Models and Tabu Search Meta-Heuristics for Service Network Design with Asset-Balance Requirements

Michael Berliner Pedersen
Teodor Gabriel Crainic
Oli B.G. Madsen

November 2007

CIRRELT-2007-49

Models and Tabu Search Meta-Heuristics for Service Network Design with Asset-Balance Requirements

Michael Berliner Pedersen[1] ,Teodor Gabriel Crainic[2,*], Oli B.G. Madsen[1]

[1] Centre for Traffic and Transport, Technical University of Denmark, Anker Engelundsvej 1, Building 101 A, 2800 kgs., Lyngby, Denmark

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 and NSERC Industrial Research Chair in Logistics Management, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8

Abstract. This paper focuses on a generic model for service network design, which includes asset positioning and utilization through constraints on asset availability at terminals. We denote these relations as "design-balance constraints" and focus on the design-balanced capacitated multi-commodity network design model, a generalization of the capacitated multi-commodity network design model generally used in service network design applications. Both arc and cycle-based formulations for the new model are presented. The paper also proposes a Tabu Search meta-heuristic framework for the arc-based formulation. Results on a wide range of network design problem instances from the literature indicate the proposed method behaves very well in terms of computational efficiency and solution quality.

Keywords. Service network design, asset balance constraints, freight consolidation carriers, tabu search.

Acknowledgements. This work was supported by the Virtual Centre for Logistics and Freight Transport (CLG) in Denmark, the Danish Technical Research Council, and Atkins Denmark. The Natural Sciences and Engineering Research Council of Canada (NSERC) offered additional support through its Discovery Grant program.

---

* Corresponding author: Teodor-Gabriel.Crainic@cirrelt.ca

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

Introduction

Network design formulations are used to model a wide range of applications in planning transportation, logistics, telecommunication, and production systems. In these applications, multiple commodities, goods, data, people, etc., must be routed between different points of origin and destination over a network with possibly limited capacity. Moreover, other than the routing cost proportional to the number of units of each commodity transported over a network link, a fixed cost must be paid the first time the link is used, representing its construction (opening), improvement, or set up cost. The network design problem then consists in finding a minimum cost design, i.e., a selection of arcs in the network to enable the flow of commodities while minimizing the total network cost: the sum of the fixed cost of including the arcs in the final design and the variable cost of routing the commodities. Presentation of different network design models and their applications can be found, for example, in Minoux (1986), Magnanti and Wong (1984), Ahuja *et al.* (1995), Balakrishnan, Magnanti, and Mirchandani. (1997), and Crainic (2000).

Service network design models generally address issues related to the planning, selection and, eventually, scheduling, of services in consolidation-based transportation systems. In such formulations, network links represent services (or parts thereof) that could be included into the transportation plan operated by the carrier. Link fixed costs then represent the cost of operating the corresponding services, while variable costs stand for the cost or moving commodities using the selected services. All these costs may include measures related to vehicle and load handling in terminals and en route from one terminal to another, delays, crews, and so on. The planning scope is generally on a tactical, mid-term horizon, level, as opposed to the strategic, long-term scope of network design models. Several efforts have been directed toward the formulation of service network design models and surveys are proposed by Christiansen, Fagerholt, and Ronen (2004) and Christiansen *et al.* (2007) for maritime transportation, Cordeau, Toth, and Vigo (1998) for rail transportation, Crainic and Laporte (1997) and Crainic (2003) for land-based long-haul transportation, and Crainic and Kim (2007) for intermodal transportation.

Service network design models usually account for issues related to various service types and characteristics, infrastructure (terminals, mostly) and service capacities and congestion, load consolidation, vehicle classification and, eventually, consolidation into convoys (e.g., multi-trailers operated by motor carriers and the blocks and trains characteristic of rail transportation), broad policies regarding the repositioning of assets (power, vehicles, crews). It is generally implicitly assumed, however, that assets – crews, power units, and vehicles, particularly – are available when needed and, consequently, their management, assignment and repositioning, is left to be dealt with at the operational level of planning on an already decided service network schedule. In fact, with the exception of Smilowitz, Atamtürk, and Daganzo (2003) who consider a fleet of trucks in the context of express-courier services, the circulation of vehicles is considered only when dealing with airplanes or ships (e.g., Armacost, Barnhart, and Ware, 2002, for express-courier services with a heavy air component) where the high vehicle acquisition and operation costs focuses the problem on the routing of the fleet. Moreover, all such contributions we are aware of propose ad-hoc solution approaches taking advantage of the structure of the particular problem considered.

With the increasing pressure on carriers to decrease costs and improve service quality, combined to the growth in demand for intermodal transportation, addressing asset management issues jointly with the design of the service network and schedule increasingly appears as the most appropriate approach. This trend is readily apparent in rail transportation, but may also be observed in some motor-carrier operations. With respect to railways and Intermodal transportation, for example, North-American railways have created intermodal subdivisions which operate so-called “land-bridges” providing efficient container transportation by long, double-stack trains between the East and the West coasts and between these ports and the industrial core of the continent (so-called “mini” land-bridges). New container and trailer-dedicated shuttle-train networks are being created within the European Community with the goal of offering high-quality services and, thus, contribute to increase the part of traffic using environmental friendlier rail transportation (Bektaş and Crainic, 2007).

Several such services implement, or contemplate to implement, some form of full-asset-utilization operation policy, which generally corresponds to operating regular and cyclically-scheduled services with fixed composition. In other words, given a specific frequency (daily or every *x* days), each service occurrence operates a train of the same capacity (length, number of cars, tonnage) and the same number and definition, i.e., origin, destination, and length, of blocks (groups of cars traveling together as a unit from the origin to the destination of the block; blocks result from classification operations at yards). Assets, engines, rail cars and, even, crews, assigned to a system based on full-asset-utilization operation policies can then “turn” continuously following circular routes and schedules (e.g., Andersen and Christiansen, 2006 and Petersen and Crainic, 2007). Little effort has been dedicated, however, to designing service network schedules whilst simultaneously considering vehicle positioning and balancing. This paper aims to contribute to fill this gap.

The contribution of this paper is twofold. First, it presents a generic model for service network design that includes asset positioning and utilization. This is addressed through constraints on the requirement of asset availability in order to perform services. We denote these relations as “design-balance constraints” and focus on the design-balanced capacitated multi-commodity network design (DBCMND) model, a generalization of the capacitated multi-commodity network design (CMND) model generally used in service network design applications. Both arc and cycle-based formulations for the DBCMND are presented. Second, the paper proposes a Tabu Search meta-heuristic framework to address the arc-based formulation of the DBCMND. This contribution is one of, if not the first solution method dedicated to the generic network design model with design-balance constraints. The algorithm is applied on network design instances from the literature and very good computational results are presented.

The paper is structured as follows. Section 1 is dedicated to the problem statement and modeling, while Section 2 presents the DBCMND formulations. Section 3 details the Tabu Search meta-heuristics. Experimental results are presented and analyzed in Section 4 and we conclude in Section 5.

1. Problem Statement and Notation

The DBCMND formulations and the Tabu Search meta-heuristic introduced in the next sections are general and may be applied to any network. To enhance the presentation, however, we first introduce the problem in the context of the scheduled service network design problem with asset-management considerations, which can be stated as follows: 1) select services and their schedules such that all commodities can be transported from their origin to their destination within their specified time windows, and 2) route assets to ensure their availability at terminals to perform the selected services. Total cost is to be minimized.

A single asset type is considered in this paper. It is also assumed that only one unit of asset is used to perform a service. Assets may be crews, trucks, tractors, ships, engines, cars, etc. The single-asset hypothesis is reasonable when services are indeed performed by "single" vehicles, e.g., ships or rail intermodal convoys in European shuttle networks, or when the management of a given asset presents more challenges and constraints to building the transportation plan than the others. Multiple asset management considerations and detailed operational constraints (e.g., maintenance periods) are beyond the scope of this paper and are left to future research.

Service network design problems are defined on networks made up of nodes and links, where nodes stand for terminals, while links represent possible services providing transportation between these terminals. Scheduled problem variants are addressed through time-space diagrams capturing the dynamics of the system over a given planning horizon (e.g., a week) discretized into a number of periods (e.g., a day). Nodes then represent the terminals at all time periods. We denote $\mathcal{N}$ the set of nodes. Figure 1 shows the space-time diagram for a three-terminal system and a seven-period planning horizon.

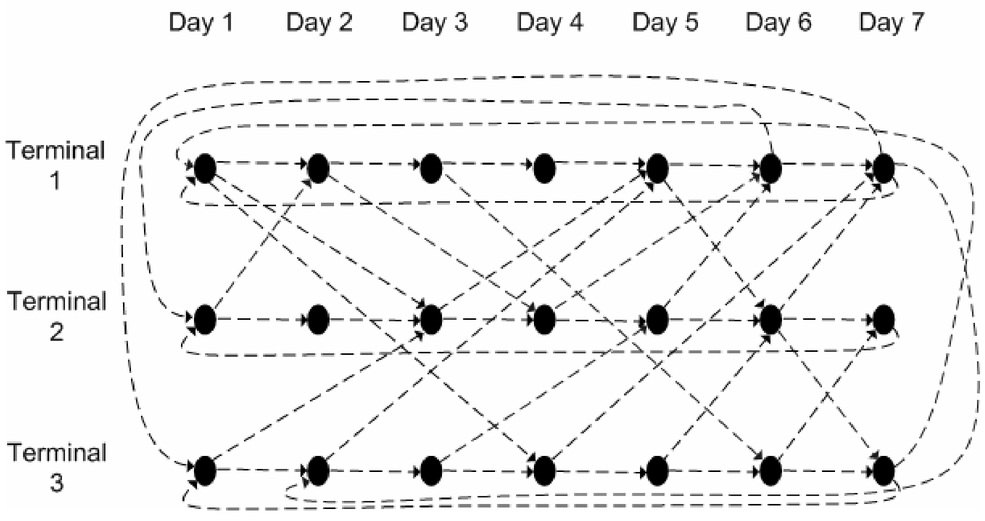

Figure 1.Time-Space Diagram for Cyclic Service Schedule

Let $\mathcal{A}$ represent the arcs of the time-space diagram. Most arcs in $\mathcal{A}$ represent the scheduled services (departures) that may be included in the service network. Assuming non-stop services that depart from terminals at the end of time periods and arrive at the beginning of time periods, we represent a service as an arc between two terminal nodes $(i, j), i, j \in \mathcal{N}$. Figure 1 illustrates a number of potential services, e.g., departing from the node representing terminal 3 at the end of day 1 and arriving at the node representing terminal 2 at the beginning of day 3. Horizontal lines between nodes are the so-called holding arcs representing the choice of keeping vehicles and loads at terminals for one time period. Waiting at terminals incurs "variable" costs (e.g., depreciation values) but fixed "selection" costs are generally not considered.

Note that, it is assumed the designed service network is repetitive for a given time horizon (e.g., a season) for which the current resource and demand conditions of the carrier hold. Schedule repetition is quite common for many types of transportation service networks: public transit, airlines, many intermodal train operations, intercontinental liner shipping, etc. Arcs going "backwards" in time represent schedule repetitiveness. Such arcs stand for services in operation at the transition from one schedule repetition to the next. Thus, in Figure 1, the arc going from terminal 1 at the end of day 6 to terminal 2 at the beginning of day 1 is a "backward" arc representing a service leaving on day 6 of the current week and arriving on day 2 of the following week.

Demand for transportation is given for a set of commodities $\mathcal{P}$, which represent possibly different products that need moving at particular times between particular origin and destination terminals. Each commodity $p \in \mathcal{P}$ thus requires that a certain quantity w_p be moved from origin node *o(p)* to destination node *s(p)*, which capture both the geographic and temporal attributes of the corresponding demand. Unit costs c_{ij}^{p} associated to each arc *(i,j)* and commodity *p* represent the costs incurred moving commodities on services or having then wait at terminals. The "fixed" cost of selecting and operating service $(i,j) \in \mathcal{A}$ is denoted f_{ij}, while the service capacity is u_{ij}.

Two sets of decision variables are normally associated with design formulations and are appropriate in this setting as well. A selection decision variable y_{ij} is associated to each service, that is $y_{ij} = 1$, if arc *(i,j)* is opened (service is selected), and 0 otherwise. Continuous variables x_{ij}^{p}, $(i, j) \in \mathcal{A}, p \in \mathcal{P}$, represent the flow distribution within the service network and stand for the volume of commodity *p* moved by service *(i,j)*.

