CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Branch-and-Price for Service Network Design with Asset Management Constraints

Jardar Andersen
Roar Grønhaug
Marielle Christiansen
Teodor Gabriel Crainic

December 2007

CIRRELT-2007-55

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Branch-and-Price for Service Network Design with Asset Management Constraints

Jardar Andersen[1,*], Roar Grønhaug[1], Marielle Christiansen[1], Teodor Gabriel Crainic[2]

[1] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, 7491 Trondheim, Norway

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and NSERC Industrial Research Chair on Logistics Management, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8

Abstract. We present a branch-and-price algorithm for solving the service network design problem with asset management constraints, SNDAM. The asset management constraints may result from vehicle management in applications from transportation. The problem is a multicommodity network design problem with additional constraints restricting and connecting the integer design variables. As a result of these constraints, design variables are cycles in a cyclic time-space network, while the flow variables are paths in the same network. Separate subproblems generate design cycles and paths for commodity flows to the restricted master problem. The computational study shows that we are able to find near-optimal solutions for large network instances.

Keywords. Service network design, branch-and-price, column generation, vehicle management, asset management.

Acknowledgements. This work has received financial support from The Norwegian Research Council through the Polcorridor Logchain project. Partial funding has also been supplied by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Discovery and Industrial Research Chair programs.

---

* Corresponding author: jardar.andersen@iot.ntnu.no

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

Introduction

Improvements in solution techniques and computational capacity continuously increase the range of tractable optimization problems. Simultaneously, real-world planning problems challenge the OR community to study increasingly larger and more complex optimization problems. New requirements from practical applications result in new constraints that have to be modeled, and this also encourages development of solution methods that can solve the new models. In particular, there has recently been an increase in new service network design models for planning problems in transportation. What these models have in common is an increased focus on utilization and management of vehicles in addition to the traditional decisions on service selection and flow routing. One issue that has been studied in the literature is ensuring that there is an equal number of vehicles entering and leaving each node in the network. Constraints ensuring this are referred to as design balance constraints (Pedersen and Crainic, 2007). Design balance constraints have been modeled for various modes of transportation, see e.g. (Smilowitz et al., 2003), (Lai and Lo, 2004), (Barnhart and Schneur, 1996) and (Kim et al., 1999). Andersen et al. (2007a) introduce more aspects of vehicle management in network design, including fleet sizing and -management.

Andersen et al. (2007b) present a framework for Service Network Design with Asset Management constraints (SNDAM), which is a richer problem than the problem studied in (Pedersen and Crainic, 2007). *Asset* is a generic term for something a person or an organization owns. In carrier terms, it refers to what we usually call resources. It may be a power unit (a tractor or a locomotive), a carrying unit (a railcar, a trailer, a truck, a ship, etc.), a loading/unloading unit (cranes in terminals), a crew, etc. The term *asset management* reflects that assets have to be managed intelligently in order to obtain efficient operations.

The SNDAM model of (Andersen et al., 2007b) addresses the case when only one asset is controlled, and to increase readability we refer to the assets as “vehicles”. The SNDAM model allows for cycles as design variables corresponding to vehicle rotations in time-space networks, and the computational study of (Andersen et al., 2007b) indicates that cycle variables contribute to efficient model solving. Cycle variables have also been implemented for other problems. For instance, Sigurd et al. (2005) work on a cycle representation of ship schedules in a pickup and delivery problem, and use column generation to generate schedules. Agarwal and Ergun (2006) introduce cycle variables for a cargo routing problem in liner shipping, where cycles represent ship schedules. They report promising computational results based on column generation and benders decomposition.

The focus in (Andersen et al., 2007b) was on model development, as the main objective was to compare alternative SNDAM formulations. The computational study was based on a priori enumeration of cycles and paths, which imposes huge memory requirements for larger instances, and the enumeration of cycles is impractical for large problems. There is thus a need for more advanced solution strategies that can solve realistically-sized problems of this character.

The purpose of this paper is to address the need for advanced solution methods for the SNDAM problem that was presented in (Andersen et al., 2007b), and we therefore develop a tailormade branch-and-price algorithm for the SNDAM problem. A variable-fixing technique is introduced to enhance the performance of the algorithm. We are able to find good integer solutions for problems that are considerably larger than what was solved in (Andersen et al., 2007b).

The contribution of the paper is a tailormade solution approach for the SNDAM problem that can solve realistically-sized problems. The computational study indicates that the algorithms are able to produce good integer solutions within reasonable computational time. The algorithms thus lay the ground for improved planning of problems where these additional constraints occur, for instance in service network design studies where fleet management is considered.

The outline of this paper is as follows. In Section 1, we recall the SNDAM formulation. In Section 2, we propose a branch-and-price algorithm for solving SNDAM. The computational study is presented in Section 3, while concluding remarks follow in Section 4.

1 Service Network Design with Asset Management

In this section we recall the service network design problem with asset management constraints, SNDAM, that was presented in (Andersen et al., 2007b). A problem description follows in Section 1.1, while we in Section 1.2 recall two SNDAM formulations from (Andersen et al., 2007b).

1.1 Problem description

In the classical service network design formulation (see e.g. (Crainic, 2000)) we consider a set of services that may be opened to accommodate a demand for flow through the system. Major decisions are selection of services and routing of the flow on services. Crainic (2000) differentiated between *frequency* and *dynamic* service network design models. Frequency models address strategic/tactical issues like what services to operate and how often to operate them. Dynamic models target planning of schedules and are typically concerned with *when* services depart. The SNDAM model is developed within a dynamic service network design setting. A given *planning horizon* is assumed, and the services have to be operated in a repetitive manner representing fixed schedules of real-world transportation services.

Demand in the system is defined in terms of commodities (products) requiring transport through the network. Each commodity has an associated volume that has to be transported from the commodity's origin node to its destination node. The commodities have specific times where they become available, but they may arrive at their destinations at any time, as long as they are transported to their destinations within the length of the planning horizon.

Operation of services requires *vehicles*, which are available in a limited quantity. There is thus a need to consider a set of vehicle management issues when the services are designed, which extend the formulations in (Crainic, 2000). Firstly, we have to make sure that for all terminals, there is an equal number of vehicles entering and leaving, referred to as vehicle balance or *design balance*. Moreover, there may not be more simultaneous activities taking place than the given fleet of vehicles allow for. A third issue from (Andersen et al., 2007b) is the existence of lower and upper bounds on the number of *occurrences* of each service, for instance that a service should be operated at least 3 times and at most 7 times each week.

A fourth aspect from (Andersen et al., 2007b) is limited durations of vehicle routes, which may be captured by a *route length* requirement. The idea is that vehicles should not have longer routes than the planning horizon considered, as illustrated in Figure 1. In Figure 1, we have three vehicles operating in a cyclic time-space network, which is representing five terminals and a planning horizon divided into seven time periods. In Figure 1, design balance is satisfied for all nodes. However, we observe that the vehicles represented with dotted arcs interchange the arcs they cover in every second realization of the planning horizon. It thus takes two repetitions of the planning

horizon for a vehicle to return to its initial pattern. The operations in Figure 1 represented with dotted arcs will not be feasible if we impose route length requirements. However, the combination of solid arcs in Figure 1 represents an operation of a vehicle that satisfies the route length requirements.

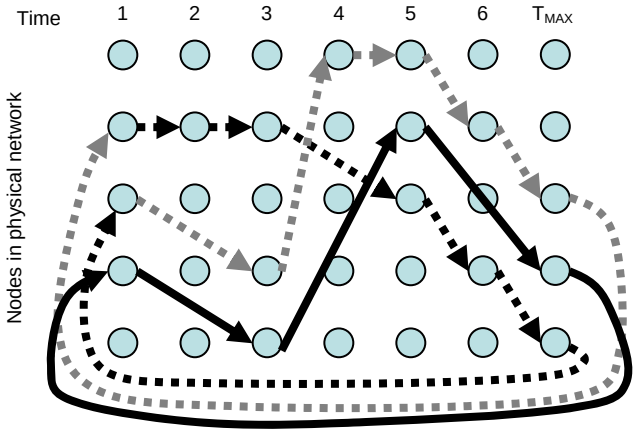

Figure 1. Time-space network with two two-horizon cycles and one 1-horizon cycle.

