CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Service Network Design with Resource Constraints

Teodor Gabriel Crainic
Mike Hewitt
Michel Toulouse
Duc Minh Vu

November 2012

CIRRELT-2012-63

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
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Service Network Design with Resource Constraints

Teodor Gabriel Crainic[1,2,*], Mike Hewit[3], Michel Toulouse[1,4], Duc Minh Vu[1,5]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[3] Kate Gleason College of Engineering, Rochester Institute of Technology, James E. Gleason Building, 77 Lomb Memorial Drive, Rochester, NY, USA 14623-5603

[4] Department of Computer Science, Oklahoma State University, 700 North Greenwood Avenue, North Hall 328, Tulsa, OK 74106-0700, USA

[5] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

Abstract. We first present a new service network design model for freight consolidation carriers, one that both routes commodities and the resources needed to transport them while explicitly recognizing that there are limits on how many resources are available at each terminal. We next present a solution approach that combines column generation, meta-heuristic, and exact optimization techniques to produce high-quality solutions. We demonstrate the efficacy of the approach with an extensive computational study and benchmark its performance against a leading commercial solver.

Keywords: Stochastic programs, network design, progressive hedging, machine learning.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Discovery Grant program and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec – Nature et technologies (FRQNT) through their infrastructure grants, Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 Introduction

Service network design formulations are extensively used to address planning issues within many application fields, in particular for the tactical planning of operations of consolidation-based modal and multimodal carriers and organizations (e.g., Christiansen et al., 2007; Cordeau et al., 1998; Crainic, 2000, 2003; Crainic and Kim, 2007; Crainic and Laporte, 1997). The main goal of such formulations is to produce an operations (or load) plan that services the estimated demand while achieving the economic and service-quality targets of the carrier. Building such a plan involves principally selecting the services to operate and their schedules (departure times) and routing the demand through the selected service network. Most service network design models proposed in the literature consider the resources required to perform the services (vehicles, power units, drivers, etc.) only indirectly, however, which is increasingly inadequate to reflect the operation strategies of a broad range of transportation systems (e.g., Crainic and Bektaş, 2008).

Only recently have researchers proposed models and solution methods that recognize management issues related to the resources needed to implement a service network (e.g., Andersen et al., 2009a,b). The range of resource resource-management issues considered is still very limited, however. Moreover, introducing resource-management considerations within service network design formulations raises significant methodological challenges far from being satisfactorily addressed. The goal of this paper is to address these challenges by enlarging the range of resource-management aspects included into tactical planning models, and by introducing an efficient and general purpose solution methodology for the *service network design with resource constraints* (*SNDRC*) formulations.

More precisely, our contributions are to 1) propose a new mathematical formulation for the scheduled SNDRC problem explicitly accounting for the limited number of resources available at each terminal and taking advantage of the structure introduced into the model by the resource-management constraints; 2) introduce an advanced matheuristic solution methodology, combining long-term memory-enhanced slope scaling, column generation, and mathematical programming techniques, which is both general for the SNDRC problem class and very efficient for the particular problem at hand; 3) demonstrate this efficiency through a comprehensive experimental study and benchmarking against a leading commercial software.

The paper is organized as follows. Following a brief literature review in Section 2, we state the problem and detail the network model and mathematical formulation in Section 3. Section 4 is dedicated to the proposed matheuristic solution approach, while Section 5 presents the experimental study. We conclude in Section 6.

2 Literature Review

One resource (or asset) management requirement modeled by researchers is that a resource must be available at the origin of a service for the service to be performed. Amongst other things, this enables a model to capture the (sometime) necessity for a resource to first move empty before it can carry a load. Mathematically, it is modeled by requiring that the number of resources entering and leaving a terminal must be the same. Pedersen et al. (2009) refer to these as "design-balance constraints" and present service network design models based on arc and cycle network formulations. Indeed, the design-balance constraints induce a cyclic structure for the resource movements supporting the service operations and the routing of flow. Pedersen et al. (2009) also proposed a two-phase tabu search method for the arc-based formulation. Meta-heuristic algorithms improving over the results of Pedersen et al. (2009) were proposed by Chouman and Crainic (2010) and Vu et al. (2012), the latter being used as a subroutine in the approach presented in this paper. This requirement has been modeled for various modes of transportation, e.g., Barnhart and Schneur (1996); Kim et al. (1999) for air-based express package delivery, Lai and Lo (2004) for ferries, Andersen and Christiansen (2009); Andersen et al. (2009a) for intermodal rail, and Erera et al. (2012); Smilowitz et al. (2003) for trucking.

The cycle structure induced by the design-balance constraints is actually modeling the fact that, in most settings, a resource is associated with a specific terminal in a network and must return there before the end of the planning horizon. An example in trucking is drivers, who, by federal (and sometimes labor union) regulations, must periodically return to the terminal closest to their home. Andersen et al. (2009b) thus study the computational performance of arc and cycle-based formulations, by solving limited-size instances with a commercial mixed integer programming solver, and concluded that the latter offers the most promising approach. Recognizing that a priori complete cycle generation will not scale to large instances, Andersen et al. (2011) then presented an effective branch-and-price scheme for the cycle-based formulation. The problem is NP-hard, however, and even more efficient methods are required. This is the goal of this paper for the problem setting that also extends the range of resource limitations considered.

3 Problem Statement and Model

We model the operations of a carrier with a time-space network, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where terminal activities in different periods (likely days) are modeled with different nodes. Specifically, we assume a set $\mathcal{L}$ of terminals in the carrier's network and that the planning horizon is divided into $\mathcal{T} = \{1, 2, \ldots, TMAX\}$ time periods. We then define the set of nodes, $\mathcal{N}$, to model the operations of terminals in different periods, i.e. $\mathcal{N} = \mathcal{L} \times \mathcal{T} =$

$\{l_t | l \in \mathcal{L}, t \in \mathcal{T}\}$, where l_t represents terminal l at period t.

There are two types of arcs in the set $\mathcal{A}$. The first is a *service arc* and models the operation of a service between two terminals at a particular point in time, whereas the second is a *holding arc* and models the opportunity for a resource or shipment to idle at a terminal from one period to the next. With S denoting the set of possible services between terminals in $\mathcal{L}$, for each $s = (l, m) \in \mathcal{S}$ and $t \in 1, \ldots, TMAX$, we add the arc $(l_t, m_{(t+\pi) \bmod TMAX})$ to $\mathcal{A}$ (assuming the arc length is π (in periods)). As in many transportation planning methods, we assume that the shipment demands seen during the planning horizon will repeat over time and thus, our time-space network "wraps around." Specifically, we model a service in $\mathcal{S}$ of length π that departs from a terminal in period t such that $t + \pi > TMAX$ as arriving at the destination in period $t + \pi \bmod TMAX$. As an example, see the arc originating from T1 in period 7 and terminating at T2 in period 1 in Figure 1.