The scope of design-balanced formulations is to address the asset availability requirements at terminals. The movements of the selected services result in a number of vehicles leaving and arriving at terminal nodes. Within the hypotheses of a full-asset-utilization policy, the number of vehicles available for service at any given period and terminal, just arrived or in waiting at the terminal, must be in sufficient numbers for the planned services to be able to leave the terminal during the period. This translates in the requirement of a vehicle balance at terminal nodes, i.e., the number of services destined to a node must equal the number of services emanating from the same node. (To follow the basic idea of full-asset-utilization policies and without loss of generality, we do not consider safety stocks.) Adding balance requirements for vehicles at each node addresses the vehicle availability and positioning implicitly and yields the DBCMND formulations presented in the next section.

2. Design-Balanced Service Network Design Models

Let X and Y represent the vectors of flow (x_{ij}^p, $(i,j) \in \mathcal{A}, p \in \mathcal{P}$) and design ($y_{ij}$, $(i,j) \in \mathcal{A}$) variables, respectively. Define $\mathcal{N}^+(i) : \{j \in \mathcal{N}^+ \mid (i,j) \in \mathcal{A}\}$ and $\mathcal{N}^-(i) : \{j \in \mathcal{N}^- \mid (j,i) \in \mathcal{A}\}$, the outward and inward neighbors of node $i \in \mathcal{N}$, respectively. For each commodity p and node i, set

$$d_i^p = \begin{cases} w_p & \text{if } i = o(p), \\ -w_p & \text{if } i = d(p), \\ 0 & \text{otherwise.} \end{cases}$$

The arc-based formulation of the design-balanced scheduled service network design problem can then be written as follows:

$$\textit{Minimize } z(X,Y) = \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \quad (1)$$

$$s.t. \sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p, \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \quad (2)$$

$$\sum_{j\in\mathcal{N}^-(i)} y_{ji} - \sum_{j\in\mathcal{N}^+(i)} y_{ij} = 0 \quad \forall i \in \mathcal{N}, \quad (3)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij} y_{ij}, \quad \forall (i,j) \in \mathcal{A}, \quad (4)$$

$$x_{ij}^p \ge 0, \quad \forall (i,j) \in \mathcal{A}, \quad \forall p \in \mathcal{P}. \quad (5)$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A} \quad (6)$$

The objective function (1) accounts for the total system cost, the fixed cost of selected arcs plus the cost of routing the commodity demand, and aims to select the minimum cost design. Equations (2) are the commodity flow conservation relations. Equations (3) are the design-balance constraints, which state that the total number of open design arcs terminating into any node must equal the number of open design arcs going out of that node. The linking (forcing) constraints (4) state that the total commodity flow assigned to an arc cannot exceed the arc capacity if the arc is opened ($y_{ij} = 1$) and must be 0 if the arc is closed ($y_{ij} = 0$). Relations (5) and (6) are non-negativity and integrality constraints for the decision variables.

Notice that removing constraints (3) from the formulation yields the general arc-based capacitated multi-commodity network design model (CMND) formulation. We thus dub the model (1) – (6) the arc-based *design-balanced capacitated multi-commodity network design* formulation (a-DBCMND). Further notice that, as for the CMND formulation, the capacitated multi-commodity minimum cost flow problem (CMCF) is obtained for a given design-balanced feasible design vector $\overline{Y}$:

$$Minimize\ z(x(\bar{Y})) = \sum_{p \in \mathcal{P}} \sum_{(i,j) \in \mathcal{A}(\bar{Y})} c_{ij}^p x_{ij}^p \qquad (7)$$

$$\text{s.t.} \sum_{j \in \mathcal{N}^+(i)} x_{ij}^p - \sum_{j \in \mathcal{N}^-(i)} x_{ji}^p = d_i^p \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \qquad (8)$$

$$\sum_{p \in \mathcal{P}} x_{ij}^p \le u_{ij} \quad \forall (i,j) \in \mathcal{A}(\bar{Y}), \qquad (9)$$

$$x_{ij}^p \ge 0 \quad \forall (i,j) \in \mathcal{A}(\bar{Y}), \forall p \in \mathcal{P}, \qquad (10)$$

where $\mathcal{A}(\bar{Y}) \subseteq \mathcal{A}$ stands for the set of service arcs included in the design $\bar{Y}$, while $x_{ij}^p = 0$, for all arcs $(i,j) \notin \mathcal{A}(\bar{Y})$.

The design-balance constraints (3) strongly link arc choices. To illustrate, assume arc (Terminal 3, day 2 to Terminal 1, day 5) in Figure 1 is selected. Both nodes connected by the selected service are then "out of balance" with respect to constraints (3), which are not obeyed (Figure 2). The balance at these nodes can be restored only by selecting additional services in a cascading effect as shown in Figure 3. Notice that, the four selected arcs form a cycle. Recalling that assets (vehicles) are associated to each opened arc (service), a cycle represents a vehicle rotation over the planning horizon of the schedule. We use this insight to reformulate the model.

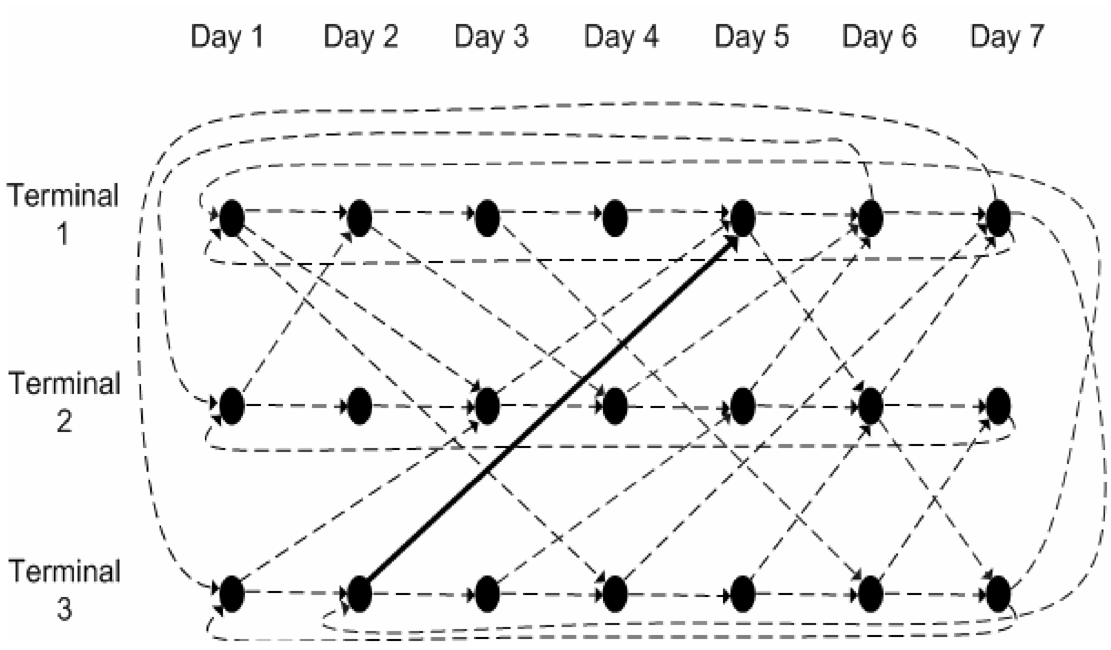

Figure 2. The Service (Terminal3, Day3) to (Terminal 1, Day 5)

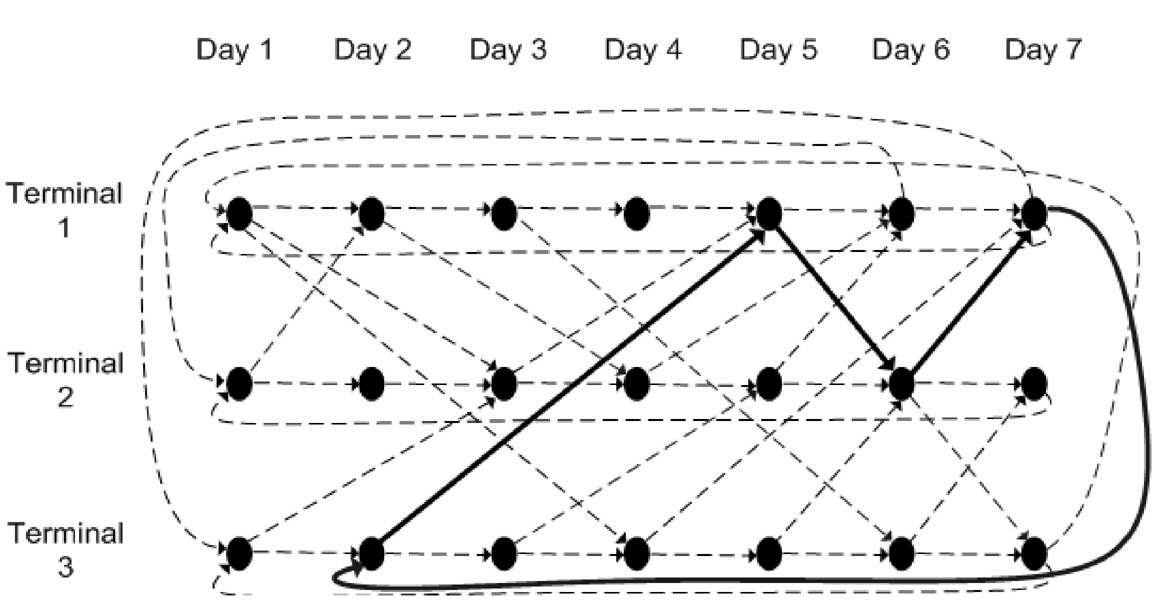

Figure 3. A Feasible Vehicle Rotation

Define the set of possible design cycles $\mathcal{K}$ given the set of design and holding arcs, and set $a_{ij}^{k}, (i,j) \in \mathcal{A}, k \in \mathcal{K}$, to 1 if service arc *(i,j)* is in cycle *k* and to 0, otherwise. Define cycle-design variables $\eta_k, k \in \mathcal{K}$, equal to 1 if cycle *k* is selected (and 0, otherwise). We may then reformulate the arc model to obtain the cycle-based formulation, c-DBCMND:

$$Minimize\ z(X,Y) = \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} f_{ij} a_{ij}^{k} \eta^{k} + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^{p} x_{ij}^{p} \quad (11)$$

$$\text{s.t.} \sum_{j\in\mathcal{N}^{+}(i)} x_{ij}^{p} - \sum_{j\in\mathcal{N}^{-}(i)} x_{ji}^{p} = d_{i}^{p} \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \quad (12)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^{p} \le u_{ij} a_{ij}^{k} \eta^{k} \quad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, \quad (13)$$

$$\sum_{k\in\mathcal{K}} \eta^{k} a_{ij}^{k} \le 1 \quad \forall (i,j) \in \mathcal{A}, \quad (14)$$

$$x_{ij}^{p} \ge 0 \quad \forall (i,j) \in \mathcal{A}, \forall p \in \mathcal{P}, \quad (15)$$

$$\eta^{k} \in \{0,1\} \quad \forall k \in \mathcal{K}. \quad (16)$$

The advantage of the c-DBCMND formulation compared to a-DBCMND is that the design-balance constraints (3) are taken care of implicitly through the cycle-design variable definition and constraints (14), which are set-packing constraints stating that an arc may be included in one selected cycle only. The disadvantage of c-DBCMND is the size of the set of cycles $\mathcal{K}$. It is computationally infeasible to enumerate all cycle-design variables and linking constraints (13) for large scale instances. Dynamic cycle (column) generation procedures will most certainly be required for any exact or meta-heuristic procedure addressing the c-DBCMND. We do not address c-DBCMND in this paper but have introduced it for a more complete description of the model, and the insight cycle variables offer into feasible solution structures.