In traditional service network design studies, one considers *service selection cost* associated with operation of services, and *flow costs* associated for commodities utilizing services (Crainic, 2000). In the SNDAM problem of (Andersen et al., 2007b), it is assumed that there are high fixed costs associated with vehicles, and that these costs dominate the service selection costs. This is based on a planning problem in the rail freight industry, where the acquisition costs for locomotives appeared to be the dominant cost factor. Inclusion of service selection costs would however not introduce any structural changes to the SNDAM models that we recall in the next subsection.

1.2 SNDAM models

We assume that we have a static network $G' = (N', A')$ with nodes representing terminals and intersections, and arcs representing connections. The planning horizon is divided into a set of time periods $T \equiv \{1,..,T_{MAX}\}$, and we introduce the graph $G = (N \times A)$ for the time-space network. A node $i' \in N'$ in the static network has T_{MAX} realizations in the time-space network. Each of these nodes $i \in N$ in the time-space network has an associated time period, $T_i \in T$, and represents a physical node $N_{OD_i} \in N'$. The arcs $(i', j') \in A'$ in the static network are also available in T_{MAX} realizations, one for each time period. We consider one repetition of the planning horizon with time period T_{MAX} preceding time period 1, giving a *cyclic time-space network*.

In time-space networks, holding arcs are required when modeling real-world planning problems. Holding arcs represent vehicles or flow units kept at a static node from one time period to the next. In the following we assume that the models always have holding arcs for the flow, so that flow units may be kept at a physical node from one time period to the next. Thus, the holding arcs link consecutive representations in time of the same physical node. Holding arcs are assumed to have infinite capacity both for vehicles and for flow. We do not complicate the model formulation with notation and variables for the holding arcs.

Without loss of generality we assume that all service arcs $(i, j) \in \mathrm{A}$ are design arcs. To simplify the presentation, we assume that the arcs $(i', j') \in \mathrm{A}'$ represent services, hence we do not consider services consisting of multiple arcs. The different time realizations $(i, j) \in \mathrm{A}$ represent alternative departure times of the service connections $(i', j') \in \mathrm{A}'$. Lower and upper bounds on the number of occurrences of each service are labeled $L_{i'j'}$ and $U_{i'j'}$, respectively. For the demand, we define for each commodity $p \in \mathrm{P}$ the volume w^p to be transported from the commodity's origin node o^p to its destination node d^p.

In order to model the asset management constraints, we introduce *vehicles* $v \in \mathrm{V}$, available in a quantity of $V_{MAX} = |\mathrm{V}|$. Any service arc that is opened has to utilize one of these vehicles. For each vehicle $v \in \mathrm{V}$ we introduce a binary decision variable δ_v, which is 1 if vehicle v is utilized, and 0 otherwise. The cost for utilizing a vehicle is f.

In classical service network design formulations, there are binary variables indicating whether a service arc is opened or not. In order to satisfy the route length requirements, these variables have to be indexed by vehicle in the SNDAM formulation, and are labeled y_{ijv}. Flow variables x_{ij}^p are nonnegative real numbers. Each design arc y_{ijv} has an associated capacity u_{ij}. For each unit of commodity p there is a flow cost c_{ij}^p for traversing arc $(i, j) \in \mathrm{A}$. For each node we define sets $\mathrm{N}^+(i) = \{ j \in \mathrm{N} : (i, j) \in \mathrm{A} \}$ and $\mathrm{N}^-(i) = \{ j \in \mathrm{N} : (j, i) \in \mathrm{A} \}$ of outward and inward neighbors. We also define $b_{ij}^p = \min \{ w^p, u_{ij} \}$. The SNDAM model is recalled in (1) – (11):

$$Min\ z = \sum_{(i,j)\in \mathrm{A}} \sum_{p\in \mathrm{P}} c_{ij}^p x_{ij}^p + \sum_{v\in \mathrm{V}} f \delta_v \tag{1}$$

$$\sum_{(i,j)\in \mathrm{A}: T_i \le t < T_j :} y_{ijv} - \delta_v = 0, \qquad \forall t \in \mathrm{T}, v \in \mathrm{V}, \tag{2}$$

$$\sum_{j\in \mathrm{N}^+(i)} y_{ijv} - \sum_{j\in \mathrm{N}^-(i)} y_{jiv} = 0, \qquad \forall i \in \mathrm{N}, v \in \mathrm{V}, \tag{3}$$

$$\sum_{v\in \mathrm{V}} y_{ijv} \le 1, \qquad \forall (i, j) \in \mathrm{A}, \tag{4}$$

$$L_{i'j'} \le \sum_{(i,j)\in \mathrm{A}: N_{OD_i} = i' \cup N_{OD_j} = j'} \sum_{v\in \mathrm{V}} y_{ijv} \le U_{i'j'}, \qquad \forall (i', j') \in \mathrm{A}', \tag{5}$$

$$\sum_{j\in \mathrm{N}^+(i)} x_{ij}^p - \sum_{j\in \mathrm{N}^-(i)} x_{ji}^p = \begin{cases} w^p, & i = o^p \\ -w^p, & i = d^p \\ 0, & otherwise \end{cases}, \qquad \forall p \in \mathrm{P}, i \in \mathrm{N}, \tag{6}$$

$$\sum_{p\in \mathrm{P}} x_{ij}^p - \sum_{v\in \mathrm{V}} u_{ij} y_{ijv} \le 0, \qquad \forall (i,j) \in \mathrm{A}, \tag{7}$$

$$x_{ij}^p - \sum_{v\in \mathrm{V}} b_{ij}^p y_{ijv} \le 0, \qquad \forall (i,j) \in \mathrm{A}, p \in \mathrm{P}, \tag{8}$$

$$x_{ij}^p \ge 0, \qquad \forall (i,j) \in \mathrm{A}, p \in \mathrm{P}, \tag{9}$$

$$y_{ijv} \in \{0,1\}, \qquad \forall (i,j) \in \mathrm{A}, v \in \mathrm{V}, \tag{10}$$

$$\delta_v \in \{0,1\}, \qquad \forall v \in \mathrm{V}. \tag{11}$$

The objective function (1) minimizes the sum of flow costs and costs associated with use of vehicles. Constraints (2) state that for each time period, if a vehicle is utilized, it should be engaged in one and only one activity, corresponding to the route length requirements. Constraints (3) are the vehicle balance constraints, stating that for each node, there is an equal number of vehicles entering and leaving. In constraints (4) we ensure that only one vehicle can operate an arc $(i,j) \in \mathrm{A}$, which reflects the property that design arcs in traditional network design formulations are binary variables. In (5) we present the lower and upper bounds on the number of occurrences of services while flow balance, as found in traditional network design formulations, is ensured in (6). Constraints (7) and (8) are weak and strong forcing constraints, respectively, where we need to aggregate over all vehicles used. Finally, variable-type constraints are given in (9) – (11).

