesman problem and minimum spanning trees: Part ii. Mathematical Programming 1, 6–25.

Jin, M., Liu, K., Bowden, R.O., 2007. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. International Journal of Production Economics 105, 228–242.

Kantorovich, L.V., 1960. Mathematical methods of organizing and planning production. Management Science 6, 366–422.

Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack problems. Springer.

Lee, C., Epelman, M., White, C., Bozer, Y., 2006. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. Transportation Research Part B 40, 265–284.

Lübbecke, M., Desrosiers, J., 2005. Selected topics in column generation. Operations Research 53, 1007–1023.

Mak, V., 2007. Iterative variable aggregation and disaggregation in ip: An application. Operations Research Letters 35, 36–44.

Moreno, L., de Aragão, M.P., Uchoa, E., 2010. Improved lower bounds for the split delivery vehicle routing problem. Operations Research Letters 38, 302 – 306.

Naddef, D., Rinaldi, G., 2002. Branch-and-cut algorithms for the capacitated vrp, in: The vehicle routing problem. SIAM, pp. 53–84.

Naoum-Sawaya, J., Elhedhli, S., 2013. An interior-point benders based branch-and-cut algorithm for mixed integer programs. Annals of Operations Research 210, 33–55.

Ozbaygin, G., Karasan, O., Yaman, H., 2018. New exact solution approaches for the split delivery vehicle routing problem. Euro Journal on Computational Optimization 6, 85–115.

Pereira do Valle, N.M., 2012. Split Delivery Vehicle Routing Problem. An exact method. Master's thesis. Technical University of Denmark. DTU Transport.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers and Operations Research 34, 2403–2435.

Potvin, J.Y., Rousseau, J.M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66, 331–340.

Ralphs, T., Kopman, L., Pulleyblank, W., Trotter, L., 2003. On the capacitated vehicle routing problem. Mathematical Programming 94, 343–359.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Santini, A., Ropke, S., Hvattum, L.M., 2018. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. Journal of Heuristics 24, 783–815.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems, in: CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming), pp. 417–431.

Silva, M.M., Subramanian, A., Ochi, L.S., 2015. An iterated local search heuristic for the split delivery vehicle routing problem. Computers and Operations Research 53, 234–249.

Truffot, J., Duhamel, C., 2008. A branch and price algorithm for the k-splittable maximum flow problem. Discrete Optimization 5, 629–646.

Vanderbeck, F., 2011. Branching in branch-and-price: a generic scheme. Mathematical Programming 130, 249–294.

Villeneuve, D., Desrosiers, J., Lübbecke, M.E., Soumis, F., 2005. On compact formulations for integer programs solved by column generation. Annals of Operations Research 139, 375–388.