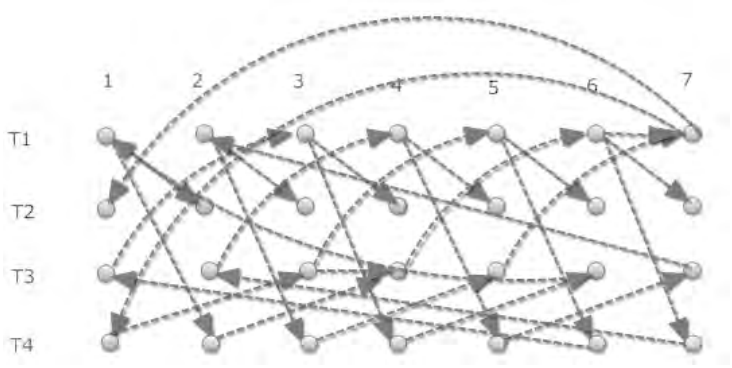


Figure 1: Time-Space Network for Cyclic Service Schedules

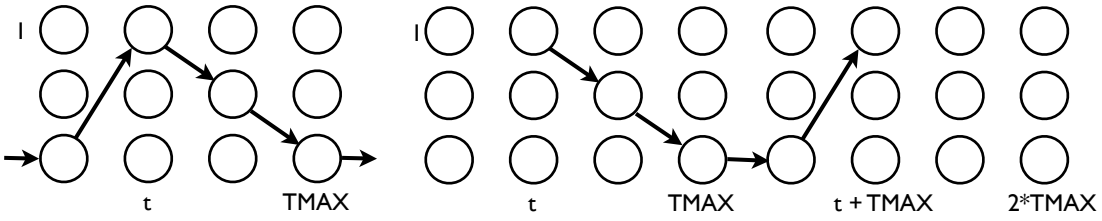
We also assume a limit, u_s, on how much shipment demand can be carried by service s and set the capacity, u_{ij}, of executions of that service at different times to u_s. Regarding the *holding arcs*, we add to $\mathcal{A}$ an arc of the form $(l_t, l_{(t+1) \bmod TMAX})$ for each terminal l and period t. While we assume these arcs are uncapacitated (both with respect to shipment demands and resources) in our experiments, terminal capacities (on shipments or resources) could be modeled by placing capacities on these arcs.

We model a shipment that needs to be delivered from terminal l and available in period t to terminal, m, by period t' as a commodity with index k, origin node $o(k) = l_t$, and destination node $d(k) = m_{t'}$. We denote the size of this shipment as w_k. The set of all shipments is represented by $\mathcal{K}$.

We model three types of costs. The first is a variable cost associated with commodity k traveling on arc $(i, j) \in \mathcal{A}$ and is denoted c_{ij}^k. These costs can model the impact the weight of a shipment can have on the cost of executing a service or the labor costs

associated with handling a commodity at a terminal. The second is a fixed cost associated with the use of a resource, and is denoted by F. These costs can model the cost of paying drivers or the depreciation of a capital asset. The third is a fixed cost associated with executing a service departing from terminal l and arriving at terminal m and is denoted by f_{lm}. These costs can model the actual transportation cost of a resource traveling from terminal l to terminal m.

Like many network design models, for a commodity to travel on arc $(l_t, m_{t'}) \in A$, then that service must be "executed", or, the arc must be "installed" in the network. However, we also model the situation where a resource is needed to execute that service, and these resources travel in cycles. For our experiments, our rules regarding what constitutes a valid cycle are that it must begin and end at the terminal $l \in \mathcal{L}$ and take exactly $TMAX$ periods. We search for a cycle in $\mathcal{G}$ that begins and ends at node l_t by creating a second network, $\mathcal{G}'$ where we append a copy of $\mathcal{G}$ after $\mathcal{G}$ and search for a path in this network from l_t to l_{t+TMAX}. We illustrate the network $\mathcal{G}$ and a cycle that originates at terminal 1 in period t in Figure 2(a) and the corresponding path in $\mathcal{G}'$ in Figure 2(b). Thus, we often describe cycles in $\mathcal{G}$ as paths in $\mathcal{G}'$.



(a) Cycle in $\mathcal{G}$ (b) Path in $\mathcal{G}'$

Figure 2: Correspondence between cycles and paths

We allow a cycle beginning at l_t to return to l multiple times (see the path beginning at terminal 1 in Figure 3), and if it last returns to l in period $t' < t + TMAX$ we append holding arcs so that it reaches l_{t+TMAX} (see the path beginning at terminal 3 in Figure 3). While our rules governing what is a valid cycle are simple, much of the methodology we propose still applies to cases where more complicated rules, such as those representing union or federal regulations of what a driver may do, must be modeled.

We associate with node $l_t \in \mathcal{N}$ a set of cycles, θ_{lt}, that depart from terminal l at period t. We next associate with terminal $l \in \mathcal{L}$ the set $\theta_l = \cup_{t=1}^{TMAX} \theta_{lt}$, or, the set of all cycles that depart from terminal l during the planning horizon. Because a terminal may have a fixed set of resources available during the planning horizon, we also let ub_l denote the maximum number of resources that can depart from terminal l on cycles in θ_l. Finally, r_{ij}^{τ} is a binary indicator of whether arc (i, j) is contained in cycle τ, and $\theta = \cup_{l \in \mathcal{L}} \theta_l$.

We have two types of variables in our model. The first, x_{ij}^k, is a continuous variable

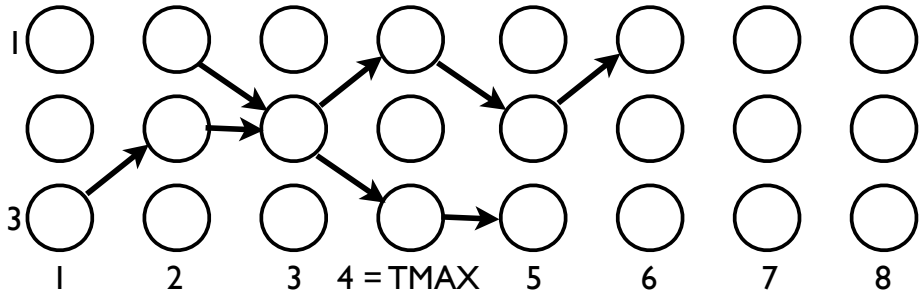


Figure 3: Valid Cycles (shown as paths in $\mathcal{G}'$)

that indicates the amount of commodity k that flows on arc $(i,j) \in \mathcal{A}$. The second, z_τ, is a binary variable that indicates whether cycle $\tau \in \theta$ is selected. Thus, the cycle-based formulation of scheduled service network design with resources constraints (SNDRC), which seeks to select cycles that can carry all commodities from their origins to their destinations and route those commodities, seeks to

$$\text{minimize} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{\tau \in \theta} F z_\tau + \sum_{(i,j) \in \mathcal{A}} \sum_{\tau \in \theta} r_{ij}^\tau z_\tau f_{ij}$$

subject to

$$\sum_{j:(i,j) \in \mathcal{A}} x_{ij}^k - \sum_{j:(j,i) \in \mathcal{A}} x_{ji}^k = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k)\ \ \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \\ -w^k & \text{if } i = d(k) \end{cases} \tag{1}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^k \leq u_{ij} \sum_{\tau \in \theta} r_{ij}^\tau z_\tau, \quad \forall (i,j) \in \mathcal{A}, \tag{2}$$