In (Andersen et al., 2007b), four different formulations of the SNDAM model were presented In this section we recall the formulation with design cycles and flow paths as decision variables, which in (Andersen et al., 2007b) produced the most promising computational results, and which was significantly faster solved than (1) – (11). A design cycle consists of a set of design arcs satisfying design balance constraints (3) and covering each time period exactly once. The latter is a necessary condition for avoiding multi-planning-horizon cycles, which would violate the route length requirements. The design variable g^k for cycle $k \in \mathrm{K}$ is 1 if cycle k is in the solution and 0 otherwise. Moreover, parameter m_{ij}^k is 1 if arc $(i,j) \in \mathrm{A}$ is in cycle k. The cycles have to cover all time periods in the time-space representation, as exemplified with the cycle indicated with solid arcs in Figure 1. As is frequently done in network design (see e.g. (Ahuja et al., 1993)), we further define the set of paths $\mathcal{L}^p$ that commodity p may use from its origin node to its destination node. For these paths, we define parameters a_{ij}^{pl} =1 if arc $(i,j) \in \mathrm{A}$ belongs to path $l \in \mathcal{L}^p$ for commodity p, 0 otherwise. The flow of commodity p on path l is h^{pl}, while the flow cost for transporting

commodity p on path l is k^{pl}, $k^{pl} = \sum_{(i,j)\in A} c_{ij}^{p} a_{ij}^{pl}$. The SNDAM model with cycle and path variables is recalled in (12) – (20):

$$Min\ z = \sum_{p\in P} \sum_{l\in \mathcal{L}^p} k^{pl} h^{pl} + \sum_{k\in K} f\ g^k \tag{12}$$

$$\sum_{k\in K} g^k \le V_{MAX}, \tag{13}$$

$$\sum_{k\in K} m_{ij}^k g^k \le 1, \qquad \forall (i,j) \in A, \tag{14}$$

$$L_{i'j'} \le \sum_{k\in K} \sum_{(i,j)\in A: N_{OD_i}=i' \cup N_{OD_j}=j'} m_{ij}^k g^k \le U_{i'j'}, \qquad \forall (i',j') \in A', \tag{15}$$

$$\sum_{l\in \mathcal{L}^p} h^{pl} = w^p, \qquad \forall p \in P, \tag{16}$$

$$\sum_{p\in P} \sum_{l\in \mathcal{L}^p} a_{ij}^{pl} h^{pl} - \sum_{k\in K} u_{ij} m_{ij}^k g^k \le 0, \qquad \forall (i,j) \in A, \tag{17}$$

$$\sum_{l\in \mathcal{L}^p} a_{ij}^{pl} h^{pl} - \sum_{k\in K} b_{ij}^{p} m_{ij}^k g^k \le 0, \qquad \forall (i,j) \in A, p \in P, \tag{18}$$

$$h^{pl} \ge 0, \qquad \forall p \in P, l \in \mathcal{L}^p, \tag{19}$$

$$g^k \in \{0,1\}, \qquad \forall k \in K. \tag{20}$$

The objective function (12) minimizes the sum of flow costs on paths and fixed costs for vehicles that are utilized. The structurally new constraint compared to model (1) – (11) is (13). This constraint restricts the cycle selection, stating that the number of selected cycles is limited by the fleet of vehicles. This property was ensured by the size of set $V = \{1,..,V_{MAX}\}$ in model (1) – (11). Constraints (14) state that each arc $(i,j) \in A$ can be chosen by at most one cycle, analogous to (4). Lower and upper bounds on the number of realizations of static arcs are formulated in (15), and demand satisfaction is ensured in (16). Weak and strong forcing constraints are found in (17) and (18), while variable-type constraints appear in (19) – (20). Note that the introduction of cycles and paths has removed the need for design balance constraints (3) and flow balance constraints (6). Moreover, because each cycle corresponds to a vehicle and fleet size is accounted for in (13), we no longer need the binary δ_v variables.

Magnanti and Wong (1984) show that the uncapacitated fixed charge network design problem is *NP*-hard. As the capacitated version is even harder (Balakrishnan et al., 1997), this problem also belongs to the class of *NP*-hard problems. There is no reason to believe that the problem becomes easier to solve when additional asset management constraints are introduced.

2 Solving SNDAM with branch-and-price

Among four formulations of the SNDAM model in (Andersen et al., 2007b), formulation (12) – (20) appeared to be more easily solved than the other formulations. However, the computational study was limited to instances that could be handled with a priori enumeration of cycles and paths. For real-world planning problems, complete enumeration is impracticable. In this section we develop a branch-and-price algorithm for the SNDAM problem, consisting of a column generation algorithm embedded in a branch-and-bound framework. The algorithm is based on formulation (12) – (20), and the columns that are generated are design cycles and flow paths. This branch-and-price framework addresses the need for an exact solution algorithm that may solve real-world instances of the SNDAM problem. Several topics in column generation are covered in (Desaulniers et al., 2005), and general introductions to branch-and-price and column generation can be found in (Barnhart et al., 1998) and (Lübbecke and Desrosiers, 2005).

We recall basic principles of column generation and branch-and-price in Section 2.1. Sections 2.2-2.4 are devoted to the solution of the linear relaxation of the problem; we define the restricted master problem in Section 2.2 and subproblems in Section 2.3 and 2.4, respectively. Aspects of the branch-and-price algorithm for the integer problem are presented in Sections 2.5-2.7. Branching strategies are discussed in Section 2.5, while strategies for obtaining lower and upper bounds are presented in Sections 2.6 and 2.7, respectively.

2.1 Branch-and-price

Column generation usually refers to the solution of a linear problem, and for (mixed) integer problems this may correspond to solving the linear relaxation of the underlying problem. To obtain integer solutions, the column generation is embedded in a branch-and-bound framework. Because of the pricing of columns within the column generation at each branch-and-bound node, we refer to the overall approach as *branch-and-price*. In this subsection we recall the fundamentals of column generation and branch-and-price.

2.1.1 Column generation

Column generation is a powerful tool for solving optimization problems with a huge number of variables. The basic principle is to work on a restricted version of the problem including all the rows, but only a subset of the variables. This restricted problem is referred to as the *Restricted Master Problem* (RMP). The variables that are not explicitly represented in the RMP are considered implicitly by evaluation of one or more *subproblems* (pricing problems) that evaluate what impact it would have to include new variables in the RMP given the current solution of the RMP, and add variables to the RMP if that contributes to an improved solution of the RMP. If no new variables will contribute to an improved solution of the RMP, we can conclude that the existing solution of the RMP is the optimal (linear) solution of the original formulation. Because the variables occur as columns in the master problem, this process is referred to as *column generation.* We sketch the column generation process in Figure 2. We first have to initialize the RMP with a feasible basis, and then dual prices are transferred to the subproblems. If the subproblems identify columns that would improve the solution of the RMP, these columns are added to the RMP and the process is repeated. Otherwise, the process is ended and the current solution to the RMP is the optimal one.

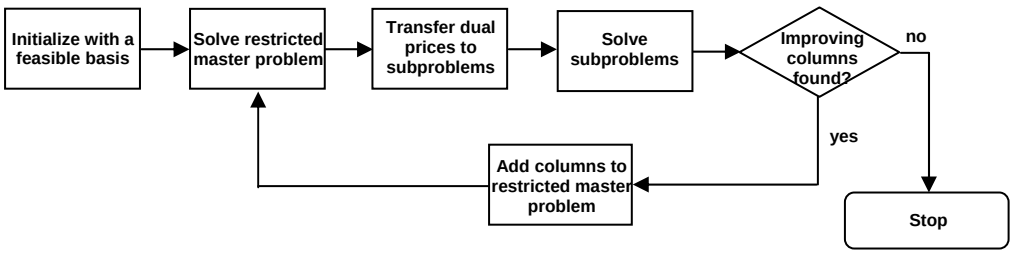

Figure 2. Flow chart for linear column generation.

2.1.2 Branch-and-price algorithm

We present a branch-and-price flowchart in Figure 3. We start by adding the linear relaxation of the initial problem to a pool of unexplored nodes, this is the *root node* problem. In a major iteration of the algorithm, one node is extracted from this pool and solved. If the solution is worse than the current upper bound, we fathom the node and extract a new node from the pool. Otherwise, we check whether the solution is integral. If so, we have a new upper bound, otherwise we have to branch to create new problems that are added to the pool of unexplored nodes.

The box in Figure 3 with thick bold frame contains the column generation algorithm illustrated in Figure 2. In other words, the column generation is run for each branch-and-bound node. If the pool of unexplored nodes is empty or if the lower bound is equal to the upper bound, the search is terminated. If the upper bound has been updated during the search, this is the optimal integer solution. Otherwise, no feasible integer solution exists.