Un-capacitated fixed-charge network design problems are difficult as they belong to the class of NP-hard problems (Magnanti and Wong, 1986). Capacitated problems, represented by the CMND model, are even harder (Balakrishnan, Magnanti, and Mirchandani, 1997) due to, among other factors, the competition of commodities for the network capacity, the degeneracy of the minimum cost network flow sub-problem, and the difficulty of representing trade-offs between the fixed costs and capacities of design arcs. Adding the node balance constraints adds further complexity to the problem because of the additional interdependency among arc choices mentioned above. Consider, for example, that one obtains the CMND formulation by relaxing the design-balance constraints (3) in the a-DBCMND model. Even finding feasible solutions is far from trivial. In the case of the CMND, a (not necessarily good) feasible solution may be easily found by setting all binary design variables to one (opening all arcs) and solving the CMCF problem. This is not the case for the DBCMND problem, where the design-balance constraints prevent the trivial "all arcs open" solution to be feasible, except for special instances. We therefore assume that DBCMND belongs to the class of NP-hard problems. Then, because tailored heuristics appear to perform best for large-scale CMND problem instances (see the survey papers indicated in the Introduction and the references therein), we introduce a tabu search meta-heuristic as an initial attempt at finding good feasible solutions for the arc formulation of the DBCMND problem.

3. Tabu Search Meta-heuristic

Tabu Search (TS) methods are neighborhood-based meta-heuristics (Glover, 1986; Glover and Laguna, 1997). Thus, in its basic form, TS iteratively moves from the current, *incumbent*, solution to a new one selected from a *neighborhood* of the incumbent. The neighborhood of a solution is a set of other solutions that can be obtained from it by applying one or a few simple changes. The transition from the incumbent solution to a neighbor solution is called a *neighborhood move*. The best solution of the neighborhood is selected as the new incumbent, non-improving moves being accepted to attempt to escape local minima. Learning processes build memories relative to attributes of visited solutions to guide the search. In particular, *tabu lists* are introduced to avoid cycling. A tabu list is a short-term memory mechanism, which stores attribute values identifying the moves that produced recent solutions. When examining an incumbent's neighborhood, moves that have attributes equal to those stored in the tabu list are discarded. TS meta-heuristics may be divided into *phases*, each with its own distinctive neighborhoods.

The TS algorithm we propose addresses the arc formulation and searches through the space of the design vectors. It is composed of two local-search phases. The first, denoted *exploration phase* searches for potentially "good" solutions through relatively simple moves between possibly infeasible solutions. The neighborhood is based on add and drop moves of individual arcs. A penalty is added to each infeasible solution based on an estimation of how far from feasibility it is. The purpose of the penalty is to control infeasibility and induce the search to aim for a trade off between cost and infeasibility. The goal of the second phase, denoted *feasibility phase*, is to reach good feasible solutions starting from the results of the first phase. The neighborhood moves in the second phase are based on an add/drop procedure of paths of design arcs.

These neighborhoods may be very large and an exhaustive search can be very computationally demanding. *Candidate lists* of potentially "good" neighbors are therefore determined in each of the two phases of the TS procedure to decrease the computational requirements of selecting the next move. A candidate list contains a sub-set of neighbor solutions that are deemed worth investigating, i.e., neighbors that one believes may lead to improved solutions, according to one or several criteria.

The following subsections present the various elements of the algorithm, whose overall layout with section pointers can be found in Figure 4.

Procedure phase / stage	**Section**
Initialization • Solve the LP relaxation of the a-DBCMND formulation • Round up design variables → a possibly infeasible starting solution	3.1
Phase I: Exploration	3.2 – 3.5
• Set up the candidate list of non tabu neighbors, i.e., arcs to drop from or to add to the incumbent	3.4, 3.5
• Evaluate approximate solution value for each neighbor	3.2, 3.3
• Select the neighbor with the minimum estimated value	
• Move to this neighbor, which becomes the incumbent o Update the design vector & solve the corresponding CMCF o Update the tabu list o Update the residual capacity and the penalty lists	3.4, 3.5
• Verify stopping condition; Go to Phase II if the value of the overall best solution did not improve sufficiently in the last iterations	3.1
Phase II: Feasibility	3.6 – 3.8
• Set up the candidate list of four moves: paths to open or close	3.6, 3.7
• Evaluate each move by solving the corresponding CMCF • Select the move yielding the lost total system cost	3.7
• If no feasible solution is found from the candidates, return to Exploration, Phase I, otherwise update the incumbent	3.8
Global update: Keep track of best overall solution	
Stop: On reaching time limit	3.1

Figure 4. Overall Tabu Search Algorithmic Structure

3.1 Initialization and stopping criteria

To obtain an initial solution, we first solve r-DBCMND, the linear relaxation of the a-DBCMND formulation, given by relations (17) – (22). If all design variables are integer in the LP solution, we have a feasible initial solution. Otherwise, a possibly infeasible initial solution is built by rounding up the fractional values for the $y_{ij}^0 > 0$ design variables. Notice that, since the exploration phase allows moves between infeasible solutions, the initial solution needs not be feasible.

$$Minimize\, z(X, Y^0) = \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij}^0 + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \quad (17)$$

$$\text{s.t.} \sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p, \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \quad (18)$$

$$\sum_{j\in\mathcal{N}^-(i)} y_{ji}^0 - \sum_{j\in\mathcal{N}^+(i)} y_{ij}^0 = 0, \quad \forall i \in \mathcal{N}, \quad (19)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij} y_{ij}, \quad \forall (i,j) \in \mathcal{A}, \quad (20)$$

$$x_{ij}^p \ge 0, \quad \forall (i,j) \in \mathcal{A}, \forall p \in \mathcal{P}, \quad (21)$$

$$y_{ij}^0 \in [0,1], \quad \forall (i,j) \in \mathcal{A}. \quad (22)$$

The exploration phase stops and the algorithm proceeds to the feasibility phase when the best overall solution value did not improve by at least l_g% over the last l_i iterations. We denote these parameters the *improvement gap, l_g,,* and the *improvement range, l_i*, respectively.

The Tabu Search meta-heuristic could be stopped based of various criteria, e.g., after a number of successive iterations without notable improvements. To facilitate the comparative analysis of the proposed TS algorithm relative to a MIP solver, we stop the meta-heuristic after a predefined amount of time, $\tau_{\max}$.

3.2 Neighborhood structure for the exploration phase

The add/drop idea is an intuitive scheme to define moves, neighbors being obtained by either adding or dropping arcs from the design vector of the incumbent. This approach was adopted, for example, in Powell (1986), Koskosidis, Powell, and Solomon (1992), and Crainic *et al.* (1993). The add/drop procedure has proven to be less effective in addressing the general CMND problem, however, path (Crainic, Gendreau, and Farvolden, 2000) and cycle-based procedures (Ghamlouche, Crainic, and Gendreau, 2003) having offered superior performances. This behavior is due to the fact that in a multi-commodity network, there are generally several paths available for each commodity. Then, opening or closing a single link may prove of little consequence, since one may often reroute traffic and obtain an almost equivalent solution.

We believe, however, that the design-balance constraints of the DBCMND model induce strong connections between design decision variables and thus increase the impact of arc add/drop moves. Recall that selected design arcs in a feasible solution form cycles.

Then, adding/dropping an arc to/from a cycle may restore feasibility with respect to the design-balance constraints at the two extreme nodes of the arc or, alternatively, may destroy it. Thus, for example, adding arc *(i,j)* to the network in Figure 5 would result in nodes *i* and *j* becoming imbalanced, while dropping any black arc would also make the solution infeasible. We therefore base the first-phase neighborhood on arc add/drop moves enhanced by ideas from Ghamlouche, Crainic, and Gendreau (2003) resulting in possibly several arc openings along a given path. Each move is defined by an arc being added or dropped and, thus, the total number of potential neighbor solutions is equal to the cardinality of the set of arcs, $|\mathcal{A}|$ (each arc can be closed if open, and opened if closed).

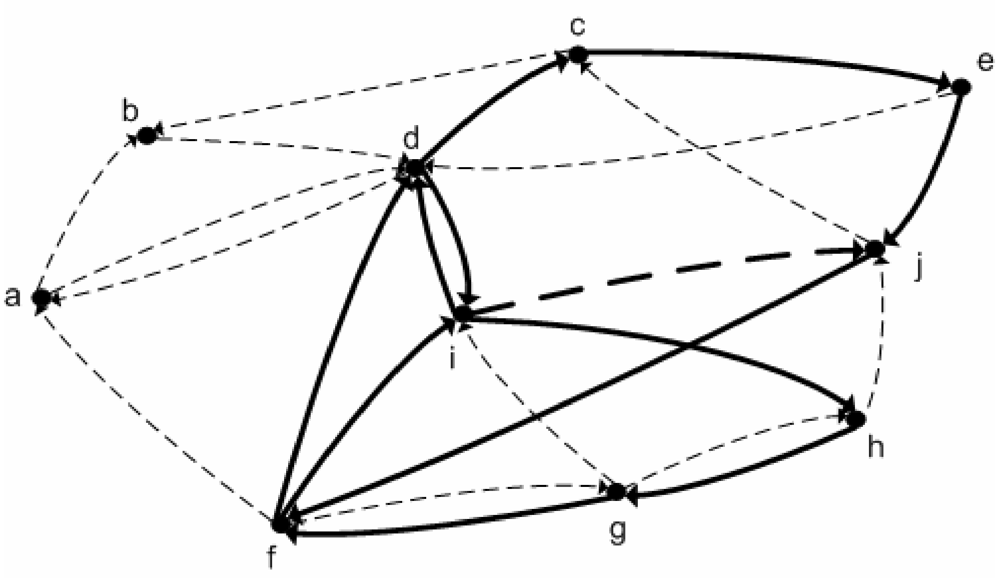

Figure 5. Illustration of Add/Drop Ideas

Using the add/drop procedure presented in Section 3.3 yields feasibility with respect to commodity flows, but does not guarantee feasibility with respect to the design-balance constraints. We therefore implement an infeasibility-monitoring scheme to guide the search toward feasible solutions or toward solutions close to feasibility. Let the *penalty value, P,* represent an estimate of how far the solution is from feasibility with respect to the design-balance constraints. The penalty value is then added to the total system cost, Z, to yield the *solution value, V = Z + P,* according to which neighboring solutions are compared. By using this measure, a solution with a relatively high total system cost that is close to feasibility may be preferred to one with a lower total system cost that is further from feasibility.

The penalty value is obtained using two parameters, the *total system imbalance* and the *maximum absolute imbalance*. Both parameters are obtained from the *node imbalances* in the network, computed as the difference between the number of open outgoing and incoming arcs at each node *i*, $\psi^i = \sum_{j \in \mathcal{N}^-(i)} y_{ji} - \sum_{j \in \mathcal{N}^+(i)} y_{ij}$. The node imbalance is thus negative when there are more outgoing than incoming arcs and vice versa. Figure 6 shows two nodes, one with negative imbalance, the other with positive node imbalance.

The *total system imbalance*, $\psi^{\mathcal{N}}$, is calculated as the sum over all absolute node imbalances, $\psi^{\mathcal{N}} = \sum_i \left|\psi^i\right|$, and gives an indication of the infeasibility of the system by representing the total number of imbalances that need to be fixed at all the nodes. The *maximum absolute imbalance*, $\psi^{\max}$, is the largest absolute value of all node imbalances, $\psi^{\max} = \max\left(\left|\psi^i\right|\right)$, and points to the difficulty of achieving feasibility relative to the node with the highest imbalance.

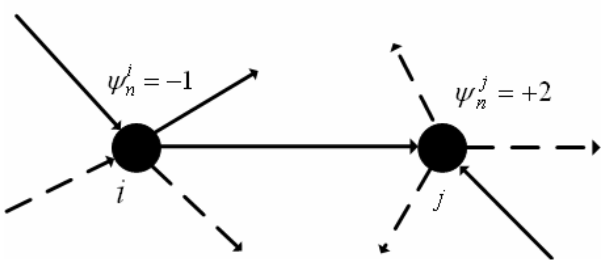

Figure 6. Illustration of Node Imbalance

Using the analogy of scheduled transportation networks, the node imbalance at a given node *i* can be interpreted as having to add $|\psi^i|$ empty vehicle movements in or out of that node. The cost of adding an "empty-vehicle" move, $\tilde{f}$, is here approximated as the product of the average of the fixed costs of the arcs in the network, $\bar{f}$, and an empirical scaling parameter, τ, used to control the importance of the penalty value in evaluating neighbor solutions. Then, in order to induce a penalty that increases "non-linearly" with the infeasibility of the solution, we multiply the estimated cost of an "empty-vehicle move" (arc), the total system imbalance, and the maximum node imbalance. The penalty value is thus computed as $P = \tilde{f}\psi^{\mathcal{N}}\psi^{\max}$, where $\tilde{f} = \bar{f}\tau$.