$$\sum_{\tau \in \theta} r_{ij}^\tau z_\tau \leq 1, \quad \forall (i,j) \in \mathcal{A}, \tag{3}$$

$$\sum_{\tau \in \theta_l} z_\tau \leq ub_l, \quad \forall l \in \mathcal{L}, \tag{4}$$

$$x_{ij}^k \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K}, \tag{5}$$

$$z_\tau \in \{0,1\}, \quad \forall \tau \in \theta. \tag{6}$$

The objective is to minimize the total cost of using resources ($\sum_{\tau \in \theta} F z_\tau$), operating services ($\sum_{(i,j) \in A} \sum_{\tau \in \theta} r_{ij}^\tau z_\tau f_{ij}$), and routing commodities ($\sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k$). Constraint set (1), named the ***flow balance constraints***, ensures that each commodity is routed from its origin node to its destination node. Constraint set (2), named the ***weak forcing constraints***, ensures that a service is executed when it is used by a commodity

and that its capacity is not exceeded. Constraint set (3), named the ***0-1 service constraints***, ensures that each service belongs to at most one executed cycle. Constraint set (4), named the ***resource bound constraints***, ensures that the total number of resources originating from each terminal does not exceed the number available. Lastly, constraint sets (5) and (6) define the domains of the variables. While constraints of the form $x_{ij}^k \leq \min(w^k, u_{ij}) \sum_{\tau \in \theta} r_{ij}^\tau z_\tau$ can significantly strengthen the linear relaxation, there are often too many of them to consider them all explicitly.

4 Solution approach

One of the challenges in producing a high quality solution to the SNDRC is that for even reasonably-sized instances, the set θ is too large to be enumerated. Thus, to produce a high-quality solution to the SNDRC, one needs a method for producing the cycles that appear in good solutions and a method that can produce a high quality solution given a fixed set of cycles. We illustrate the steps of the solution approach in Figure 4, where the number next to each step corresponds to the section where it is described. As indicated in Figure 4, column generation (Barnhart et al., 1998; Desaulniers et al., 2005) is one method used by the solution approach to generate cycles and we next describe how it is done for the SNDRC.

4.1 Column generation

We define SNDRC($\bar{\theta}$) to be the SNDRC restricted to the cycles in $\bar{\theta}$ and its linear relaxation to be the SNDRC with the constraints $z_\tau \in \{0, 1\}, \tau \in \bar{\theta}$ replaced by $0 \leq z_\tau \leq 1, \tau \in \bar{\theta}$. The motivation behind how column generation solves a linear program is that variables need not be explicitly added to the instance until optimality or feasibility conditions dictate that they may be necessary for finding the optimal solution. In particular, assuming $\bar{\theta}$ contains cycles such that SNDRC($\bar{\theta}$) is feasible, SNDRC($\bar{\theta}$) will be repeatedly solved, with new cycles (z_τ variables) added to $\bar{\theta}$ when reduced cost calculations indicate that they may lead to an improved solution.

Thus, we associate the dual variables α_{ij} (≤ 0) with each constraint in set (2), β_{ij} (≤ 0) with each constraint in set (3), and γ_l (≤ 0) with each constraint in set (4). Then, the reduced cost associated with variable z_τ is $\pi_\tau = F - \gamma_l + \sum_{(i,j) \in \tau} (f_{ij} + \alpha_{ij} u_{ij} - \beta_{ij})$. Given these dual values, the search for a cycle with negative reduced cost can be performed by finding the shortest path in the time-space network $\mathcal{G}'$ with respect to arc costs $f_{ij} + \alpha_{ij} u_{ij} - \beta_{ij}$ from l_t to $l_{t+\text{TMAX}}$ for each $l \in \mathcal{L}$. We present the algorithm for solving the linear relaxation of SNDRC in Algorithm 1.

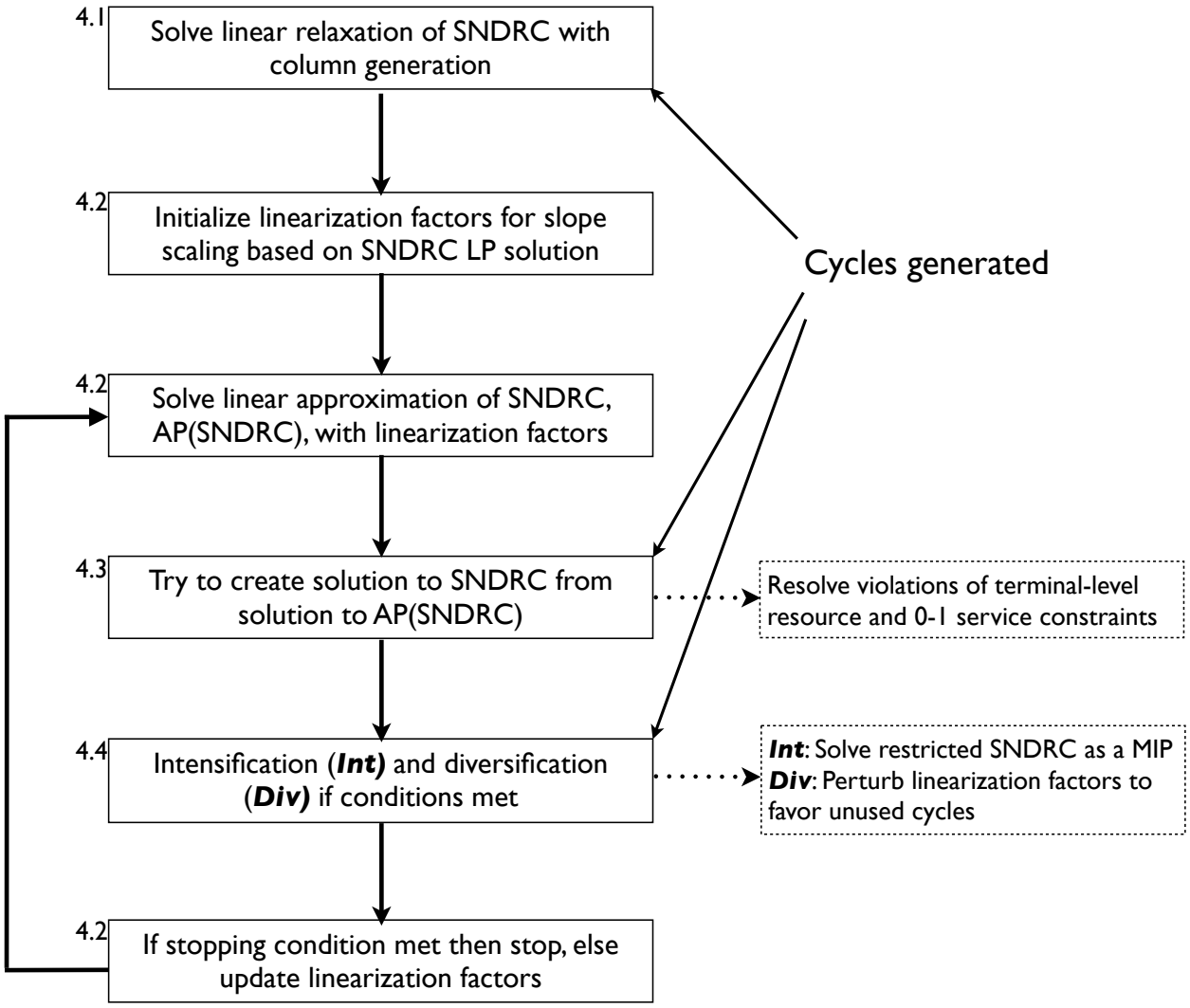


Figure 4: Steps of solution approach

Algorithm 1 Column generation procedure for solving the linear relaxation of SNDRC

```
Find an initial set of cycles θ̄
while have not solved linear relaxation do
  Solve linear relaxation of SNDRC(θ̄)
  Determine dual values
  for all l_t ∈ N do
    Find the shortest path in G from l_t to l_{t+TMAX} with respect to arc costs f_ij +
    α_ij u_ij − β_ij.
    if F − γ_l+ the length of the shortest path < 0 then
      Add the corresponding cycle to θ̄
    end if
  end for
  Stop if no variables with negative reduced cost found
end while
```