The lower bound is based on solution of the linear problem, and the lower bound is initially set to the solution value of the first problem solved (corresponding to the root node). Updates depend on the search strategy; with a best-first search the lower bound can be updated at each node. In contrast, with a depth-first strategy, the lower bound can only be updated each time the current top of the tree is reached after backtracking.

We infer from the branch-and-price literature (see e.g. (Lübbecke and Desrosiers, 2005)), that branching directly on the integer variables is not likely to succeed in branch-and-price. If we for instance branch on the binary cycle variables in (12) – (20), we fix the cycle to 1 in one branch and 0 in the other branch. In the branch with the cycle fixed to 0, that same column will immediately be regenerated, and we will not have reached further towards an integer solution. In addition, the search tree will be unbalanced. The standard branching technique in branch-and-price is to branch on the underlying network structure, which for the SNDAM problem means branching on network arcs. We elaborate on branching strategies in Section 2.5.

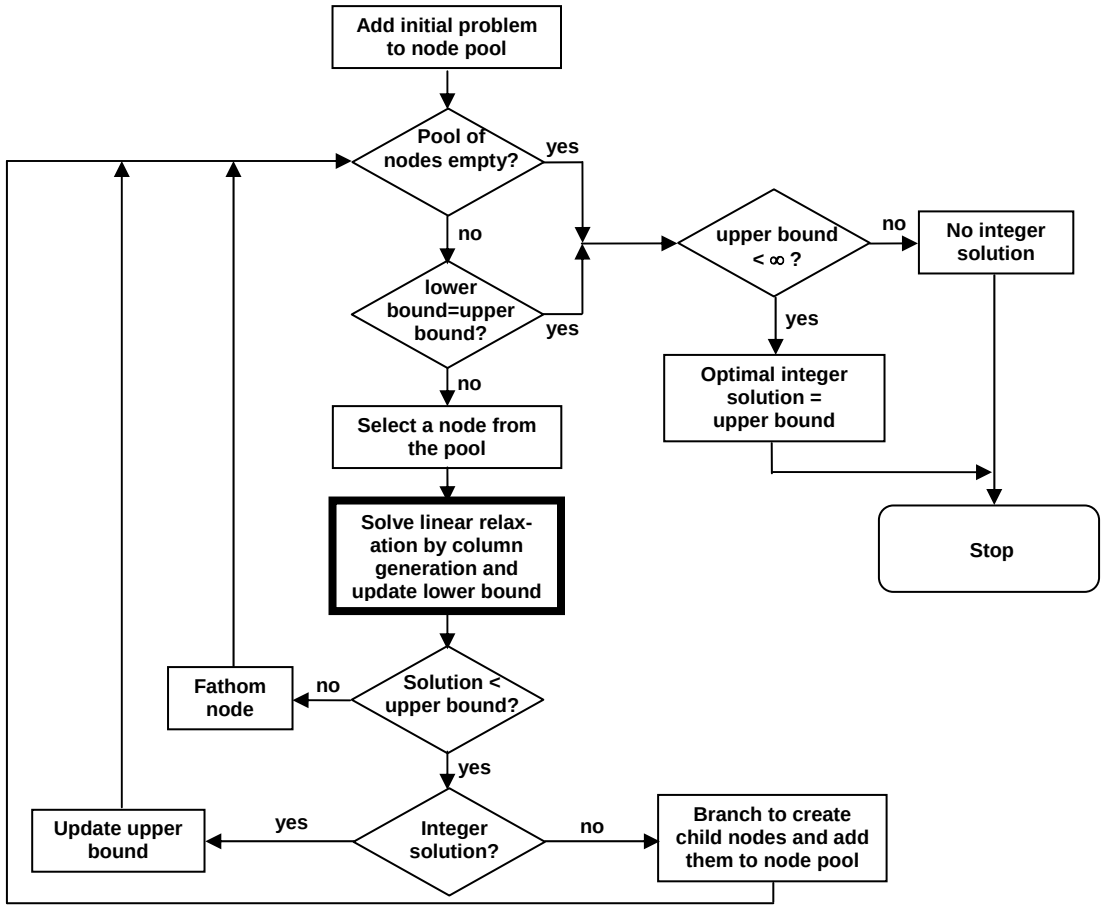

Figure 3. Flow-chart for branch-and-price algorithm.

2.2 Restricted master problem

The restricted master problem (RMP) is a linear relaxation of the SNDAM problem with cycle and path variables, formulated in (12) – (20). However, in order to reduce the size of the matrix, we initially exclude the strong forcing constraints (18) from the formulation. For cycles and paths, we work on subsets from the total amount of cycles and paths; $\tilde{\mathrm{K}} \subseteq \mathrm{K}$ and $\tilde{\mathcal{L}}^p \subseteq \mathcal{L}^p$, respectively. To obtain the linear relaxation we replace integrality constraints (20) with (20b). We also reformulate (15) to (15b) by introducing slack variables $s_{i'j'}\ \forall (i', j') \in \mathrm{A}'$, $0 \le s_{i'j'} \le (U_{i'j'} - L_{i'j'})$. The RMP can be solved with commercial LP-solvers, but the solution times may be substantial for large instances.

$$\sum_{k \in \mathrm{K}} \sum_{(i,j) \in \mathrm{A}: N_{OD_i} = i' \cup N_{OD_j} = j'} m_{ij}^k g^k + s_{i'j'} = U_{i'j'}, \qquad \forall (i', j') \in \mathrm{A}', \tag{15b}$$

$$0 \le g^k \le 1, \qquad \forall k \in \tilde{\mathrm{K}}. \tag{20b}$$

We associate dual variables α, β_{ij}, $\theta_{i'j'}$, σ^p, η_{ij} and ρ_{ij}^p with constraints (13) – (14), (15b) and (16) – (18). From the solution of the RMP, we transfer the dual information to subproblems for generation of design cycles and flow paths. There is one subproblem for each commodity generating flow paths, and one subproblem for design cycles. If the subproblems identify variables with negative reduced costs, these are added to the RMP which is resolved. Otherwise, the current solution of the RMP is an optimal solution for the linear problem.

Because we have two different subproblem categories, there are basically two different approaches of the column generation. The first approach is to solve both cycle and path generation subproblems for a given set of dual multipliers before resolving the RMP. The second approach is to generate either paths or cycles first, and then resolve the RMP before solving the other subproblem(s). Presumably, the second approach would perform relatively better if the RMP is solved efficiently. In the next two subsections we define the two categories of subproblems.

2.3 Subproblem for design cycle generation

A design cycle consists of a set of arcs satisfying the design balance constraints, and not covering a time period more than once. In the subproblem for generation of design cycles, we need to make sure that these properties are maintained. The reduced cost of a cycle is calculated in (21), and a negative reduced cost implies that inclusion of the cycle in the RMP would improve the objective function value of the current RMP. The subproblem identifies the design cycle with smallest reduced cost, and this cycle is added to the RMP if its associated reduced cost is negative. The subproblem for cycle generation is formulated in (21) – (24):

$$Min\ \bar{z}_C = f - \alpha - \sum_{(i,j)\in A}\left(\beta_{ij} - u_{ij}\eta_{ij} - \sum_{p\in P} b_{ij}^p \rho_{ij}^p + \theta_{i'j'}\Big|_{i'=N_{OD_i}\cup j'=N_{OD_j}}\right) y_{ij} \tag{21}$$

$$\sum_{(i,j)\in A:T_i\le t<T_j:} y_{ij} \le 1, \qquad \forall t \in T, \tag{22}$$

$$\sum_{j\in N^+(i)} y_{ij} - \sum_{j\in N^-(i)} y_{ji} = 0, \qquad \forall i \in N, \tag{23}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{24}$$

We use binary variables y_{ij} representing whether an arc $(i,j) \in A$ in the underlying network is selected to be in the cycle or not. The objective function (21) minimizes the sum of fixed vehicle costs f and all dual cost of the arcs constituting a cycle. Constraints (22) ensure that cycles do not cover multiple planning horizons by stating that at most one arc can be opened in each time period. Constraints (23) are the design balance constraints, while (24) restrict the y_{ij}'s to take binary values.