3.3 Moves for the exploration phase

Arc-add/drop moves change the design vector of the incumbent resulting in a new design vector and flow distribution. To evaluate such a move, one must evaluate the change in design and flow-related costs and account for the differences in the infeasibility measures (if any). For drop moves, one must also verify whether closing the arc would result in connectivity loss for some commodities and, thus, in an infeasible flow distribution. Solving the capacitated multi-commodity minimum cost flow (CMCF) problem for the new design vector would provide these results, but at a heavy computational cost if performed for each potential neighbor. We therefore propose fast heuristics described in the following two sub-sections for closing and opening an arc, respectively.

3.3.1 Closing arcs

Closing arc *(i,j)* with positive flow in the incumbent solution implies the corresponding flow distribution is no longer feasible because the paths carrying the commodities currently using arc *(i,j)*, $\mathcal{P}_{ij}$:$\{p \in \mathcal{P} \mid x_{ij}^p > 0\}$, are disconnected. Figure 7 illustrates a network with flow of two commodities (light from node *d* to node *a,* and dark grey from node *a* to node *d*) and arc attributes (fixed cost, variable cost, capacity, total flow). Closing arc *(i,j)* results in the dark-grey commodity no longer having a feasible flow path.

To maintain flow feasibility, the commodities on arc *(i,j)* must be redirected onto other paths. These paths could use currently open arcs only, or could also use currently-closed arcs that would be open following the move. It is this latter idea, inspired by the work of Ghamlouche, Crainic, and Gendreau (2003), which we develop.

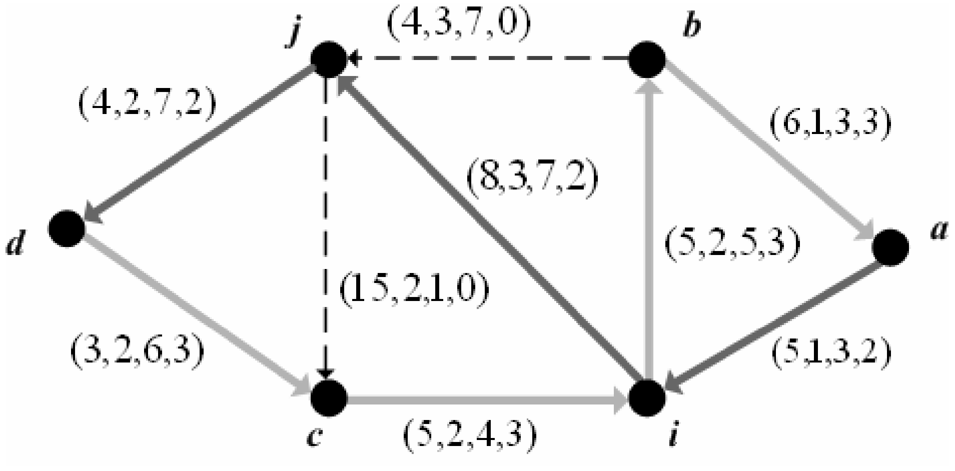

Figure 7. 2-Commodity Network

The redirection of the commodities may be done individually, for each commodity, or as an aggregated entity resulting in a smaller computational effort. We favor the first alternative, because redirecting commodities individually allows for a more efficient use of the residual capacity of open arcs, $r_{ij} = u_{ij} - \sum_{p \in \mathcal{P}} x_{ij}^p$, as the flow volumes are smaller.

Several procedures may be used to redirect the flow of a given commodity, each requiring the definition of a particular residual graph. One could, for example, solve the minimum cost flow problem for the particular commodity on a graph where the flows of the other commodities are fixed and closed arcs are eligible to pass flow. Such an approach would be computationally heavy, however. Then, to limit the computational burden, we choose to redirect flow on a single path only, namely the "cheapest" path on a suitably defined residual graph. If such a graph cannot be found, the move is declared infeasible.

The redirection is thus performed by constructing a Γ^p-residual graph, $\mathcal{G}_p^\Gamma = (\mathcal{N}, \mathcal{A}_p^\Gamma)$, where $\Gamma^p = w^p$ is the total volume of commodity $p \in \mathcal{P}$. All open arcs, except arc *(i,j)*, are included in the residual graph if their residual capacity, r_{kl}, is larger than Γ^p,

$$\mathcal{A}^{\Gamma,open} = \{(k,l) \in \mathcal{A} \mid y_{kl} = 1 \wedge r_{kl} = u_{kl} - \sum_{p \notin \mathcal{P}_{ij}} x_{kl}^p \geq \Gamma^p\},$$

while closed arcs are included in the residual graph if their capacity u_{ij} is larger than Γ^p:

$$\mathcal{A}^{\Gamma,closed} = \{(k,l) \in \mathcal{A} \mid y_{kl} = 0 \wedge u_{kl} \geq \Gamma^p\}$$

The union of the two sets makes up the arc set of the residual graph, $\mathcal{A}_p^{\Gamma} = \mathcal{A}_p^{\Gamma,open} \cap \mathcal{A}_p^{\Gamma,closed}$. Notice that, the residual capacity of an arc is calculated without the flows of the commodities in $\mathcal{P}_{ij}$. This means that the flows on any arc used by the commodities in $\mathcal{P}_{ij}$ are removed entirely from the network, and new paths must be found from their origin to their destination nodes.

The path on which the flow is redirected is determined for each commodity by solving the shortest path on its residual graph with arc costs:

$$c_{kl}^{\Gamma} = \begin{cases} f_{kl} + c_{kl} \cdot \Gamma^p & if \ (k,l) \in \mathcal{A}^{\Gamma,closed} \\ c_{kl} \cdot \Gamma^p & if \ (k,l) \in \mathcal{A}^{\Gamma,open} \end{cases}$$

The cost of using a closed arc represents the cost of opening it, f_{kl}, plus the cost of routing the commodity. The cost of using an open arc is the cost of routing the commodity only. Solving the shortest path for a commodity $p \in \mathcal{P}_{ij}$ results in a path π^p with set of arcs $\mathcal{A}_p^{\Gamma}(\pi^p)$. The subset of arcs in $\mathcal{A}_p^{\Gamma}(\pi^p)$ that require the original arcs to be opened is denoted $\Delta Y_p^{\pi} = \{(i,j) \in \Delta Y_p^{\pi} \mid (i,j) \in \mathcal{A}_p^{\Gamma}(\pi^p) \wedge y_{ij} = 0\}$ and the total cost of using path π^p can be expressed as $\Delta Z_p^{\Gamma} = \sum_{(i,j) \in \mathcal{A}_p^{\Gamma}(\pi^p)} c_{ij}^{\Gamma}$.

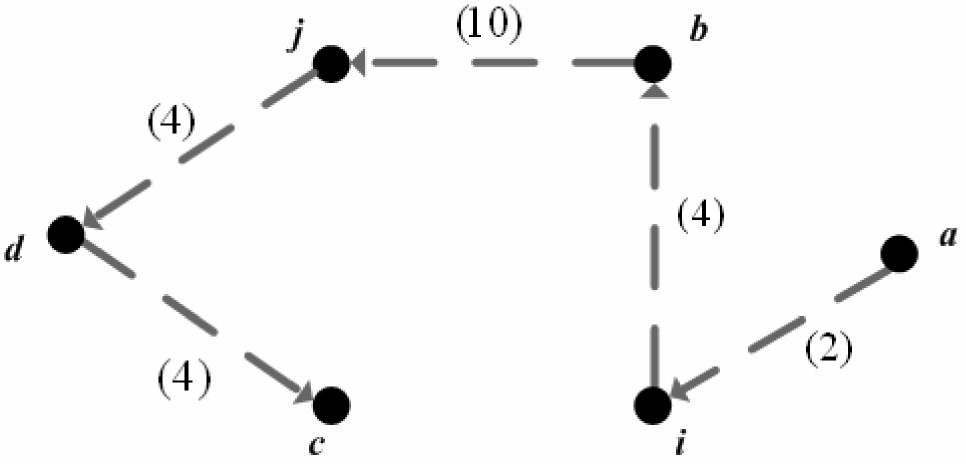

Figure 8. Residual Network for Dark-Grey Commodity Closing Arc *(i,j)*

Figure 8 shows the residual graph for the dark-grey commodity obtained closing arc *(i,j)* in Figure 7. Arc costs are shown on each arc. Notice, e.g., that arc *(j,c)* is not included in the residual graph because its capacity u_{jc}=1 is less than the required redirection flow $\Gamma^p = 2$. Similarly arc *(b,a)* is not included because its residual capacity is zero. The shortest path (and coincidentally the only feasible path) in the residual network is *(a,i)*, *(i,b)*, *(b,j)*, and *(j,d)* and requires the opening of arc *(b,j)* in the original network. Figure 9 displays the updated network.

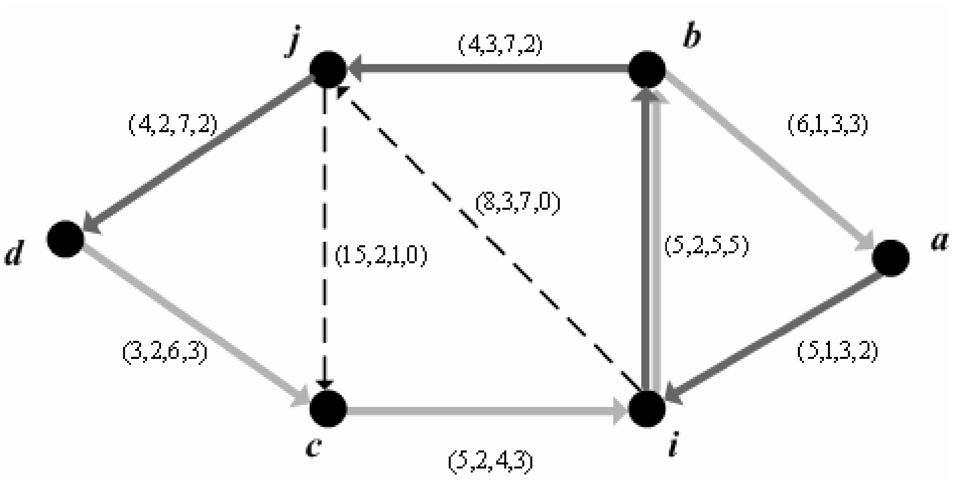

Figure 9. Updated Network Design and Flows

We perform flow redirection sequentially for each commodity. Because the design changes when a closed arc is used (i.e., opened) in the path, and because the residual capacity of several open arcs is modified as well, the residual network needs to be updated after redirecting each commodity $p \in \mathcal{P}_{ij}$. In the current implementation, commodities are redirected in lexicographical order. Whenever a commodity has been redirected or when no path may be found (the corresponding move is infeasible), it is removed from the set $\mathcal{P}_{ij}$.

The procedure used to calculate the total system cost of a neighbor obtained by closing arc *(i,j)* is summarized in Figure 10. The procedure starts by initializing the total system cost to that of the incumbent less the fixed cost, f_{ij}, of arc *(i,j)* that is to be closed and less the routing cost of commodities $\mathcal{P}_{ij}$ on all arcs. The total system cost is updated following each commodity redirection by adding the new path cost. The design vector is updated with the arcs to be opened, ΔY_p^{π}, and the flow vector is updated with the flow of the new commodity path. Repeating the update of the flow and design vectors results in an approximation for the neighbor solution $(\widetilde{Y}_n, \widetilde{X}_n)$ and the corresponding estimated total system cost $\widetilde{Z}_n$; this is then used for the neighbor solution value comparison.

3.3.2 Opening arcs

Opening an arc can improve the node imbalances of an incumbent solution and thereby move the search towards a feasible solution. We therefore include arc-add moves in the exploration phase of the search.