At the beginning of Algorithm 1 we generate cycles that can deliver flow for at least one commodity from its source to its destination. Specifically, for each commodity $k \in \mathcal{K}$, originating at $o(k) = l_t$, we generate λ cycles that begin at $o(k)$, pass through $d(k) = m_{t'}$ and then return to $o(k)$. We find these cycles by finding λ shortest paths from $o(k) = l_t$ to $l_{t+\mathrm{TMAX}}$ in $\mathcal{G}'$ with respect to the arc costs f_{ij} for service arcs, 0 for holding arcs not associated with terminal m, and a value $< -1 * \sum_{(i,j)\in\mathcal{A}} f_{ij}$ for holding arcs associated with terminal m. A very negative value is assigned to holding arcs associated with terminal m to ensure each shortest path visits the destination terminal for commodity k. To find multiple shortest paths, we use Yen's algorithm (Yen, 1971).

Solving the linear relaxation of SNDRC yields a bound on the optimal value of SNDRC. However, we also use the cycles generated during this solution process and the solution to the linear relaxation itself to inform a slope scaling procedure that produces high-quality primal solutions. We next describe this slope-scaling procedure.

4.2 Slope scaling

When using slope scaling (Kim and Pardalos, 1999; Crainic et al., 2004) to produce solutions to an integer program, a linear relaxation of the problem is repeatedly solved with an objective function that is parameterized by *linearization factors*. Each time the linear relaxation is solved, a solution to the integer program is constructed (typically through rounding) and the linearization factors are updated so that the cost of the solution to the linear relaxation equals the cost of the solution to the integer program. Our slope scaling approach differs from what is traditionally seen in that instead of solving the linear relaxation of SNDRC, our approach solves an approximation (which is also a linear program) and then executes a separate procedure to construct a feasible solution

to SNDRC from a solution to the approximation. We next describe this approximation problem, called AP(SNDRC).

We build an instance of AP(SNDRC) with a fixed set of cycles $\tilde{\theta}$ and solve it to identify a set of cycles enabling each commodity's demand to flow from its source to its sink. As such we introduce additional (continuous) variables $x_{ij}^{k\tau} \geq 0$ which indicates the amount of commodity k's demand routed on arc (i,j) and carried by cycle τ. We relate these new variables to the variables x_{ij}^{k} with the equation $x_{ij}^{k} = \sum_{\tau \in \tilde{\theta}} x_{ij}^{k\tau}$. With linearization factors $\rho_{ij}^{k\tau}(t)$ that are parameterized by an iteration counter, t, AP(SNDRC($\rho(t)$)) seeks to

$$\text{minimize} \sum_{\tau \in \theta} \sum_{ij \in \mathcal{A}} \sum_{k \in K} x_{ij}^{k\tau} \left(c_{ij}^{k} + \rho_{ij}^{k\tau}(t) \right)$$

subject to

$$\sum_{\tau \in \theta} r_{ij}^{\tau} x_{ij}^{k\tau} = x_{ij}^{k}, \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{K}, \tag{7}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{k} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{k} = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k) \\ -w^k & \text{if } i = d(k) \end{cases} \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \tag{8}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^{k} \leq u_{ij}, \quad \forall (i,j) \in \mathcal{A}, \tag{9}$$

$$x_{ij}^{k\tau} \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K},\ \tau \in \theta. \tag{10}$$

Solving AP(SNDRC(ρ)) in iteration t yields the vector $\tilde{x}_{ij}^{k\tau}(t)$ which can be used to produce a (possibly infeasible) solution to SNDRC by setting $\tilde{z}_{\tau}(t) = 1$ if $\sum_{(i,j)\in\tau} \sum_{k \in \mathcal{K}} \tilde{x}_{ij}^{k\tau}(t) > 0$. This solution $(\tilde{x}(t), \tilde{z}(t))$ has the objective function value

$$\sum_{\tau \in \theta} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^{k} \tilde{x}_{ij}^{k\tau}(t) + \sum_{\tau \in \theta} F \tilde{z}_{\tau}(t) + \sum_{\tau \in \theta} \sum_{(i,j) \in \mathcal{A}} r_{ij}^{\tau} f_{ij} \tilde{z}_{\tau}(t).$$

Thus, for a solution $\tilde{x}(t)$ of AP(SNDRC($\rho(t)$)), the values $\rho_{ij}^{k\tau}(t+1)$ are calculated to satisfy the relation $\sum_{k \in \mathcal{K}} \sum_{(i,j)\in\mathcal{A}} \tilde{x}_{ij}^{k\tau}(t)(c_{ij}^{k} + \rho_{ij}^{k\tau}(t+1)) = \sum_{k \in \mathcal{K}} \sum_{(i,j)\in\mathcal{A}} \widetilde{x}_{ij}^{k\tau}(t) c_{ij}^{k} + F\tilde{z}_{\tau}(t) + \sum_{(i,j)\in\mathcal{A}} r_{ij}^{\tau} f_{ij} \tilde{z}_{\tau}(t)$, which can be done by setting

$$\rho_{uv}^{k\tau}(t+1) = \begin{cases} \frac{F + \sum_{(i,j)\in\tau} r_{ij}^{\tau} f_{ij}}{\sum_{k \in \mathcal{K}} \sum_{ij \in \tau} \tilde{x}_{ij}^{k\tau}(t)}, & \text{if } \sum_{k \in \mathcal{K}} \sum_{ij \in \tau} \tilde{x}_{ij}^{k\tau}(t) > 0 \\ \rho_{uv}^{k\tau}(t) & , \ otherwise \end{cases}, \ \forall \tau \in \theta,\ k \in \mathcal{K},\ (u,v) \in \tau. \tag{11}$$

If the cycle τ appears in the solution to the linear relaxation of the SNDRC, we set $\rho_{ij}^{k\tau}(0) = 1$. Otherwise, we set $\rho_{ij}^{k\tau}(0) = \left(F + \sum_{(i,j)\in S} r_{ij}^{\tau} f_{ij} \right) \Big/ \sum_{(i,j)\in S} r_{ij}^{\tau} u_{ij}$.