We solve the cycle generation problem with shortest path calculations using a label-correcting algorithm. We extend the time-space network beyond the planning horizon by the duration of the longest arc in the network. With the new network, we are able to produce any feasible cycle by one

traversal of the network. The construction of cycles can be handled by two different phases that together capture all potential cycles.

Firstly, if we start from all nodes in time period 1 and create paths through the network, we obtain cycles when we find arcs that have their destination node in the origin node of that path, i.e. crossing the planning horizon. If all arcs have duration of one time period, this approach would be sufficient to generate all possible cycles.

However, for arcs with longer duration, we may have the case that arcs "pass by" time period 1, for instance if an arc has its origin in the last time period, passes the end of the planning horizon and has its destination node in time period 2. This arc will not be evaluated with the approach described above. In order to capture such arcs, we identify with the second approach all arcs having the property that they pass by time period 1. We illustrate these ideas in Figure 4.

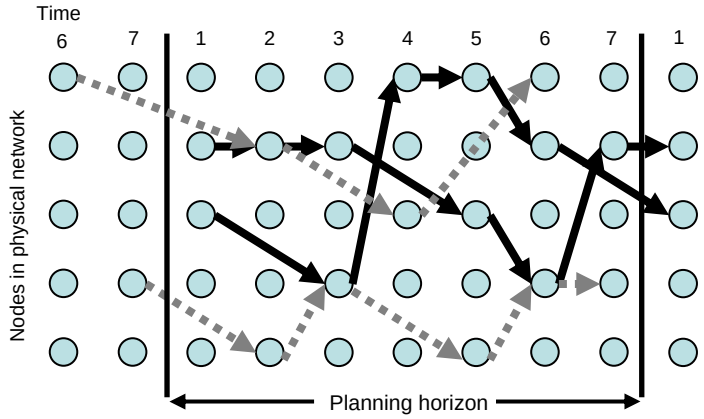

Figure 4. Illustration of cycle generation based on arcs originating in the first time period (solid arcs) and on cross-horizon arcs (dotted arcs)

In Figure 4, solid arcs form cycles that are established from nodes in the first time period, while dotted arcs form cycles that are established based on the cross-horizon arcs that pass by time period 1. Because of constraints (22) we do not continue on the extension of a path if the time period representing the origin node of the path has been reached or passed.

In the cycle generation, we have to be careful with applying dominance rules. The calculation of reduced costs in (21) includes all arcs constituting a cycle, including the last arc that closes a cycle, and we are therefore not able to apply dominance rules between paths with different origins. We therefore keep one label for each node belonging to time period 1, and one additional label for each node that has at least one arc emanating from it that passes by time period 1. In the end, this approach is very well suited for producing several cycles simultaneously.

2.4 Subproblems for flow path generation

The flow path subproblems generate paths that contribute to improved solutions of the current RMP. Requirements to paths are that they have their origins in the commodity's unique origin node in the time-space network, node balance has to be ensured, and the paths have to end in one of the time realizations of the commodity's given physical destination node. The subproblem for flow path generation finds the origin-destination path with smallest reduced cost, as defined in (25). This path

is added to the RMP if the associated reduced cost is negative. The subproblem for generation of commodity paths for commodity *p* is defined in (25) – (27):

$$Min\ \bar{z}_P^p = -\sigma^p + \sum_{(i,j)\in A} \left(c_{ij}^p - \eta_{ij} - \rho_{ij}^p\right) x_{ij}^p \tag{25}$$

$$\sum_{j\in N^+(i)} x_{ij}^p - \sum_{j\in N^-(i)} x_{ji}^p = \begin{cases} 1, & i = o^p \\ -1, & i = d^p \\ 0, & otherwise\ , \end{cases} \qquad \forall i \in N, \tag{26}$$

$$x_{ij}^p \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{27}$$

The x_{ij}^p variables are now binary variables representing flow on service arcs in the underlying network, and $x_{ij}^p = 1$ indicates that $arc(i,j) \in A$ is in the path. The objective function (25) minimizes the sum of dual cost of the demand satisfaction constraints (16) and reduced costs for all arcs constituting the path. Constraints (26) are flow balance constraints, while we in (27) restrict the x_{ij}^p variables to take binary values.

Paths are established by shortest-path computations using label-correcting algorithms. Shortest paths are created from the commodity's unique origin node to the nodes in the time-space network corresponding to the commodity's physical destination node. In principle, all these time realizations need to be examined in order to find the path with the most negative reduced cost. However, as the costs on the arcs are positive, we can terminate if the cost at any stage is higher than the computed cost of reaching the current best destination node. Moreover, if path costs exceed σ_p, we can stop evaluating that path because it cannot result in a path with negative reduced costs.

2.5 Branching strategies for the integer problem

In order to obtain integer solutions for the SNDAM problem, we embed the column generation in a branch-and-bound framework, where the solution obtained from the linear relaxation of the problem represents the *root node* of a search tree. In this subsection we present branching strategies for the problem. As described in Section 2.1, it is not desirable to branch directly on the cycle variables. Instead we consider four alternative branching strategies.

The first approach (BB-1) is to branch on service arcs in the underlying network structure, i.e. arcs in the time-space network. In evaluating the solutions of the RMP, we choose the arc with fractional value closest to 0.5, and fix this arc to 1 in one branch and to 0 in the other branch. The interpretation of this branching is that fixing an arc to 1 means that the sum of the cycle variables using this arc is 1. In the 0-branch, all cycles using this arc have to be 0.

The second approach (BB-2) is an adaptation of constraint branching (Ryan and Foster, 1981), where the basic principle is to branch on two arcs, letting at most one of them be 1 in one branch, and both or none of them be 1 in the other branch.

The third branching strategy (BB-3) is to use the slack variables $s_{i'j'}$ from (15b). For integer solutions to (12)-(20), integrality of $U_{i'j'}$ and of all r_{ij}^k also ensures integrality of the $s_{i'j'}$ variables. When we obtain fractional $s_{i'j'}$ variables in the linear solution, we branch by identifying the $s_{i'j'}$ variable with fractional value closest to 0.5 and round it down to the nearest integer in one branch and round it up to the nearest integer in the other branch. However, even if all $s_{i'j'}$ are integer, we do not necessarily have integer values on the cycles. We therefore need to switch branching entity if all $s_{i'j'}$ are integer, but the solution is not integer in the g^k variables. In such cases, we branch on the arcs in the underlying network structure, which was the first approach.

The last branching strategy (BB-4) is a cycle-based greedy heuristic. We select a fractional cycle and fix all the arcs in the cycle to 1, this allows for more flexibility than fixing the cycle itself to 1. This approach is not a proper branching, because we would never consider cycles with a subset of the arcs that have been fixed further up in the tree, and would thus not cover the complete solution space. We refer to this approach as *heuristic branching*.

2.6 Lower bounds

Capacitated multicommodity network design problems have in general poor linear relaxations. For SNDAM, it was shown in (Andersen et al., 2007b) that the introduction of cycle variables improved the tightness of the linear relaxations. We therefore use the linear relaxation as the lower bound in the branch-and-price approach. However, in contrast to many other problems that have been solved with column generation, even solving the linear problem constitutes a significant challenge when the dimensions increase.