Opening an arc means the fixed cost associated to it is incurred and must be added to the total system cost. It also implies that new routing possibilities for commodities may exist resulting in a decrease of the flow distribution cost. It is not clear at all, however, that this reduction is more important than the increase in fixed cost. Solving the CMCF would provide the answer to this question but, similarly to drop moves, we desire to avoid the burden of performing this computation for every possible add move. Thus, because the flow distribution of the incumbent is still feasible once a new arc is opened, we will simply assume that no other optimization is possible and approximate the total system cost for the candidate solution by adding the fixed cost of the arc to be opened to the total system cost of the incumbent. Ignoring the impact of the flow re-distribution overestimates the value of potential add moves, but we prefer to allocate the available computing time to the exploration of the solution space.

```
Initialize
– $\tilde{Z}_n := Z_{n-1} - f_{ij} - \sum_{p \in \mathcal{P}_{ij}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^p x_{ij}^p$
– $\tilde{Y}_n := Y_{n-1}$, except arc (i,j)
– $\tilde{X}_n := X_{n-1}$, except $x_{ij}^p = 0 \;\forall\, p \in \mathcal{P}_{ij}, (i,j) \in \mathcal{A}$
While $\mathcal{P}_{ij} \notin \emptyset$
– Construct residual graph $\Gamma^p$ for commodity $p \in \mathcal{P}_{ij}$
– Find shortest path $\pi^p$ to determine $\Delta Z_p^{\Gamma}$ and $\Delta Y_p^{\pi}$
Update
  - $\tilde{Z}_n = \tilde{Z}_n + \Delta Z_p^{\Gamma}$
  - $\tilde{Y}_n = \tilde{Y}_n + \Delta Y_p^{\pi}$
  - $\tilde{X}_n = \tilde{X}_n, set\, x_{ij}^p = \Gamma^p \;\forall (i,j) \in \mathcal{A}_p^{\Gamma}(\pi^p)$
  - Remove $p$ from $\mathcal{P}_{ij}$
```

Figure 10. Estimating the Total System Cost Following a Close Move

3.4 The candidate list for the exploration phase

The basic add/drop move has $|\mathcal{A}|$ potential neighbor solutions. An exhaustive search of the neighborhood may result in the examination of uninteresting moves and the waste of computational time. It is more effective to consider a limited number of "good" potential solutions only. A list of candidate arcs, that is a subset of network arcs, $\mathcal{C}_1^n \subseteq \mathcal{A}$, is therefore built for the incumbent solution at iteration *n*.

In the implementation proposed here, we determine the candidate list by concatenating four candidate sub-lists, each ordered according to a particular criterion:

$\mathcal{C}_f$, open arcs with the highest fixed cost;

$\mathcal{C}_v$, open arcs with the highest variable cost;

$\mathcal{C}_r^n$, open arcs with highest residual capacity;

$\mathcal{C}_p^n$, select arcs with the lowest estimated penalty value $\tilde{P}_n$.

In the current implementation, the length of each sub-list is fixed empirically at l_f, l_v, l_r, and l_p, respectively.

The motivations behind the four criteria are as follows. Closing an arc with a high fixed or variable cost can reduce the total system cost. Arcs with high residual capacity do not fully use their capacity and may therefore have a high marginal fixed cost per unit of flow. Closing an arc with large residual capacity could thus remove the fixed cost associated to the arc while only a small amount of flow would have to be redirected. Choosing an arc, be it either open or closed, with a low expected penalty value would decrease the solution value and thus move the search towards a feasible solution.

The order of the fixed cost $\mathcal{C}_f$ and variable cost $\mathcal{C}_v$ sub-lists is known *a priori* and will not change during with the progress of the algorithm. The residual-capacity and penalty sub-lists, $\mathcal{C}_r^n$ and $\mathcal{C}_p^n$, respectively, need to be updated at each iteration. The impact of closing an arc may involve opening other arcs and, due the interdependency between opening and closing arcs and the node imbalances, it is not possible (short of solving the resulting problem) to determine the exact penalty value of a potential neighbor solution when establishing the candidate list for an incumbent solution. An estimate of the penalty value and, in turn, of the order of the sub-list is proposed instead. This approximation is based on the simplifying assumption that the design only changes by adding or dropping the candidate arcs, without considering the impact of the single-path-based enhancement. The resulting system imbalance, $\tilde{\psi}_n^{\mathcal{N}}$, is then easily computed based on the incumbent solution's system imbalance, $\psi_{n-1}^{\mathcal{N}}$, by updating the imbalance changes in the candidate arc's two nodes:

$$\tilde{\psi}_n^{\mathcal{N}} = \psi_{n-1}^{\mathcal{N}} + \Delta(\psi_n^i) + \Delta(\psi_n^j), \text{ where } \Delta(\psi_n^i) = \left|\psi_n^i\right| - \left|\psi_{n-1}^i\right| \text{ and } \Delta(\psi_n^j) = \left|\psi_n^j\right| - \left|\psi_{n-1}^j\right|.$$

This operations is not sufficient, however, to estimate the maximal node imbalance, $\tilde{\psi}_n^{\max}$, of potential neighbor solution. To illustrate, let's assume that node *i* has the maximal node imbalance in the incumbent solution, $\left|\psi_{n-1}^i\right| = \psi_{n-1}^{\max}$, and that arc (i, j) is closed. Let us further assume that the absolute value of the node imbalance at node *i* is smaller in the potential neighbor solution than in the incumbent solution, $\left|\psi_n^i\right| < \left|\psi_{n-1}^i\right|$. There is no guarantee that the maximal node imbalance is equal to the node imbalance in node *i* for the potential neighbor solution, as there might be another node *k* with the same maximal imbalance in the incumbent solution, $\left|\psi_{n-1}^k\right| = \left|\psi_n^k\right| = \left|\psi_{n-1}^i\right| = \psi_n^{\max}$. The maximal imbalance for the potential solution would not change then, even though the imbalance at node *i* becomes smaller. To determine the real value of the maximal node imbalance, all the nodes in the network would need to be examined for potential neighbor solutions. To limit computational requirements, $\tilde{\psi}_n^{\max}$ is estimated using the following approximation:

$$\tilde{\psi}_n^{\max} = \begin{cases} \psi_{n-1}^{\max} & \text{if} \left|\psi_{n-1}^i\right| < \psi_{n-1}^{\max} \text{ and} \left|\psi_{n-1}^j\right| < \psi_{n-1}^{\max} \\ \max(\psi_n^i, \psi_n^j) & \text{otherwise} \end{cases}$$

The approximation assumes, correctly, that the maximal node imbalance does not change if the node imbalances at nodes *i* and *j* in the incumbent solution are both smaller than the maximal node imbalance. When, however, the imbalance at node *i* or *j* in the incumbent solution is equal to the maximal node imbalance, $\psi_{n-1}^{\max}$, the value may change.

When the maximal node imbalance increases, the new maximal node imbalance is calculated correctly by the approximation. On the other hand, when the node imbalance decreases, there is, as explained, no guarantee that there is not another node in the network with the maximal node imbalance. However, to reward the decrease of a (large) node imbalance, the estimated penalty cost is calculated as if the maximal node imbalance for the network has decreased.

The estimated penalty value of a potential neighbor solution is based on the potential system imbalance and the estimated potential maximal imbalance $\widetilde{P}_n = \widetilde{f} \cdot \widetilde{\psi}_n^{\mathfrak{N}} \cdot \widetilde{\psi}_n^{\max}$, while the estimated total solution value for a potential neighbor solution, used to order the penalty sub-list $\mathcal{C}_p^n$, is calculated as the sum of the estimated total system cost (see Section 3.3) and the estimated penalty value $\tilde{V}_n = \tilde{Z}_n + \tilde{P}_n$.

3.5 Tabu list for the exploration phase

The tabu list contains information on the previous l_t solutions and, to avoid cycling, solutions in the tabu list may not be considered as potential neighbors of the incumbent. Since moves are based on flipping the value of a single arc-design variable, a move selecting the same arc within the span of the following l_t iterations is defined as tabu. The tabu list can thus be seen as a list holding the previous l_t selected arcs.

An arc added to the tabu list at iteration *n* thus stays tabu for an add/drop move until iteration $n+l_t$. In our definition of neighborhoods and moves, however, the value of a design variable may also be modified during the flow redirection operation, as described at Section 3.3. Applying the tabu criterion to these modifications would be abusive. In fact, given the differences in the arc manipulations performed (single arc versus several arcs in paths), the solutions yielded by the two operations are quite different in most cases. Furthermore, since redirection paths point toward potential improvements in the problem solution, the inclusion of a tabu arc in a redirection path may be viewed as an indication that a new and better solution will be obtained. In Tabu Search vocabulary, this is identified as an *aspiration criterion*. In our context, this means that an arc in the tabu list can re-enter the solution if it is chosen on a commodity redirection path.

The tabu list is initialized as an empty list, $\mathfrak{T} = \{\emptyset\}$. For iterations $n \le l_t$, the arc selected at iteration *n*, $c_n \in \mathcal{C}_1^n$, is added to the tabu list $\mathfrak{T} = \mathfrak{T} \cup c_n$. For iterations $n > l_t$, the first (oldest) element of the tabu list, arc c_{n-l_t}, is removed and the arc of the current iteration $c_n \in \mathcal{C}_1^n$ is put at the end of the tabu list to get $\mathfrak{T} = \mathfrak{T} \setminus c_{n-l_t} \cup c_n$.

3.6. Neighborhood structure for the feasibility phase

The neighborhood structure of the exploration phase allows moves between infeasible solutions with respect to the design-balance constraints. There is therefore no guarantee that the exploration of the solution space in this phase will result in feasible solutions. The feasibility phase is designed specifically to find feasible solutions.

Assume an infeasible incumbent solution c_{n-1} is available at the termination of the exploration phase, i.e., a number of nodes are imbalanced. Consider once more the network in Figure 5. Assuming arc (i,j) is not included in the design, the solution is feasible with respect to the design-balance constraints. Adding arc (i,j) makes the solution infeasible because there are imbalances at nodes i and j. More specifically, the imbalance at node i is -1, while that of node j is +1. It is easily seen that for any imbalanced solution, the sum of the absolute negative imbalances is always equal to the sum of the positive imbalances. The addition of an arc (i,j) to the design will decrease the imbalance by 1 at origin node i and increase the imbalance by 1 at destination node j.

The idea behind the neighborhood structure for the feasibility phase is to determine a pair of oppositely imbalanced nodes, to close or open a path of arcs between them, and thereby reduce the imbalances at the two nodes. Closing a path from a node with negative imbalance to a node with a positive imbalance will reduce the imbalance at both nodes, as will opening a path from a node with a positive imbalance to a node with negative imbalance.

Figure 11 illustrates connecting two nodes i and j by closing and opening a path from i to j, in the top and lower networks, respectively. The imbalance at each node is indicated. Closing or opening a path does not impact the imbalances at the nodes of the path different from i and j, because both an incoming and an outgoing arc are removed or added to each of these nodes, respectively. Feasible solutions with respect to the design-balance constraints can be achieved by closing the arcs on path (i,d), (d,a), and (a,j) as shown with grey arcs in the top network, or by opening the arcs on path (j,h), (h,g), (g,f), and (f,i) in the lower network.

Iteratively matching oppositely imbalanced nodes and closing or opening paths between them eventually reduces the imbalances to zero at all the nodes in the network and generates a feasible solution.

3.7. Moves and neighbor solution evaluation for the feasibility phase

The template for the neighborhood moves for the feasibility phase is the add/drop procedure. Instead of adding or dropping a single arc, as in the exploration phase, however, the add/drop procedure is performed on paths. As indicated previously, to achieve a feasible solution with respect to the design-balance constraints, we iteratively match pairs of oppositely-signed imbalanced nodes and connect them with paths that close or open arcs. If the procedure is successful, $\psi^{\mathfrak{N}}/2$ iterations are performed, where $\psi^{\mathfrak{N}}$ is the total network imbalance, which is equivalent to reducing the imbalance at two nodes (one negative and one positive) at each iteration. Otherwise, the exploration search is restarted from the best infeasible incumbent of the feasibility phase.

The following procedure is used to match the nodes:

- The maximum imbalance node is identified, $i{:}\{\left|\psi^i\right| = \psi^{\max}\}$;
- The set of nodes with oppositely signed imbalances are identified, $\mathfrak{N}^c{:}\{j \in \mathfrak{N}^c | \psi^i \cdot \psi^j < 0\}$.

- A candidate set, $\mathcal{C}_2^{n-1}$, of paths to open or close is then constructed between node *i* in $\mathcal{N}^c$ and any of the other nodes in $\mathcal{N}^c$. The candidates are evaluated and the best path is selected as the move to implement. Section 3.8 addresses the issue of selecting the candidate set.