When the solution $(\tilde{x}(t), \tilde{z}(t))$ violates the ***0-1 service constraints*** (constraint set (3)), or the ***resource bound constraints*** (constraint set (4)), our approach next

executes the following procedures to try and construct a feasible solution to SNDRC from $(\tilde{x}(t), \tilde{z}(t))$.

4.3 Creating a feasible solution to SNDRC from a solution to AP(SNDRC)

Because the AP(SNDRC) does not include all the constraints that are present in the SNDRC (constraint sets (3) and (4)), and unlike most slope scaling approaches, rounding procedures are not sufficient to construct a feasible solution to the SNDRC from a solution, $(\tilde{x}(t), \tilde{z}(t))$, to the AP(SNDRC). Instead we must execute another procedure to construct a feasible solution to the SNDRC. Our procedure, instead of modifying the solution to the AP(SNDRC) directly, creates a subgraph, $\bar{\mathcal{G}}$ of $\mathcal{G}$ that contains the nodes $\mathcal{N}$ and a subset, $\bar{\mathcal{A}}$, of the arcs, $\mathcal{A}$, that can be decomposed into cycles. Then, the procedure attempts to extract cycles from this subgraph that can be used to construct a feasible solution to the SNDRC.

We create $\bar{\mathcal{G}}$ by first adding to $\bar{\mathcal{A}}$ all service arcs (i, j) in $\mathcal{A}$ such that either $\tilde{x}(t)^k_{ij} > 0$ for some $k \in \mathcal{K}$, or they belong to a cycle used in the AP(SNDRC) solution. We also add all holding arcs to $\bar{\mathcal{A}}$. Next, if in the solution $(\tilde{x}(t), \tilde{z}(t))$, there are terminals where the resource bound constraint is violated (i.e. $\sum_{\tau \in \theta_l} \tilde{z}_l(t) > ub_l$), we solve an optimization problem to add service arcs that are in $\mathcal{A} \setminus \bar{\mathcal{A}}$ to $\bar{\mathcal{A}}$ that will enable an unused resource at one terminal to move to a terminal where an excess number of resources is used. Finally, we solve another optimization problem to add service arcs in $\mathcal{A} \setminus \bar{\mathcal{A}}$ to $\bar{\mathcal{A}}$ to ensure that $\bar{\mathcal{G}}$ can be decomposed into cycles in such a way that each service arc appears in at most one cycle but holding arcs may appear in multiple cycles, as doing so maximizes the number of cycles that can be extracted from $\bar{\mathcal{G}}$. The objective of both of these optimization problems is to minimize the total cost of the arcs added with respect to the cost coefficient f_{ij}. Lastly, we note that we formulate and solve both of these optimization problems as a minimum cost, maximum flow problem (Ahuja et al., 1994). See Vu et al. (2012) for details of how a similar procedure was done for a network design problem with Eulerian-type constraints.

After creating $\bar{\mathcal{G}}$, we next extract a set of cycles from this network that are guaranteed to satisfy all the constraints of the SNDRC other than the flow conservation constraints. Specifically, to extract cycles from $\bar{\mathcal{G}}$ into the set $\bar{\theta}$ we execute Algorithm 2.

With steps 3 and 8 of Algorithm 2 and the condition of the for loop in step 5, we ensure that at most u_l resources from each terminal l are used. Similarly, step 10 ensures that each service arc in $\bar{\mathcal{G}}$ will appear in at most one cycle in $\bar{\theta}$. As a result, setting $z_\tau = 1 \ \ \forall \tau \in \bar{\theta}$ will not violate either the resource bound (4) or 0-1 service (3) constraints of the SNDRC. To see if the flow conversation constraints can be satisfied using the

Algorithm 2 Cycle extraction procedure

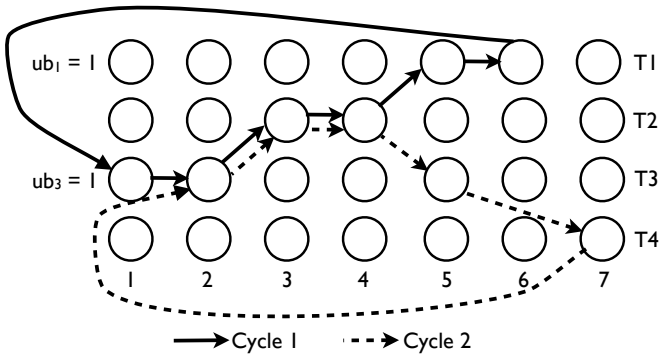
Require: Graph $\bar{\mathcal{G}}$
Require: An ordering $O_{\mathcal{N}}$ of the nodes $\mathcal{N}$
1: Set $\bar{\theta} = \emptyset$
2: **for all** $l \in \mathcal{L}$ **do**
3: Set $\kappa_l = ub_l$
4: **end for**
5: **for all** $l_t \in O_{\mathcal{N}}$ such that $\kappa_l > 0$ **do**
6: Perform depth-first search from l_t to identify a feasible cycle τ in $\bar{\mathcal{G}}$.
7: Add τ to $\bar{\theta}$
8: Set $\kappa_l = \kappa_l - 1$
9: **for all** service arcs $(i, j) \in \tau$ **do**
10: Remove (i, j) from $\bar{\mathcal{A}}$.
11: **end for**
12: **end for**
13: Repeat steps 5 to 12 until no cycles found

cycles in $\bar{\theta}$ we solve a minimum-cost multicommodity network flow problem (Ahuja et al., 1994) with respect to the arc costs c^k_{ij} on the network induced by those cycles. If the flow conservation constraints can be satisfied, we have a new feasible solution. Finally, to generate a diverse set of solutions, we call Algorithm 2 multiple times, each time perturbing the ordering $O_{\mathcal{N}}$ of how nodes are processed and the ordering in which nodes are considered during the execution of the depth-first search.

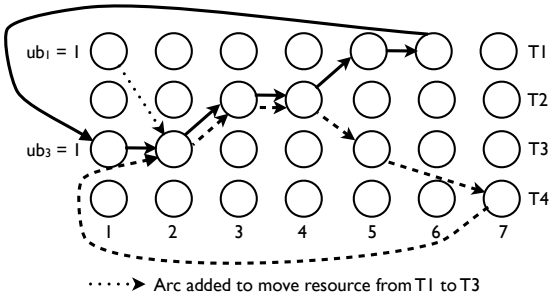
We illustrate this procedure in Figures 5(a) to 5(d). In Figure 5(a) we depict a solution to the AP(SNDRC) that violates both the resource bound constraints (two cycles depart from T3 at period 2 when there is only one located at that terminal) and the 0-1 service constraints (they depart on the service arc from T3 at period 2 to T2 at period 3). Next, we depict in Figure 5(b) the addition of the arc from T1 at period 1 to T3 at period 2 to enable the unused resource at T1 to move to T3 where an excess number of resources are used. Next, in Figure 5(c) we show the arcs that are added to ensure that the subgraph may be decomposed into cycles, and in Figure 5(d) we illustrate the cycles that are extracted from that graph. While we do not illustrate this, the last step executed is to solve a minimum-cost multicommodity network flow problem to determine whether the flow balance constraints can be satisfied using only these two cycles.