The lower bounds for the SNDAM problem are significantly tighter when they are based on the strong linear relaxation including strong forcing constraints (18), compared to the weak linear relaxation that is obtained when only weak forcing constraints (17) are included. For large networks with many arcs and commodities, the number of strong forcing constraints would grow extremely large if all of them were included explicitly in the problem. In order to overcome this difficulty, we generate dynamically those strong forcing constraints that are violated in the optimal solution of the RMP. To do this, we first solve the weak linear relaxation to optimality with column generation. Then, we verify whether strong forcing constraints are violated in the optimal linear solution, and if so, these are added to the formulation. Then the new problem is solved to optimality with column generation, and the process is repeated if additional strong forcing constraints are needed. When no new strong forcing constraints are needed, we have obtained the strong linear relaxation. The drawback of this approach is that we spend time on testing if these constraints are violated and on reoptimization after new constraints have been added. For large instances, these reoptimizations may become computationally expensive, and an important issue is thus to which degree strong forcing constraints should be generated. These constraints may be generated at all branch-and-bound nodes, at the root node only, each time the lower bound is updated, or none of these.

To strengthen the bound, we introduce the following idea from a vehicle routing setting in (Dumas et al., 1991): The number of vehicles used has to be integer in an integer solution. If the linear solution has a fractional sum of design cycles, we round this fractional value up to the nearest integer, captured in (28) with *m* being the sum of cycle variables in the current LP solution. In addition, we introduce the complement of (28) in (29), and round the fractional sum of cycle

variables down to the nearest integer. The effect of the strengthening with (28) and (29) is most significant if introduction of (29) implies an infeasible problem.

$$\sum_{k \in \mathrm{K}} g^k \geq \lceil m \rceil, \tag{28}$$

$$\sum_{k \in \mathrm{K}} g^k \leq \lfloor m \rfloor. \tag{29}$$

2.7 Upper bounds

The upper bound is updated each time a new best integer solution is found. In order to encourage fathoming, it is of high importance to obtain good integer solutions early in the traversal of the tree. We introduce an accelerating technique with the aim of finding integer solutions quickly and thus improve the performance of the branch-and-price algorithm.

From the solution of the linear RMP, we fix to 1 cycle variables g^k with fractional values above a threshold. Then, new columns are generated until the solution of the RMP cannot be further improved. Again, the fractional g^k variables with values above the threshold are fixed to 1, and this procedure is repeated until an integer solution has been found, the solution is worse than the current best integer solution, or the problem becomes infeasible.

3 Computational study

In this section we present a computational study based on the branch-and-price algorithm that was introduced in Section 2 for the SNDAM problem. We have programmed the algorithms in C++, and the models have been solved on computers with 3 GHz processor and 8 GB RAM running on Rock Cluster v 4.2.1 operating system. The restricted master problems are solved with the LP-solver of XPRESS Optimizer v 17.1. We first discuss implementation and calibration issues in Section 3.1, before giving an introduction to the data sets used in Section 3.2. Computational results appear in Section 3.3.

3.1 Implementation and calibration

In the computational study, we explore three different approaches “A”, “B” and “C”, which are targeted at different instance sizes. One important difference between the approaches is the role of strong forcing constraints. These constraints improve the bound and may also contribute to better branching decisions, so it is desirable to include them. However, for large problems, it is time-consuming to solve even the linear version of the problem if many constraints are added to the problem. The rationale underlying implementations A-C is that strong forcing constraints are generated more often for small instances.

There are a significant number of parameter settings and issues that could be explored in a computational study, but we have kept a focus on the role of the strong forcing constraints in the implementations. The three solution approaches A-C are developed through extensive testing in a calibration phase, and the parameter selections are based on what has been observed to give reasonable performance over a set of instances. It was however not possible to perform full-scale

testing of all combinations of the different parameters and issues. We summarize significant implementation issues and parameter selections in Table 1.

Table 1.Implementation issues and parameter selections.

Issues and parameters	Solution approach A	Solution approach B	Solution approach C
Search strategy	Depth first		
Maximum search time	36000 seconds		
How often is the master problem solved	All subproblems are solved for given dual information before the master problem is resolved		
Strategy for solving RMP at each BB-node	Save basis before adding columns, and reoptimize with dual simplex from this basis when columns have been added to the problem		
Max number of paths generated in each iteration	One per commodity		
Branching strategies	BB-3 (slack variables)	BB-3 (slack variables) / BB-4 (heuristic)	BB-4 (heuristic)
Max and min values for fixing cycles to 1 in acc. technique	0.7 and 0.55		0.6 and 0.55
Max number of cycles generated in each iteration	5*		10*/20*/30*
How often are strong forcing constraints generated	All nodes	Nodes where lower bounds are updated	Nodes where lower bounds are updated
How long do we keep on generating strong forcing constraints at a node?	As long as they are violated		No new iterations after 18000 seconds or if improvement < 0.1%
Keep strong forcing constraints in problem after generation?	Yes		No

*In the accelerating technique at most one cycle may be added in each iteration

For the three approaches, we implemented a depth-first strategy in branch-and-price, and we have set a maximum running time of 10 hours (36000 seconds). In the computations it appeared to be more efficient to solve all subproblems before reoptimizing the RMP, so we do not report results for the second approach described in Section 2.2. In solving the RMP, we save the basis before adding columns, and when columns have been added, we reoptimize with dual simplex from the saved basis. As pointed out in Sections 2.3 and 2.4, it is computationally cheap to add more than one cycle or path when the subproblems are solved. There is a trade-off between the gains of saving iterations as a consequence of adding multiple columns, versus the increased computational time from having more columns in the master problem. There are a significant amount of commodities in our data sets, and we therefore allow inclusion of at most one path for each commodity in an iteration of the column generation. However, because of the many commodities, it makes sense to include multiple cycles in each iteration if there are several cycles with negative reduced costs. We allow most cycles in solution approach C, which is targeted at the instances with the largest number of commodities.

For branching, the calibration phase indicated that branching on service arcs in the underlying network (BB-1) and constraint branching (BB-2) was outperformed by branching on slack variables (BB-3). The computational study is therefore limited to branching strategies BB-3 and BB-4, where BB-4 is applied to the largest instances.

For solution approach A, we use branching strategy BB-3. For the variable fixing technique described in Section 2.7, the limit for fixing cycles is initially set to 0.7, and sequentially reduced to 0.55. In each iteration of the column generation, at most 5 cycles may be added. Strong forcing

constraints are generated on all nodes, and we keep generating them as long as they are violated. The strong forcing constraints are also kept in the problem at subsequent nodes.

Solution approach B may be used either with BB-3 or BB-4. The additional difference from solution approach A is that strong forcing constraints are only generated at nodes where the lower bounds are updated during branch-and-bound. This allows exploration of more nodes within the time limit.

Solution approach C is targeted at very large problem instances, and it utilizes branching strategy BB-4. The variable fixing technique starts with a limit for fixing cycles at 0.6, and to facilitate instances with several hundred commodities, we test inclusion of at most 10, 20 and 30 cycles in one iteration of the column generation algorithm. Strong forcing constraints are only generated when the lower bound is updated, and because the reoptimizations are time-consuming, we include the opportunity to stop this process if the time exceeds 18000 seconds or if the improvement in objective function value in the last iteration was below 0.1%. Moreover, the strong forcing constraints are removed from the matrix as soon as the bound is updated, to facilitate faster exploration of the search tree.

3.2 Data sets

In (Andersen et al., 2007b) the SNDAM problem with cycle and path variables was solved directly with the MIP-solver in CPLEX for problem instances where all design cycles and commodity paths were generated a priori. We test the branch-and-price algorithm on a few of the instances from (Andersen et al., 2007b), but the computational study is focused on instances that are significantly larger than what could be solved with full a priori generation. In Table 2 we present the dimensions of the problems that are tested in the computational study. The first three columns of Table 2 present number of nodes and arcs in the static network, as well as number of time periods. In the fourth column we present the corresponding number of service and holding arcs, and number of commodities appears in the sixth column. In the seventh column a *size indicato*r for problem dimension is presented, computed as the product of number of service arcs and number of commodities and divided by 1000. This measure gives a fair description of the relative dimensions of the instances. In the last column we indicate which of the solution approaches of Section 3.1 we apply to each instance.