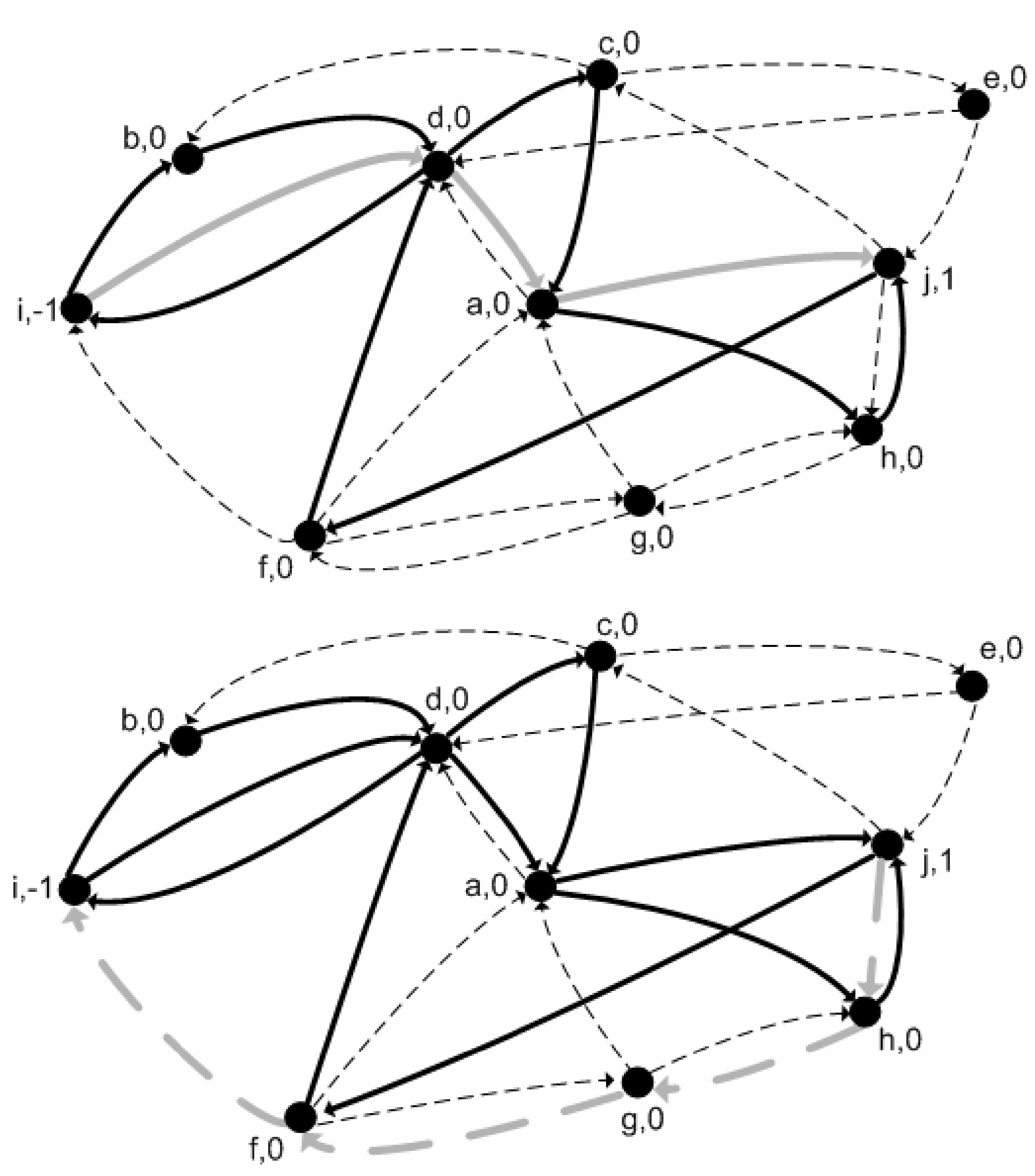

Figure 11. Eliminating Node Imbalances through Path Opening and Closing

Opening or closing a path modifies significantly the design and may have a big impact on the commodity flows. When a path is closed, many commodities have to be redirected. Opening a path may result in better route possibilities for some commodities. Any of these operations may start a ripple effect of flow redistribution. Such modifications are too deep and complex to be approximated by the procedures introduced previously for adding or closing individual arcs. Therefore, to get a reasonable evaluation of the impact of a feasibility move on the flow distribution, the CMCF problem is solved for each candidate path in $\mathcal{C}_2^{n-1}$. The design and flow vectors obtained by adding or removing the path resulting in the smallest total system cost, Z_c^*, are implemented resulting in a new incumbent solution.

3.8. The candidate list for the feasibility phase

Several paths could possibly be closed or open for a single pair of imbalanced nodes. When several imbalanced nodes exist, the number of possible paths between positive and negative imbalanced nodes is large. Furthermore, because the evaluation of each potential path is performed by solving the CMCF problem, this step may represent a costly procedure with respect to computational time. We thus restrict the number of examined candidate paths to a limited number of interesting ones.

We propose a candidate list containing four paths. Each of the four candidate paths is found by solving the shortest path problem between node $i : \left\{ \left| \psi^i \right| = \psi^{\max} \right\}$ and nodes $j \in \mathcal{N}^c$, on four modified networks with appropriately customized arc costs. For each of the four modified networks, the best of the calculated shortest paths is selected as candidate. The four modified networks and the associated arc costs are:

Small-commodity-flow network. The network $\mathcal{G}^1 = (\mathcal{N}, \mathcal{A}^1)$ is built of the set of open arcs $\mathcal{A}^1 : \{(i,j) \in \mathcal{A}^1 \mid (i,j) \in \mathcal{A} \wedge y_{ij} = 1\}$ and the arc costs are set to the their total flow values $c_{ij}^1 = \sum_{p \in \mathcal{P}} x_{ij}^p \ \forall\, (i,j) \in \mathcal{A}^2$. The candidate path identified on this network, π^1, should have little flow on it and closing it would require only a relatively small amount of commodity flow to be redirected. Thus, closing this path offers a significant chance of finding a feasible flow while saving the fixed costs of the arcs.

High-fixed-cost network. The network $\mathcal{G}^2 = (\mathcal{N}, \mathcal{A}^2)$ is built of the set of open arcs $\mathcal{A}^2 : \{(i,j) \in \mathcal{A}^2 \mid (i,j) \in \mathcal{A} \wedge y_{ij} = 1\}$ and the arc costs are set equal to the largest fixed cost of all arcs in $\mathcal{A}$ minus the fixed cost of each arc $c_{ij}^2 = \max_{(k,l) \in \mathcal{A}} (f_{kl}) - f_{ij} \ \forall\, (i,j) \in \mathcal{A}^2$. The candidate path computed on this network, π^2, will consist of high-cost arcs. Closing the arcs on this path will remove the high fixed costs and reduce the total system cost.

Small-variable-cost network. The network $\mathcal{G}^3 = (\mathcal{N}, \mathcal{A}^3)$ is built of the set of inversed closed arcs $\mathcal{A}^3 : \{(i,j) \in \mathcal{A}^3 \mid (j,i) \in \mathcal{A} \wedge y_{ij} = 0\}$. The arc costs are set to the variable costs of the arcs in $\mathcal{A}$, $c_{ij}^3 = c_{ji} \ \forall\, (i,j) \in \mathcal{A}^3$. The candidate path from this network, π^3, will consist of arcs with small variable costs. Opening this path may enable some cheaper routing alternatives for commodities, thereby reducing the variable cost component of the total system cost although fixed costs are incurred by opening the arcs in the path.

Low-fixed-cost network. The network $\mathcal{G}^4 = (\mathcal{N}, \mathcal{A}^4)$ is built of the set of inversed closed arcs $\mathcal{A}^4 : \{(i,j) \in \mathcal{A}^4 \mid (j,i) \in \mathcal{A} \wedge y_{ij} = 0\}$ and the arc cost are set to the fixed cost of the arcs in $\mathcal{A}$, $c_{ij}^4 = f_{ji} \ \forall\, (i,j) \in \mathcal{A}^4, (j,i) \in \mathcal{A}$. The candidate path computed on this network, π^4, will consist of arcs with low fixed cost. Opening this path will result in an increase in the total system cost and is included as a “last resort” candidate to reduce the absolute imbalances by opening the cheapest possible path.

Because the candidate list examines a subset of four paths only, there is no guarantee that it is possible to determine a new incumbent solution. If no feasible solution is found from any of the candidates in $\mathcal{C}_2^{n-1}$, the feasibility phase is terminated and the exploration phase continues from the infeasible incumbent solution obtained thus far in the feasibility phase.

There could be other interesting candidate paths, possibly found on other graphs with different cost structures or alternative path-construction methods. However, as shall be shown in the next section, limiting the candidate list to the four proposed paths is more than adequate for the cases studied.

4. Computational results

The performance of the algorithm is tested on the network design instances used in Crainic, Gendreau, and Farvolden (2000) and Ghamlouche, Crainic, and Gendreau (2003). There are two sets of instances, identified as R-problems and C-problems, out of which a number of the most difficult ones have been selected for the final experimentation. Both sets of instances are general transshipment networks with no parallel arcs and one commodity per origin-destination pair. The instances differ in size (nodes, arcs, commodities). Furthermore the fixed cost, variable cost, and capacity vary in relative importance for each of the instances. The same arc unit cost is used for all commodities.

The network design instances have varying arc capacities. It must be presumed that in a scheduled service network design application, service capacities are somewhat homogenous. However, intuitively, it can be assumed that it is more difficult to solve the DBCMND model on networks with heterogeneous capacities due to the fixed cost/capacity trade-off. We thus assume the results presented here do not over-estimate the performance of the proposed algorithm on scheduled service design network applications.

The shortest path instances within the Tabu Search algorithm are solved with the ML-Thresh-X2 procedure from Jørgensen and Madsen (2004), while the CMCF problems are solved using the Xpress-MP LP-solver. The algorithm is implemented in C++ using Microsoft Visual Studio NET 2003 using the Xpress-BCL builder component library. The performance of the Tabu Search meta-heuristic is compared to that of the Xpress-MP MIP-solver. All computations are performed on a laptop computer with a 2.26 GHz Intel Pentium 4 processor.

4.1 Initial parameter tuning

Eight parameters appear in the Tabu Search meta-heuristic and need to be tuned. An initial calibration phase was therefore performed with the goal of finding a single parameter configuration that could be applied to all instances regardless of their network characteristics.

Ten instances were selected from the C and R problem sets for the parameter tuning experiment. These problems represent the various possible network characteristics and are listed in Table 1. For each problem indicated in the first column, the next three columns indicate the problem dimensions: numbers of nodes, arcs, and commodities, respectively. The “capacity ratio” figure is computed as the ratio of commodity demand over the total network capacity; A “tight” capacity ratio signifies limited capacity in relation to the total demand and a “loose” figure, a surplus of capacity. The “cost ratio” figure is the ratio of the fixed cost over the variable cost of arcs. The “Opt. solution” column indicates whether an optimal solution was found for the instance solving the problem within a 3600s time limit using the Xpress-MP MIP solver. As can be seen, the selected instances display various combinations of characteristics and are representative of the instances in the data sets.

Instance name	Network size			Network characteristic		
	Nodes	Arcs	Commodities	Capacity ratio	Cost ratio	Opt. solution
R10,F05,C2	20	120	40	Medium	Medium	Yes
R12,F10,C2	20	120	200	Medium	High	Yes
R13,F01,C8	20	220	40	Tight	Low	Yes
R15,F10,C8	20	220	200	Tight	High	Yes
C20,230,200,F.L.	20	230	200	Loose	High	No
R16,F1O,C1	20	314	40	Loose	High	No
R17,FO1,C1	20	318	100	Loose	Low	Yes
R18,F05,C2	20	315	200	Loose	Medium	No
C30,520,100,V,T	30	519	100	Tight	Low	No
C100,400,30,F,L,10	100	400	30	Loose	High	No

Table 1. Problem Instances for Calibration

The parameter tuning experiment was conducted using a 2-level 2^8 factorial experimentation plan (Montgomery, 2000), where each of the eight parameters was evaluated on two levels, low and high. Table 2 displays for each parameter the section where it was first introduced and the values for the two levels. The details of the experiment can be found in Pedersen (2006). A single parameter configuration was selected based on the results of this experimentation and was used to evaluate the algorithm performance. The selected parameter values are shown in the last column of Table 2. The results of the calibration gave indications of a robust algorithm with respect to the parameter settings. A claim further supported by the computational results in the following section.

4.2. Tabu Search versus MIP-Solver

The algorithms were tested on instances from the R and C data sets. The 24 most difficult instances have been selected from the C data set and the 54 most difficult ones from the R data set. The TS meta-heuristic add the Xpress-MP MIP solver were applied to each of the 78 problem instances with a 3600 seconds CPU-time limit.