4.4 Intensification and Diversification

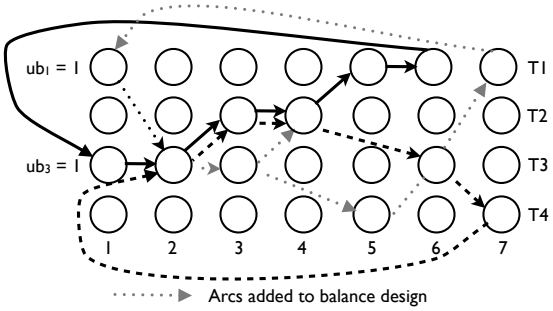
Metaheuristics (Glover and Laguna, 1997) have long included intensification procedures, wherein a region of the solution space is explored deeply, and diversification procedures, wherein the search is directed towards regions of the solution space that have not been



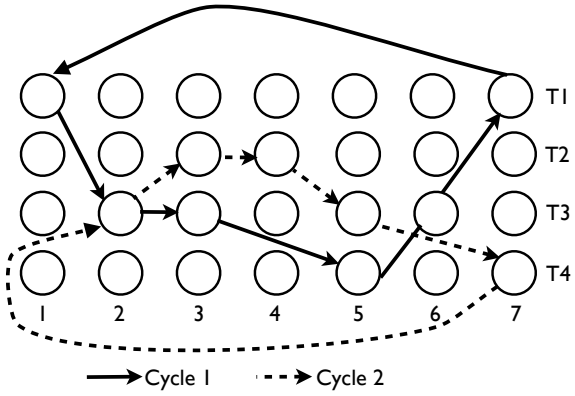
(a) Solution to AP(SNDRC)



(b) Resolve resource bound constraints



(c) Ensure subgraph can decompose into cycles



(d) Extract feasible cycles

Figure 5: Converting a solution to AP(SNDRC) to a solution to SNDRC

thoroughly searched. Because such procedures often enable the search to find high quality solutions in limited times we have included them in our solution approach. We first present our intensification procedure and then the diversification procedure.

For intensification, we introduce an exact optimization step into the search, wherein SNDRC($\tilde{\theta}$) is solved (for what is presumably a set of cycles, $\tilde{\theta}$, of very limited cardinality) with a commercial mixed integer progrmaming solver. See De Franceschi et al. (2006); Archetti et al. (2008); Hewitt et al. (2010); Erera et al. (2012) for other examples of heuristics that find high-quality solutions by solving a restriction of the original problem. To create the set of cycles $\tilde{\theta}$, our intensification procedure first chooses the cycles that appear in the last q solutions to AP(SNDRC). Then, a subgraph of $\mathcal{G}$ is created based on the arcs that appear in those cycles. Because some cycles may share arcs, this graph may not be Eulerian and thus we next add arcs to the subgraph to make it Eulerian with the procedure presented in Vu et al. (2012). Finally, a depth-first search-type approach is performed on this subgraph to extract cycles which are $TMAX$ periods long. These extracted cycles are used to construct the set $\tilde{\theta}$ which is then used to create SNDRC($\tilde{\theta}$). This mixed integer program (MIP) is then solved with the value of the best-known solution as an upper bound on the objective function value and time limit t_{int} (in seconds). The intensification procedure is executed when an improved solution has not been found after a predefined number of iterations.

For diversification, because it is the cycles in the solution to AP(SNDRC) at a given iteration that dictate the structure of the resulting solution to SNDRC, we periodically modify the objective function of AP(SNDRC) to avoid frequently used cycles. To do so, we collect the number of times, fre_τ, each cycle τ appears in a solution to AP(SNDRC). Then, if the diversification condition is met for each cycle τ in the current solution of the approximated problem AP(SNDRC), we set the linearization factor $\rho_{uv}^{k\tau}(t)$ to $\rho_{uv}^{k\tau}(t)(1+\epsilon * fre_\tau)$ where ϵ is an algorithm parameter. We continue to update $\rho(t)$ in this manner for a fixed number of iterations that is dictated by the algorithm parameter I_{max}^{diver}. Lastly, it is possible for two consecutive solutions of AP(SNDRC) to be the same, in which case the slope-scaling procedure will terminate. To continue the execution of slope-scaling when this occurs, we add a penalty value P to the linearization factors as seen in (12). In our experiments, P is set to the objective value of the current solution to AP(SNDRC).

$$\rho_{uv}^{k\tau}(t) = \begin{cases} \frac{P+F+\sum_{(i,j)\in\mathcal{S}} r_{ij}^{\tau} f_{ij}}{\sum_{k\in\mathcal{K}}\sum_{ij\in\tau}\tilde{x}_{ij}^{k\tau}}, & \text{if } \sum_{k\in\mathcal{K}}\sum_{ij\in\tau}\tilde{x}_{ij}^{k\tau} > 0 \\ \rho_{uv}^{k\tau}(t-1) & , \ otherwise \end{cases}, \forall \tau \in \theta,\ k \in \mathcal{K},\ (u,v) \in \tau \quad (12)$$

5 Computational experiments

The purpose of the computational experiments is to determine the effectiveness of the solution approach presented in Section 4 and to understand which of its features contribute to its effectiveness. We performed all experiments on a cluster of computers with 2 Intel Xeon 2.6 GHz processors and 24 GB of RAM that were running Scientific Linux 6.1. We used CPLEX 12.4 to solve linear and mixed integer programs. Initial calibration experiments on a small number of instances yielded the parameter values displayed in In Table 5.

Parameter	Description	Section	Value
λ	Number of initial cycles created for each commodity	4.1	3
I_{max}^{diver}	Number of diversification iterations	4.4	5
ϵ	The effect of frequency on linearization factor during diversification	4.4	1
t_{int}	Time limit for MIP solved during intensification	4.4	600 sec.
q	Number of iterations from which AP(SNDRC) solution cycles are used	4.4	1

Table 1: Algorithm parameter values

We execute the solution approach on three sets of instances. The first is a set of instances that are small enough that all cycles can be enumerated and a commercial MIP solver can quickly solve nearly all of the resulting instances of the SNDRC to optimality. The second is the set of instances used in Andersen et al. (2011) and are based on a case study in rail transportation. In these instances the resource fixed cost (F) is the driving factor in the cost structure, with the cost coefficient $f_{ij} = 0, \quad \forall (i,j) \in \mathcal{A}$. Thus, to see whether the performance of the algorithm is dependent on this cost structure we created a third set of instances that are exactly the same as those in Andersen et al. (2011) except that the cost parameters $f_{ij} > 0$ are set to a value that is proportional to the duration (in periods) of the service.

5.1 Benchmarking the approach

We first compare the performance of the solution approach against near-optimal solutions produced by a commercial MIP solver. We present characteristics of the five instances used for these tests in Table 2.

Instance	Terminals	Services	Periods	Service+Holding arcs	Commodities
1	5	10	15	150+75	20
2	5	15	20	300+100	25
3	5	15	25	375+125	25
4	5	15	15	225+75	100
5	5	15	15	225+75	200

Table 2: Characteristics of "small" instances.