The instances are inspired by a real-world case in rail transportation planning. Instances 1-5 are extracted from (Andersen et al., 2007b), while instances 6-15 are significantly larger than what could be handled with total enumeration of cycles and paths. We observe that there are significant differences in problem size between the instances. As pointed out in Section 3.1, initial testing has suggested that different approaches should be applied depending on the sizes of the instances.

Instances 1-5 have size indicators in the range 4-45, and for these problems we apply solution approaches A and B. However, for solution approach B we only apply the branching on slack variables (BB-3). For instances 6 to 12, the size indicator is in the range 120-450, which is significantly larger than for instances 1-5. For these cases, solution approach A is left out, as it for these instances appeared to be disadvantageous to generate strong forcing constraints on all nodes of the search tree. Finally, for instances 13-15, the size indicator exceeds 1000, and these instances are so large that we only apply solution approach C.

Table 2.Dimensions of problem instances.

Problem id #	Static nodes	Static service arcs	Time periods	Service + holding arcs	Commodities	Size indicator	Solution approaches
1	5	10	20	200 + 100	20	4	A and B
2	5	15	20	300 + 100	25	8	A and B
3	5	15	25	375 + 125	25	9	A and B
4	5	15	15	225 + 75	100	23	A and B
5	5	15	15	225 + 75	200	45	A and B
6	5	15	40	600 + 200	200	120	B
7	5	15	50	750 + 250	400	300	B
8	7	30	30	900 + 210	200	180	B
9	7	30	30	900 + 210	400	360	B
10	7	30	50	1500 + 350	300	450	B
11	10	40	30	1200 + 300	200	240	B
12	10	50	30	1500 + 300	100	150	B
13	7	30	60	1800 + 420	800	1440	C
14	10	50	30	1500 + 300	1000	1500	C
15	10	50	50	2500 + 500	400	1000	C

3.3 Results

In this section we present results from model runs for the problem instances that were defined in Table 2. We allow a maximum CPU time of 10 hours in all cases. In Section 3.3.1 we present results for problems that have been solved with a priori generation of columns, while we in Section 3.3.2 present results for the problems that have not been solved earlier.

3.3.1 Instances solved with a priori generation of columns

Results from model runs for instances 1-5 can be found in Table 3. For each scenario, there is one row with results obtained with solution approach A, and one row with results obtained with solution approach B. In each case, we report lower bound at termination and best MIP solution at termination. Termination either refers to that 10 hours of CPU time has been reached, or that the algorithm has found a proven optimal integer solution. Thereafter the solution time in seconds for finding proven optimal solution is returned, or alternatively the remaining optimality gap after 10 hours of CPU time. In the last two columns of Table 3, we present the best integer solution that was obtained with a priori enumeration of cycles and paths in (Andersen et al., 2007b) and the corresponding solution time. The a priori enumeration was implemented on different computers but with comparable specifications and a maximum running time of 10 hours.

The results reported in Table 3 are very similar for approaches A and B. For instances 1-3, the branch-and-price algorithm returns proven optimal integer solutions with both solution approaches. For instances 1 and 3, the solutions are obtained reasonably quickly, while instance 2 requires around 25000 and 20000 seconds with the two approaches. The a priori enumeration was more efficient for instances 1 and 2, while instance 3 was solved faster with branch-and-price. The a priori enumeration utilized the advantages of a standard MIP-solver with advanced heuristics and more advanced search algorithms than we have implemented. It is nevertheless good news that the branch-and-price algorithm returns proven optimal integer solutions for problems where these optimal

solutions are known. For instances 4 and 5, neither a priori enumeration nor branch-and-price returned proven optimal solutions within 10 hours. For instance 4, the integer solutions obtained were slightly better with a priori enumeration, and the gap also smaller in this case due to more advanced bounding. For instance 5, the branch-and-price algorithm returns better integer solutions than what was obtained with a priori enumeration. However, the remaining optimality gap after 10 hours is smaller with a priori enumeration; this reflects the stronger focus on bounding in a commercial MIP-solver compared to the branch-and-price algorithm presented in this paper. The encouraging news from Table 3 is however that the branch-and-price algorithm solves small instances to proven optimality, and that both approaches A and B return better integer solutions than what was obtained with a priori enumeration for the largest instance.

Table 3. Results from model runs for instances that have been computed with a priori generation of columns.

Instance	Solution approach	Lower bound at termination	MIP solution at termination	Solution time (sec) / Optimality gap	Best MIP with a priori enumeration	Time use a priori enumeration (sec) / Optimality gap
1	A	48 838	48 838	98	48 838	9
	B	48 838	48 838	129		
2	A	52 156	52 156	25 134	52 156	1235
	B	52 156	52 156	19 799		
3	A	47 805	47 805	68	47 805	143
	B	47 805	47 805	69		
4	A	171 532	174 697	1.8 %	174 233*	0.7%
	B	171 532	174 697	1.8 %		
5	A	378 347	380 848	0.66 %	381 533*	0.5%
	B	378 347	381 002	0.70 %		

**For these instances, the a priori approach did not return a proven optimal solution within 10 hours.*

3.3.2 Larger instances

We present results from model runs for problem instances 6-12 in Table 4. In these model runs, all integer solutions have been obtained by use of the accelerating technique presented in Section 2.7. No problems are solved to proven optimum within 10 hours of CPU time. Again, there are two rows for each instance, representing branching on slack variables (BB-3) and heuristic branching (BB-4).

In Table 4, the first column of results presents the lower bounds at termination, which in all cases corresponds to the lower bound obtained at the root node, because the depth-first search does not return to the root node within 10 hours for any of the instances. Then best integer solutions obtained and optimality gaps are reported. The next two columns elaborate on the best integer solutions that were obtained, and report node number where the best integer solution was found and the elapsed time, respectively. In the last two columns we report number of branch-and-price nodes visited during the tree search, and time needed to solve the root node. For each branching strategy, we present average results from instances 6-12 in the last two rows.

Table 4. Results from model runs for instances 6-12. Maximum computational time is 10 hours.

Instance	Branching strategy	Lower bound	Best MIP-solution	Optimality gap	Best MIP-node	Best MIP-time	Nodes visited	Root time
6	Slack variables	334 026	338 314	1.3 %	3	115	1 143	67
	Heuristic	334 026	339 198	1.5 %	217	7 373	3 828	85
7	Slack variables	465 997	482 250	3.4 %	14	29 020	19	15 558
	Heuristic	465 997	482 427	3.4 %	12	28 706	18	15 306
8	Slack variables	360 466	376 049	4.1 %	11	1 853	1 905	381
	Heuristic	360 466	374 804	3.8 %	6	1 173	912	379
9	Slack variables	363 332	381 307	4.7 %	21	23 968	34	2 522
	Heuristic	363 332	377 775	3.8 %	69	29 597	129	2 525
10	Slack variables	504 792	528 755	4.5 %	11	27 026	15	6 791
	Heuristic	504 792	524 450	3.7 %	10	26 114	14	6 775
11	Slack variables	412 463	434 118	5.0 %	10	12 235	35	2 156
	Heuristic	412 463	439 179	6.1 %	10	11 528	30	2 157
12	Slack variables	284 211	292 223	2.7 %	32	4 706	416	219
	Heuristic	284 211	291 677	2.6 %	163	15 745	300	218
Average	Slack variables	389 327	404 717	3.7 %	15	14 132	510	3 956
	Heuristic	389 327	404 216	3.6 %	70	17 177	747	3 921

From Table 4 we observe that the optimality gaps range from 1.3% to 6.1%, and the averages over all the instances are 3.7% and 3.6% with the two branching strategies. The optimal integer solutions are not known for these problems, but from similar problem types in (Andersen et al., 2007b) we observed that the gaps between strong linear relaxation of the root node and optimal integer solutions were in the ranges 0-9%, with an average of about 3%. It is thus likely that the integer solutions reported in Table 4 represent near-optimal solutions to the instances. This assumption is supported by reviews of computational studies of similar problems, for instance reported in (Ghamlouche et al., 2004) and (Pedersen et al., 2007).