The computational results for the selected C and R instances are shown in Tables 3 and 4, respectively. The two tables are composed identically. The “XpressMP” column shows the results obtained with the Xpress-MP MIP solver within the 3600 seconds computational time limit. The “Bound” and “XP/Bound” columns show the lower bound of the instance obtained by the MIP-solver and the relative gap between the MIP solution and this lower bound, respectively. The “TS” column gives the results obtained using the Tabu Search meta-heuristic within 3600 CPU seconds with the selected parameter setting. Finally the “TS/XP” and the “TS/Bound” columns show the relative gaps between the Tabu Search and the MIP-solver results, and the TS algorithm results and the lower bound, respectively. A ‘n/a’ signifies that no integer solution was obtained within the time limit. Note that for the nine C problems and the two R problems for which the MIP-solver failed to find a feasible integer solution, the TS algorithm succeeded. The TS algorithm ability to find feasible solutions for all 78 instances gives a first indication of its robustness.

Parameter	**Section**	**Values Tested**	**Value Selected**
Iteration improvement range l_i	3.1	10, 30	10
Improvement gap l_g	3.1	0.5%, 5%	5%
Penalty scaling factor Π	3.2	0.5, 2	0.5
Length of the tabu list l_t	3.5	5, 25	25
Number of fixed cost candidates l_f	3.4	5, 15	5
Number of variable cost candidates l_v	3.4	5, 15	15
Number of residual capacity candidates l_r	3.4	5, 15	15
Number of penalty capacity candidates l_p	3.4	5, 15	5

Table 2. Parameters of the Tabu Search Meta-heuristic

To analyze the results presented in Tables 3 and 4, we categorize the distribution of the relative gaps between the results obtained with the TS algorithm and with XpressMP’s MIP-solver. Table 5 shows that for 17 out of 24 C problems and for 16 out of 54 R problems, the TS algorithm outperformed the MIP-solver. In aggregated figures, the TS algorithm outperformed the MIP-solver in 33 out of 78 instances. These figures by themselves may not seem very convincing. However, it is important to note that the TS algorithm outperformed the MIP-solver for most of the larger and hence more difficult instances.

4.3. Analysis of Tabu Search algorithmic behavior

The scope of this section is to present the results of tests performed in order to better understand the behavior of the Tabu Search meta-heuristic. The analysis of these results should be of help in assessing the algorithm’s characteristics and applicability.

The first analysis regards the performance of the algorithm in relation to the network characteristics. An analysis of the relative gaps between the best feasible MIP-solver solution and the lower bound for the R-instances shows that they peak for the loose capacity instances (labeled FXX,C1) and become lower as the capacity becomes tighter. This somewhat different from the observations made for general network design problems where the tendency is that the higher the fixed/variable cost ratio, the larger the gap and, thus, the more difficult the instances (the F10,CX labeled instances being the most difficult independently of the problem size).

Instance	Nodes	Arcs	Com.	XpressMP	Bound	XP / Bound	TS	TS / XP	TS / Bound
C20,230,200,V,L	20	228	200	101.112	86.180	14,77%	102.919	1.76%	16.26%
C20,230,200,F,L	20	230	200	153.534	122.311	20.34%	150.764	-1.84%	18.87%
C20,230,200,V,T	20	229	200	105.840	92.608	12.50%	103.371	-2.39%	10.41%
C20,230,200,F,T	20	228	200	154.026	124.358	19,26%	149.942	-2.72%	17.06%
C20,300,200,V,L	20	294	200	81.184	73.894	8,98%	82.533	1.63%	10.47%
C20,300,200,F,L	20	292	200	131.876	110.533	16,18%	128.757	-2.42%	14.15%
C20,300,200,V,T	20	291	200	78.675	74.583	5,20%	78.571	-0.13%	5.08%
C20,300,200,F,T	20	291	200	127.412	106.628	16,31%	116.338	-9.52%	8.35%
C30,520,100,V,L	30	518	100	55.138	54.160	1,77%	55.981	1.51%	3.25%
C30,520,100,F,L	30	516	100	n/a	92.636	n/a	104.533	n/a	11.38%
C30,520,100,V,T	30	519	100	53.125	52.681	0,84%	54.493	2.51%	3.33%
C30,520,100,F,T	30	517	100	106.761	97.653	8,53%	105.167	-1.52%	7.14%
C30,520,400,V,L	30	520	400	n/a	111.054	n/a	119.735	n/a	7.25%
C30,520,400,F,L	30	520	400	n/a	143.335	n/a	162.360	n/a	11.72%
C30,520,400,V,T	30	516	400	n/a	114.725	n/a	120.421	n/a	4.73%
C30,520,400,F,T	30	518	400	n/a	148.210	n/a	161.978	n/a	8.50%
C30,700,100,V,L	30	680	100	48.849	48.400	0,92%	49.429	1.17%	2.08%
C30,700,100,F,L	30	680	100	65.516	59.483	9,21%	63.889	-2.55%	6.90%
C30,700,100,V,T	30	687	100	47.052	46.260	1,68%	48.202	2.39%	4.03%
C30,700,100,F,T	30	686	100	57.447	55.123	4,05%	58.204	1.30%	5.29%
C30,700,400,V,L	30	685	400	n/a	94.725	n/a	103.932	n/a	8.86%
C30,700,400,F,L	30	679	400	n/a	128.950	n/a	157.043	n/a	17.89%
C30,700,400,V,T	30	678	400	n/a	95.183	n/a	103.085	n/a	7.67%
C30,700,400,F,T	30	683	400	n/a	128.441	n/a	141.917	n/a	9.50%

Table 3. Computational Results for the C-Problem Instances (3600 sec.)

The previous observation seems to indicate that tight capacity DBCMND problems are easier to solve than loose capacity ones. This could be explained by the impact of the design-balance constraints. When the global capacity of the network is tight, a larger number of arcs need to be open in order to provide the means to flow the commodities. For DBCMND problem instances, this forces a high "open" versus "closed arcs" ratio to satisfy the design-balance constraints. Since many arcs need to be open in the final solution, the LP-relaxation is tighter and hence results in better lower bounds than for loose-capacity problem instances.

Instance	Nodes	Arcs	Com	XpressMP	Bound	XP / Bound	TS	TS / XP	TS / Bound
R13,F01,C1	20	220	40	147349	147349	0.00%	147837	0.33%	0.33%
R13,F05,C1	20	220	40	281283	269793	4.08%	281668	0.14%	4.22%
R13,F10,C1	20	220	40	404045	365588	9.52%	404434	0.10%	9.61%
R13,F01,C2	20	220	40	155887	155887	0.00%	159852	2.48%	2.48%
R13,F05,C2	20	220	40	295180	292249	0.99%	311209	5.15%	6.09%
R13,F10,C2	20	220	40	443831	411984	7.18%	470034	5.57%	12.35%
R13,F01,C8	20	220	40	218787	218787	0.00%	225339	2.91%	2.91%
R13,F05,C8	20	220	40	502811	480366	4.46%	512027	1.80%	6.18%
R13,F10,C8	20	220	40	812606	764751	5.89%	875984	7.24%	12.70%
R14,F01,C1	20	220	100	422709	422709	0.00%	431562	2.05%	2.05%
R14,F05,C1	20	220	100	835597	717432	14.14%	811102	-3.02%	11.55%
R14,F10,C1	20	220	100	1259890	954048	24.28%	1193950	-5.52%	20.09%
R14,F01,C2	20	220	100	452591	452591	0.00%	465762	2.83%	2.83%
R14,F05,C2	20	220	100	912189	848873	6.94%	942678	3.23%	9.95%
R14,F10,C2	20	220	100	1397100	1210230	13.38%	1401880	0.34%	13.67%
R14,F01,C8	20	220	100	704719	699947	0.68%	720882	2.24%	2.90%
R14,F05.C8	20	220	100	1696780	1659720	2.18%	1795650	5.51%	7.57%
R14,F10,C8	20	220	100	2874660	2681490	6.72%	2997290	4.09%	10.54%
R15,F01,C1	20	220	200	1042790	969165	7.06%	1039440	-0.32%	6.76%
R15,F05,C1	20	220	200	2297560	1718650	25.20%	2170310	-5.86%	20.81%
R15,F10,C1	20	220	200	3304180	2436030	26.27%	3194270	-3.44%	23.74%
R15,F01,C2	20	220	200	1176860	1164850	1.02%	1205790	2.40%	3.40%
R15,F05,C2	20	220	200	2723740	2368130	13.06%	2698680	-0.93%	12.25%
R15,F10,C2	20	220	200	4349910	3467490	20.29%	4447950	2.20%	22.04%
R15,F01,C8	20	220	200	2402800	2392650	0.42%	2472860	2.83%	3.24%
R15,F05,C8	20	220	200	5807050	5774260	0.56%	6067350	4.29%	4.83%
R15,F10,C8	20	220	200	9169890	9025170	1.58%	10263600	10.66%	12.07%
R16,F01,C1	20	314	40	140082	140082	0.00%	142692	1.83%	1.83%
R16,F05,C1	20	314	40	259840	236810	8.86%	261775	0.74%	9.54%
R16,F10,C1	20	314	40	364786	310584	14.86%	374819	2.68%	17.14%
R16,F01,C2	20	314	40	142381	142381	0.00%	145266	1.99%	1.99%
R16,F05,C2	20	314	40	275626	251711	8.68%	277307	0.61%	9.23%
R16,F10,C2	20	314	40	396966	341107	14.07%	391386	-1.43%	12.85%
R16,F01,C8	20	314	40	180199	178497	0.94%	187176	3.73%	4.64%
R16,F05,C8	20	314	40	396721	376351	5.13%	423320	6.28%	11.10%
R16,F10,C8	20	314	40	637944	571578	10.40%	649121	1.72%	11.95%
R17,F01,C1	20	318	100	364784	362602	0.60%	374016	2.47%	3.05%
R17,F05,C1	20	318	100	730195	583094	20.15%	718135	-1.68%	18.80%
R17,F10,C1	20	318	100	1150630	769660	33.11%	1041450	-10.48%	26.10%
R17,F01,C2	20	318	100	382593	382315	0.07%	393608	2.80%	2.87%
R17,F05,C2	20	318	100	761041	695446	8.62%	786198	3.20%	11.54%
R17,F10,C2	20	318	100	1195710	971426	18.76%	1162290	-2,88%	16.42%
R17,F01,C8	20	318	100	531791	523185	1.62%	539817	1.49%	3.08%
R17,F05,C8	20	318	100	1284720	1195670	6.93%	1348750	4.75%	11.35%
R17,F10,C8	20	318	100	2047390	1927200	5.87%	2227780	8.10%	13.49%
R18,F01,C1	20	315	200	869263	772151	11.17%	864425	-0.56%	10.67%
R18,F05,C1	20	315	200	1869230	1302720	30.31%	1640200	-13.96%	20.58%
R18,F10,C1	20	315	200	2390740	1793780	24.97%	2399230	0.35%	25.24%
R18,F01,C2	20	315	200	980178	900627	8.12%	962402	-1.85%	6.42%
R18,F05,C2	20	315	200	2146670	1703640	19.38%	1958160	-9.63%	11.62%
R18,F10,C2	20	315	200	n/a	2478890	n/a	2986000	n/a	16.98%
R18,F01,C8	20	315	200	1560790	1507840	3.39%	1617320	3.50%	6.77%
R18,F05,C8	20	315	200	4230970	3817170	9.78%	4268580	0.88%	10.58%
R18,F10,C8	20	315	200	n/a	6180620	n/a	7440780	n/a	16.94%

Table 4. Computational Results R-Problem Instances

Turning to the gaps between the TS and the MIP-solver solutions relative to the same the R instances, one notices that the Tabu Search largely outperforms the MIP-solver on high-fixed-cost, loose-capacity instances (F05,C1 and F10,C1) for the two largest instance groups (R17 and R18). Furthermore, the TS algorithm also outperforms the MIP-solver on the largest high fixed/variable cost ratio instances in R18. This indicates that the algorithm is robust with respect to instance attributes and appears particularly useful for addressing network instances where MIP-solvers may struggle.

Data set	n/a	<[-5%]	[-5%]-[-2.5%]	[-2.5%]-[0%]	[0%]-[2.5%]	[2.5%]-[5%]	>[5%]
C	9	1	6	1	6	1	0
R	2	5	5	4	19	12	7
C+R	11	6	11	5	25	13	7

Table 5. Comparative Distribution of Relative Gaps TS versus XpressMP

One notices that the performance of the Tabu Search algorithm on loosely capacitated problem instances is better than on very-tight capacitated instances. This behavior could be explained by the choice of redirecting flow on single paths when closing arcs in the Exploration phase (Section 3.3). Indeed, when arc capacities are tight, residual capacities are generally small and more than one path could be required for flow redirection. This results in infeasible moves for a number of commodities and a less thorough exploration of the search space. While the performance of the method is very good, this behavior points to the need for further research into more sophisticated, but still efficient flow-redirection mechanisms.