In Table 3 we report the value of the best solution found by the MIP solver when run for 10 hours (Best sol), the value of the best solution found by our solution approach (SSCG), a comparison of the quality of the solutions (Gap) calculated as 100*(SSCG - Best sol)/(SSCG), the cardinality of θ, ($|\theta|$), the number of cycles generated by SSCG (SSCG-Cycles), and a comparison of the number of cycles generated (% of cycles) calculated as 100*(SSCG-Cycles/$|\theta|$). The MIP solver is able to solve nearly all of the instances in one hour, except for instance 3, for which the cardinality of θ led to memory problems. However, our solution approach was very competitive, producing solutions that are on average only 1.27% worse. We also note that our solution approach typically finds this best solution in less than thirty minutes. What is also interesting to note is that while our approach generates very few cycles, it still generates cycles that can yield a high quality solution.

Instance	Best sol	SSCG	Gap	$\|\theta\|$	SSCG-Cycles	% of cycles
1	45,748	45,748	0.00%	5,520	100	1.81%
2	41,656	42,838	2.76%	126,040	238	0.19%
3	n/a	39,796	n/a	1,655,800	275	0.02%
4	162,865	164,210	0.82%	8,985	412	4.59%
5	370,160	375,888	1.52%	8,985	1,242	13.82%

Table 3: Results for "small" instances

We next benchmark the approach on the instances from Andersen et al. (2011). This test set includes 7 classes of 5 randomly generated instances based on a real-life case study in rail transportation planning, albeit with an increased numbers of terminals, services, time periods, and commodities than seen in the study. For these instances the resource fixed cost, F, is much higher than the routing cost, c_{ij}^k, and the service fixed cost, f_{ij} is assumed to be 0. The number of resources at each terminal is a random number in the interval [5,10]. Other characteristics of these instances are given in Table 4.

Instance Class	Terminals	Services	Periods	Service+Holding arcs	Commodities
6	5	15	40	600+200	200
7	5	15	50	750+250	400
8	7	30	30	900+210	200
9	7	30	30	900+210	400
10	7	30	50	1500+350	300
11	10	40	30	1200+300	200
12	10	50	30	1500+350	100

Table 4: Characteristics of rail-based instances

We benchmark against a MIP solver solving SNDRC($\bar{\theta}$) where $\bar{\theta}$ contains the cycles generated by the column generation portion of our approach. We execute the MIP solver for ten hours and record the best solution found at the end of one hour and ten hours. We also execute our solution approach twice, once for one hour and once for ten hours. For both executions of our approach we also terminate after 500 iterations of slope scaling or 100 iterations without finding an improved solution.

In Table 5 we compare the quality of the solutions produced by our solution approach when executed for one and ten hours (rows SSCG 1 Hour and SSCG 10 Hours) with the solutions produced by the MIP solver after one and ten hours (columns MIP 1 Hour and MIP 10 Hours). We report the average gap in solution quality over all 35 instances (column Gap), the number of instances where SSCG produced a better solution (# SSCG better), and the number of instances where the MIP solver could not find a feasible solution (# MIP no solution). Finally, because our approach begins by solving SNDRC with column generation, we have a dual bound on the optimal value of the instance and thus we can report (column Opt. gap) an optimality gap for the solutions produced by our approach. We see in Table 5 that our approach significantly outperforms the MIP solver, including producing in one hour solutions that are 4.5% better than the MIP solver can produce in 10 hours. Looking at the Opt. gap column we conclude that while better quality solutions are found when executing our approach for 10 hours, limiting the method to one hour does not prevent it from finding high quality solutions.

	MIP 1 Hour			MIP 10 Hours			
	Gap	# SSCG better	# MIP no solution	Gap	# SSCG better	# MIP no solution	Opt. gap
SSCG 1 Hour	-16.16%	23	14	-4.49%	5	2	10.31%
SSCG 10 Hours	-20.14%	26	14	-7.72%	11	2	7.79%

Table 5: Results for rail-based instances

Because our approach generates cycles throughout its execution (and after the linear relaxation of SNDRC is solved), we also solved SNDRC($\bar{\theta}$) as a MIP for 10 hours for

each of the 35 instances where $\bar{\theta}$ contains all the cycles generated by our approach. We found that, on average, in 10 hours our approach produced a solution that was 2.70% worse than what the MIP solver could produce. This suggests that the cycles produced by our approach after the column generation procedure are critical for its success and that our approach is capable of producing high-quality solutions given the set of cycles it generates.

We next study in Table 6 the performance of SSCG on the instances from Andersen et al. (2011) where the cost coefficients f_{ij} are positive. We see that while SSCG still performs well compared to the MIP solver it does not do as well (in comparison) as it does on the instances where $f_{ij} = 0$. Comparing the # MIP no solution columns in Tables 5 and 6 suggests that instances with $f_{ij} > 0$ are, in a sense, more difficult.

	MIP 1 Hour			MIP 10 Hours			
	Gap	# SSCG better	# MIP no solution	Gap	# SSCG better	# MIP no solution	Opt. gap
SSCG 1 Hour	-45.32%	8	15	13.65%	0	7	17.11%
SSCG 10 Hours	-58.98%	14	15	4.20%	2	7	9.62%

Table 6: Results for rail-based instances with $f_{ij} > 0$

We next study how sensitive SSCG's performance is to the distribution of resources across the network. We consider three cases: (1) each terminal is assigned the same number of resources (recall that we do not differentiate one resource from another), (2), the number of resources assigned to a terminal is proportional to the total volume of all shipments originating at that terminal, and, (3) the number of resources assigned to a terminal is determined randomly. We note that in each of these cases the instances are more constrained than those reported on in the previous tables as the total number of resources assigned to terminals is much lower. We again compare the performance of SSCG with a MIP solver given the cycles produced by SSCG during its execution. In this case we ran the approaches for 10 hours on the rail-based instances with $f_{ij} = 0$. We report in Table 7 the results of our experiments for each of the three cases, including, for each approach, the (average) optimality gap of the solutions produced (as measured against the dual bound produced by SSCG), and the number of instances wherein it produced a feasible solution.

	Equal distribution		Proportional distribution		Random distribution	
	Opt. gap	# Soln found	Opt. gap	# Soln found	Opt. gap	# Soln found
SSCG	7.26%	32	7.30%	32	7.19%	29
MIP	7.30%	31	7.23%	28	7.58%	23

Table 7: Results for different distributions of resources

We see that the performance of SSCG is robust with respect to the distribution of resources as the optimality gaps and number of instances wherein the approach was able to produce a solution are roughly the same across all three cases. While the performance of SSCG, in terms of the quality of solutions produced, is comparable to the MIP solver, for every case SSCG is able to produce a feasible solution for more instances than the MIP solver.

5.2 Understanding why the approach works

We next study the impact of using the cycles generated by solving the linear relaxation of SNCRC with column generation, initializing the linearization factors based on the solution to the linear relaxation of SNDRC, and using the diversification and intensification procedures.

To study the impact of using the cycles generated with column generation on the quality of the solution produced by the approach we ran the approach on the instances from Andersen et al. (2011) but skipped solving the SNDRC with column generation. Instead we began the approach with the cycles generated through the multiple shortest paths procedure described at the end of Section 4.1. We found that using the cycles generated by column generation enabled the approach to find solutions that were 5.16% better on average.