We observe from Table 4 that there are significant differences between the instances in root time and number of nodes visited. The computed averages reveal that the two branching strategies are very different when it comes to “Best MIP-node”. The average optimality gaps are about similar in the two cases, but the searches based on heuristic branching continue finding integer solutions later than the searches based on slack variable branching.

In Table 5 we report results that were obtained for instances 13-15. For each instance, solution approach C is tested with three alternative maximum number of cycles that may be added in each iteration of the column generation. The columns of Table 5 are identical to those of Table 4, except that one new column is added with an indication of whether the improvements from weak to strong linear relaxations in the root node were stopped because the time limit was exceeded, because there was no significant improvements, or not stopped at all. We observe from Table 5 that bounding was stopped in all model runs.

Table 5. Results from model runs for instances 13-15. Maximum computational time is 10 hours.

Instance	Max no of cycles	Bounding stopped by	Lower bound	Best MIP-solution	Optimality gap	Best MIP-node	Best MIP-time	Nodes visited	Root time
13	10	No improvement	733 310	850 732	13.8%	3	16 375	21	13 825
	20	No improvement	733 373	849 862	13.7%	5	14 509	31	10 519
	30	No improvement	733 373	850 921	13.8%	24	30 081	32	9 681
14	10	Time	630 828	708 283	10.9%	11	32 725	12	19 269
	20	No improvement	631 294	727 652	13.2%	4	18 628	13	15 633
	30	No improvement	631 304	705 243	10.5%	9	30 725	11	13 760
15	10	Time	633 392	n.a.	n.a.	n.a.	n.a.	1	36 112
	20	Time	633 416	792 926	20.1%	3	32 107	5	26 524
	30	Time	633 626	773 617	18.1%	3	31 560	5	26 225

For all instances, integer solutions were found. The only approach that did not return an integer solution was when at most 10 cycles could be generated in each iteration for instance 15. In this case only the root node could be explored within the time limit, and the accelerating technique did not return a feasible integer solution in that single attempt. The optimality gaps reported in Table 5 are larger than those of Table 4. Fewer nodes are visited in the searches because the solution time of the linear problem at each node of branch-and-price is more time-consuming. In addition, it might be that the bound could have been tightened more if we allowed more time for generation of strong forcing constraints and reoptimizations. This applies to instance 15 which has the largest gaps. Despite this, it is promising that the algorithm returns fairly good integer solutions for these large instances.

The results for the large instances indicate that we are able to find good integer solutions also for these. However, the computational study has demonstrated the need for different adaptations of the branch-and-price algorithm based on the sizes of the instances considered.

4 Concluding remarks

In this paper we have developed a branch-and-price algorithm for solving the service network design problem with asset management constraints (SNDAM). Such problems arise for instance when decisions on vehicle management are considered jointly with service network design, and represent a potential for improved planning of transportation systems.

We have compared the branch-and-price algorithm to earlier work based on a priori enumeration of columns. The branch-and-price algorithm gives comparable solutions to those obtained with a priori enumeration of columns, but the smallest instances are solved more efficiently with commercial code. However, the branch-and-price algorithm is able to find near-optimal solutions for instances that are significantly larger than what could be solved with a priori enumeration of columns. An accelerating technique that is introduced within branch-and-price contributes to finding good integer solutions fast. The promising results from the computational study indicate that the presented algorithm may contribute to improved planning of large-scale operations.

However, there is still a need for improvements of the methodology that is developed to solve even larger instances of the problem. One interesting way forward would be to improve the accelerating technique that is introduced to enhance the performance of the algorithm. This

technique can be made more advanced, for instance by reversing earlier decisions on variable fixing during the accelerating technique in order to obtain diversity. Ideas from other heuristics may also be introduced in this environment. One challenge for the solution approach presented in this paper is that the solution time for the linear problem is significant, in particular due to multiple reoptimizations when strong forcing constraints are generated within the column generation process. Moreover, network design problems suffer in general from weak lower bounds, and even if the true optimal integer solution was obtained during branch-and-price, the search may continue for a considerable time if the lower bound is weak. Introduction of cuts may be an interesting way forward to obtain smaller optimality gaps also for the problem studied in this paper.

Due to the potential savings that can be achieved by closer integration of service network design and vehicle management, we strongly encourage further research on these issues. The formulation based on cycles also can be used for problems without requirements for a maximum vehicle route length corresponding to the length of the planning horizon. In such cases, the model must allow for cycles covering multiple planning horizons, in the worst case with a number of periods corresponding to the fleet size. However, the major ideas of the formulation and the solution algorithm still apply. We also suggest that the ideas that are brought forward with design cycles could be introduced for other problems. Grouping design arcs into design paths could be an interesting idea even for problems without a time-space representation and for general network design formulations.

Acknowledgements

This work has received financial support from The Norwegian Research Council through the Polcorridor Logchain project. Partial funding has also been supplied by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Discovery and Industrial Research Chair programs.

References

Agarwal, R., Ö. Ergun. 2006. Ship scheduling and network design for cargo routing in liner shipping. Working paper, Georgia Institute of Technology.

Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. Network Flows. Prentice Hall, Upper Saddle River, NJ.

Andersen, J., T.G. Crainic, M. Christiansen. 2007a. Service network design with management and coordination of multiple fleets. To appear in *European Journal of Operational Research*.

Andersen, J., T.G. Crainic, M. Christiansen. 2007b. Service network design with asset management: formulations and comparative analyzes. Report 2007-21, CIRRELT, Université de Montréal.

Balakrishnan, A., T.L. Magnanti, P. Mirchandani. 1997. Network Design. In *Annotated bibliographies in combinatorial optimization*, Dell'Amico, M., F. Maffoli F, S. Martello (eds), John Wiley & Sons: New York, NY.

Barnhart, C., R.R. Schneur. 1996. Air network design for express shipment service. *Operations Research* 44, 852- 863.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46, 316–329.

Crainic, T.G. 2000. Service network design in freight transportation. *European Journal of Operational Research* 122, 272-288.

Desaulniers, G., J. Desrosiers, M.M. Solomon. 2005. Column generation. GERAD 25th anniversary series. GERAD 25th Anniversary Series. Springer.

Dumas, Y., J. Desrosiers, F. Soumis. 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research* 54, 7-22.

Ghamlouche, I., T.G. Crainic, M. Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* 131, 109-133.

Kim, D., C. Barnhart, K. Ware, G. Reinhardt. 1999. Multimodal express package delivery: a service network design application. *Transportation Science* 33, 391-407.

Lai, M.F., H.K. Lo. 2004. Ferry service network design: optimal fleet size, routing and scheduling. *Transportation Research A* 38, 305-328.

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

Magnanti, T.L., R.T. Wong. 1984. Network design and transportation planning: models and algorithms. *Transportation Science* 18, 1-55.

Pedersen, M.B., T.G. Crainic. 2007. Optimization of intermodal freight train service schedules on train canals. Publication CIRRELT-2007-51. CIRRELT, Université de Montréal.

Pedersen, M.B., T.G. Crainic, O.B.G. Madsen. 2007. Models and tabu search meta-heuristics for service network design with asset-balance requirements. To appear in *Transportation Science*.

Ryan, D.M., B.A. Foster. 1981. An integer programming approach to scheduling. In *Computer Scheduling of Public Transport*, Wren, A. (ed), North-Holland Publishing Company, 1981.

Sigurd, M., N.L. Ulstein, B. Nygreen, D.M. Ryan. 2005. Ship scheduling with recurring visits and visit separation requirements. In *Column generation*, Desaulniers, G., J. Desrosiers, M.M. Solomon (eds), GERAD 25th Anniversary Series. Springer.

Smilowitz, K.R., A. Atamtürk, C.F. Daganzo. 2003. Deferred item and vehicle routing within integrated networks. *Transportation Research E* 39, 305-323.