We next focus on understanding whether setting $\rho_{ij}^{k\tau}(0) = 1$ when τ appears in the solution to the linear relaxation of SNDRC enables the approach to find higher quality solutions. To do so, we ran the approach on the instances from Andersen et al. (2011) but set $\rho_{ij}^{k\tau}(0) = \left(F + \sum_{(i,j)\in S} r_{ij}^{\tau} f_{ij}\right) \Big/ \sum_{(i,j)\in S} r_{ij}^{\tau} u_{ij} \quad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, \tau \in \theta$. We found that initializing the linearization factors based on the solution to the linear relaxation of SNDRC enabled the approach to find solutions that were 3.84% better on average.

Lastly, we focus on understanding the impact of using the diversification and intensification procedures described in Section 4.4. To do so, we ran the algorithm on each of the 35 instances from Andersen et al. (2011) two more times. The first time, neither the diversification or intensification procedures are used. The second time, the diversification procedure is used. Comparing the quality of the solutions produced when the diversification procedure is used with when it is not, we saw that solutions were 3.58% better on average when the diversification procedure is used. Next, comparing the quality of the solutions produced when only the diversification procedure is used with when it and the intensification procedure is used, we saw that solutions were 6.41% better on average when both the diversification and intensification procedures were used.

6 Conclusions and future work

We extended existing service network design models that recognize the need to manage facility-based resources when routing commodities to also recognize that there are limits on the number of resources available at each terminal. Due to the cycle-based nature of the model proposed, there can be a huge number of variables for instances of even modest size, too many to enumerate in a reasonable period of time. Thus, for a solution approach to produce good solutions it must both generate cycles that appear in high-quality solutions and use them in an effective manner. We presented a solution approach that combines techniques from multiple research areas to perform both of those tasks. We next presented an extensive computational study to demonstrate that the method can produce high quality solutions.

We demonstrated the effectiveness of the approach by comparing the quality of the solutions it produced with those produced by a state-of-the art commercial solver. To understand whether the approach is capable of generating "good" cycles, we first considered instances that were small enough that all the cycle-based variables could be enumerated and that a commercial solver was able to (usually) solve to optimality. Our experiments indicated that our approach, while only generating a small fraction of the possible number of cycles, was still able to produce high quality solutions (as measured against the near-optimal solution produced by the commercial solver).

We next benchmarked our approach on a set of realistically-sized instances from the literature. Here, because all variables could not be enumerated, we executed the commercial solver on instances derived from the cycles produced by our approach. We found that our approach produced both high-quality solutions (as measured against the dual bound our approach produces) and those solutions were typically better than those produced by the commercial solver. We conclude from these experiments that our approach is capable of producing the cycles necessary to produce a high quality solution and of using those cycles in an effective manner. Finally, like many solution methods, ours relies on various parameters and design choices. We finished our computational study with experiments that showed how our choices are critical to the success of the approach.

The primary focus of our next work in this area is to extend the model to also allocate resources to facilities. Whereas the model we have studied focuses more on integrating operational-type decision-making into tactical planning formulations, this new model will instead recognize operational realities when making strategic (fleet dimensioning) and tactical decisions. While similar to location routing and location-allocation problems, we believe such a model has not yet been studied and could have many practical uses, such as helping a trucking company determine driver hiring strategies.

Acknowledgments

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant program and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Qubec through their infrastructure grants and of Calcul Qubec and Compute Canada through access to their high-performance computing infrastructure..

References

R. Ahuja, T. Magnanti, and J. Orlin. Network flows: Theory, algorithms, and applications. *Journal of the Operational Research Society*, 45(11):1340–1340, 1994.

J. Andersen and M. Christiansen. Designing new european rail freight services. *Journal of the Operational Research Society*, 60:348–360, 2009.

J. Andersen, T. Crainic, and M. Christiansen. Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, 193(2): 377 – 389, 2009a.

J. Andersen, T. G. Crainic, and M. Christiansen. Service network design with asset management: Formulations and comparative analyses. *Transportation Research Part C: Emerging Technologies*, 17(2):197 – 207, 2009b.

J. Andersen, M. Christiansen, T. G. Crainic, and R. Grønhaug. Branch and price for service network design with asset management constraints. *Transportation Science*, 45:33–49, February 2011.

C. Archetti, M. Speranza, and M. Savelsbergh. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42:22–31, 2008.

C. Barnhart and R. R. Schneur. Air network design for express shipment service. *Operations Research*, 44(6):pp. 852–863, 1996.

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

M. Chouman and T. Crainic. A MIP-Tabu Search Hybrid Framework for Multicommodity Capacitated Fixed-Charge Network Design. Technical Report CIRRELT-2010-31, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et les transports, Université de Montréal, Montréal, QC, Canada, 2010.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. *Transportation. Handbooks in Operations Research and Management Science*, volume 14, chapter Maritime transportation, pages 189–284. North-Holland, Amsterdam, 2007.

J. F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4):380–404, 1998.

T. G. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

T. G. Crainic. *Handbook of Transportation Science*, chapter Long-haul freight transportation, pages 451–516. Kluwer Academic Publishers, 2003.

T. G. Crainic and T. Bektaş. *Logistics Engineering Handbook*, chapter A brief overview of intermodal transportation, pages 1–16. Taylor and Francis Group, Boca Raton, FL, USA, 2008.

T. G. Crainic and K. H. Kim. Intermodal transportation. *Transport, Handbooks in Operations Research and Management Science.*, 14:467–537, 2007.

T. G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438, 1997.

T. G. Crainic, B. Gendron, and G. Hernu. A slope scaling/lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10:525–545, 2004.

R. De Franceschi, M. Fischetti, and P. Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming B*, 105:471–499, 2006.

G. Desaulniers, J. Desrosiers, and M. Solomon. *Column generation*, volume 5. Springer-Verlag New York Inc, 2005.

A. Erera, M. Hewitt, M. Savelsbergh, and Y. Zhang. Improved load plan design through integer programming based local search. *Transportation Science*, to appear, 2012.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

M. Hewitt, G. Nemhauser, and M. Savelsbergh. Combining Exact and Heuristic Approaches for the Capacitated Fixed-Charge Network Flow Problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010.

D. Kim and P. Pardalos. A solution to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24:195–203, 1999.

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

M. F. Lai and H. K. Lo. Ferry service network design: optimal fleet size, routing, and scheduling. *Transportation Research Part A: Policy and Practice*, 38(4):305–328, 2004.

M. B. Pedersen, T. G. Crainic, and O. B. G. Madsen. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science*, 43(2):158–177, 2009.

K. Smilowitz, A. Atamtürk, and C. Daganzo. Deferred item and vehicle routing within integrated networks. *Transportation Research Part E: Logistics and Transportation Review*, 39(4):305–323, 2003.

D.M.. Vu, T.G. Crainic, and M. Toulouse. A Three-Stage Matheuristic for the Capacitated Multi-commodity Fixed-Cost Network Design with Design-Balance Constraints. Technical report, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et les transports, Université de Montréal, Montréal, QC, Canada, 2012.

J. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11): 712–716, 1971.