Considering actual scheduled service network design applications, a rather large number of potential services and departure times are usually considered. Fixed costs tend to be important in these cases reflecting the asset-related costs associated to operating vehicles and convoys. Variable costs, reflecting the cost of transporting commodities (passengers or freight) are less important. This translates into large networks with high fixed cost and loose capacity (considering the many possible services considered initially) characteristics. It therefore seems that the Tabu Search meta-heuristic we propose would perform well addressing DBCMND problem instances coming from transport service network design applications.

Candidate selection Phase 1			
l_f	l_p	l_r	l_v
33.20%	21.70%	31.80%	13.29%
Candidate selection Phase 2			
Path 1	Path 2	Path 3	Path 4
58.03%	12.16%	15.26%	14.55%

Table 6. Distribution of Candidate Selection for the Tabu Search

Let's now turn to the behavior of the algorithm relative to the candidate lists in the two phases of the Tabu Search. The goal is to investigate whether candidates are chosen from all four sub-lists in the exploration phase and all four candidate paths are selected in the feasibility phase. The results may be used to further tune the algorithm and remove eventual redundant computations.

Table 6 displays the distribution of the best neighbor selection from the candidate sub-lists in the explorations phase (Phase 1) and that of the selection of the four candidate paths in the feasibility phase (Phase 2). The figures are averages taken over all 78 instances. As can be seen from the table, all four candidate sub-lists and path candidates contribute to the selection of the best neighbor solution. Obviously, the actual selection distributions vary with the individual instances. Thus, for instance, the penalty sub-list has a tendency of not providing any best neighbor solutions for some instances. Then, as it is often the case, the algorithm could be more finely tuned for the particular setting of a given application. All strategies contribute to the performance of the algorithm when sets of problem instances are considered, however, which is the case for the present experimentation.

The third and last analysis we present in this section targets the interaction between the exploration and the feasibility phases and the resulting impact on the search trajectory according to the values of the total solution cost and of the solution value (total solution cost + penalty value) for the current solution.

Figure 12 shows the graphs of this evolution for instance C20,200,300,F,L. As can be seen, the initial solution r-DBCMND and rounding heuristic yield a solution with high cost and penalty values. The algorithm starts off with the exploration phase and switches to the feasibility phase after eleven iterations because of the improvement gap of 5% and the iteration range value of 10. Every ten iterations the feasibility phase sets in, which can be seen at the points where the two curves meet (penalty value=0). An interesting observation is that the best solution is found after 55 iterations of the exploration phase and 5 initiations of the feasibility phase. For the remaining time, the search oscillates above the best found solution. This could signify that the algorithm produces good feasible solution very fast for this rather easy particular instance. For more complex instances, the stopping criterion used terminates the search before the improvement in feasible solutions smoothes out, as illustrated in Figure 13.

Figure 13 shows the evolution of the solution values for instance C30,520,400,F,T. The points where the curves drop indicate when the feasibility phase was run. The solution value improves with each feasibility-phase run. The search is stopped on a time limit, however, and further improvements could probably be achieved. The small number of iterations performed for this instance result from the requirement to solve several CMCF problems for the feasibility phases, which is very computationally demanding. Thus, for example, between 500 and 1000 CPU seconds where required per run of the feasibility phase depending on the level of imbalance. This raises the question of whether an alternative evaluation method should be applied during the feasibility phase to speed the evaluations or if fewer candidate paths should be investigated. Later studies might shed new light on this issue. In the meantime, it should be noted that, although the feasibility phase is computationally demanding, it also proves to be the driving force behind improving the feasible solution values and achieving good feasible solutions.

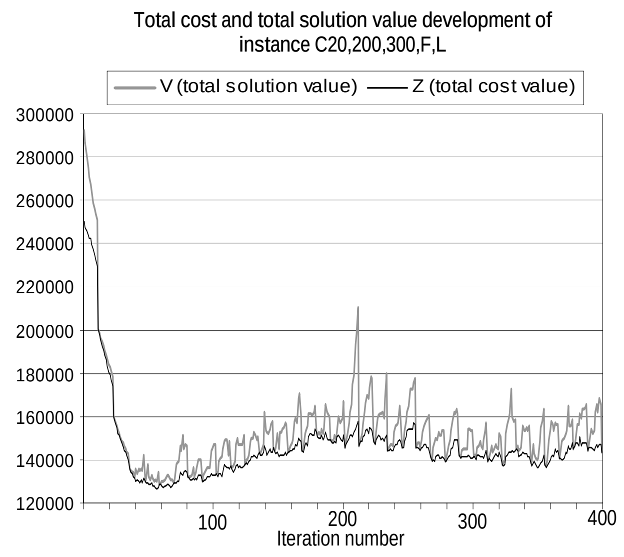

Figure 12. Evolution of Solution Values for Instance C20,200,300,F,L

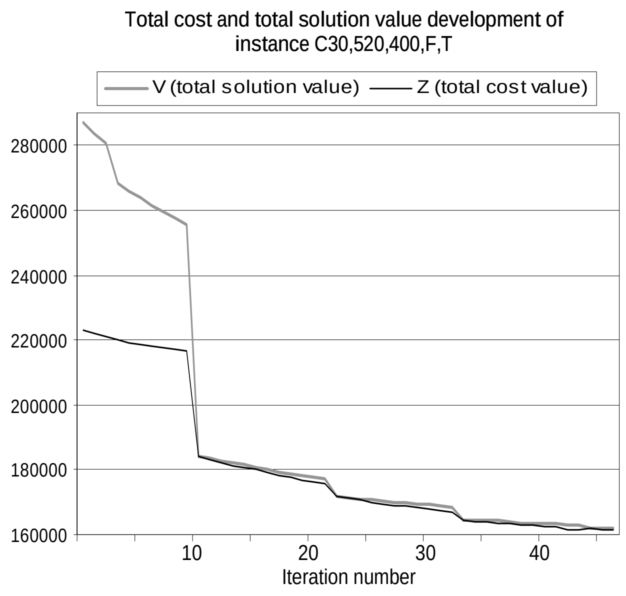

Figure 13. Evolution of Solution Values for Instance C20,520,400,F,T

4.4. Simple multi-search implementation investigation

The outcome of the initial parameter tuning was a parameter configuration that appeared to produce the overall best results for the TS algorithm. Nevertheless, different network characteristics might require different parameter configurations to obtain the best possible solutions. We examine this possibility by solving each instance with the same Tabu Search algorithm but with several parameter configurations and selecting the best result among those obtained by these configurations. We will thus see whether a significant benefit could be obtained and gauge the robustness of the method.

Note that the proposed methodology corresponds to the so-called *independent multi-search* strategy, much used to design parallel meta-heuristics (e.g., Crainic, 2005, Crainic and Nourredine, 2005, Cung *et al.*, 2002). According to this strategy, the algorithm would be deployed on different CPUs with different parameter configurations. The different runs would take place in parallel, the best solution among those found by the individual searches being selected as the overall best one.

We selected five additional parameter configurations, shown in Table 7 together with the one used in the previously. As can be seen, only six of the eight parameters vary in the six configurations. We have chosen to fix the improvement range and improvement gap due to the significant importance of the feasibility phase in finding good feasible solutions. To allow as many feasibility-phase executions as possible, the improvement range and the improvement gap were kept at their low and high level, respectively. The remaining 6 parameters were used at various levels.

Parameter	Description	Values					
l_f	# of Fixed cost candidates	5	5	10	5	10	5
l_p	# of Penalty cost candidates	5	5	5	5	10	5
l_r	# of Residual capacity candidates	15	5	5	10	10	15
l_v	# of Variable cost candidates	15	5	5	10	10	15
l_t	Tabu list length	25	25	20	20	10	10
	Penalty value scale	0.5	0.5	2	2	2	0.5
l_g	Improvement gap	5%	5	5	5	5	5
l_i	Improvement range	10	10	10	10	10	10
	Average score	**3.14**	**3.60**	**4.01**	**4.06**	**3.71**	**2.41**

Table 7. Parameter Settings for the Multi-search Experiment

The computational results from the parallel implementation can be seen in Tables 8 and 9. The three columns labeled "XpressMP", "TS", and "Parallel TS" display the computational results from the MIP-solver, the Tabu Search with the initial parameter configuration (previous results), and the best Tabu Search solution over the six parameter configurations, respectively. The figures in column "Parallel improv." show the relative improvement for the multi-search TS relative to the Tabu Search run with the initial-parameter configuration. The last two columns, "TS avg." and "St. dev.", show the average solution values obtained from the six parameter configurations and the standard deviation with respect to these average solution values, respectively. The computational results for the individual parameter settings for the individual runs can be found in Pedersen (2006).

The figures in Tables 8 and 9 indicate that some improvement can indeed be obtained from an independent multi-search implementation, where various parameter configurations characterize the individual searches. Most improvements are small, however, 59 out 78 instances displaying improvements less than 2%. There are a few instances, however, for which significant improvements can be achieved, e.g., the 10.31% improvement for instance R15,F10,C8. This confirms that a multi-search approach may improve the quality of the solutions obtained. It also raises the questions whether the proposed methodology with a single parameter setting is sufficiently robust.

We have thus analyzed the performance of the initial parameter configuration in relation to the other five. The “Average scores” row at the bottom of Table 7 displays an aggregated measure of “success” for each parameter configurations and provides the means of performance comparisons. A score from 1 to 6 (1 being the best) was given to each configuration for each problem instance, 1 representing the best solution value and 6, the worst. Table 7 displays the averages over the instances.

Instance	XpreesMp	TS	Parallel TS	Parallel improv.	TS avg.	St. dev.
C20,230,200,V,L	101.112	102.919	101.345	-1.55%	103.076	1.12%
C20,230,200,F,L	153.534	150.764	148.384	-1.60%	151.166	1.22%
C20,230,200,V,T	105.840	103.371	103.371	0.00%	105.559	1.85%
C20,230,200,F,T	154.026	149.942	144.766	-3.58%	147.652	1.39%
C20,300,200,V,L	81.184	82.533	80.269	-2.82%	80.951	1.07%
C20,300,200,F,L	131.876	128.757	126.258	-1.98%	128.297	1.20%
C20,300,200,V,T	78.675	78.571	78.444	-0.16%	79.749	1.27%
C20,300,200,F,T	127.412	116.338	116.338	0.00%	118.591	1.21%
C30,520,100,V,L	55.138	55.981	55.786	-0.35%	56.256	0.57%
C30,520,100,F,L	n/a	104.533	101.612	-2.87%	103.967	1.15%
C30,520,100,V,T	53.125	54.493	54.092	-0.74%	54.298	0.27%
C30,520,100,F,T	106.761	105.167	104.702	-0.44%	106.661	1.41%
C30,520,400,V,L	n/a	119.735	118.071	-1.41%	119.700	0.97%
C30,520,400,F,L	n/a	162.360	160.979	-0.86%	163.332	1.05%
C30,520,400,V,T	n/a	120.421	120.421	0.00%	121.310	0.68%
C30,520,400,F,T	n/a	161.978	161.978	0.00%	164.852	1.07%
C30,700,100,V,L	48.849	49.429	49.429	0.00%	49.723	0.45%
C30,700,100,F,L	65.516	63.889	63.292	-0.94%	63.635	0.50%
C30,700,100,V,T	47.052	48.202	47.487	-1.51%	47.969	0.52%
C30,700,100,F,T	57.447	58.204	57.187	-1.78%	58.343	1.20%
C30,700,400,V,L	n/a	103.932	103.932	0.00%	105.835	1.76%
C30,700,400,F,L	n/a	157.043	148.114	-6.03%	161.398	7.26%
C30,700,400,V,T	n/a	103.085	103.085	0.00%	103.797	0.69%
C30,700,400,F,T	n/a	141.917	138.609	-2.39%	141.743	1.17%

Table 8. Sequential versus Multi-search Performance Comparisons, C problems

A perfectly equal-quality performance would have yielded an average score of 3.5 for each configuration. While such a perfect match cannot be expected, the various configurations perform rather similarly. None stands out as particularly bad, the worst score being 4.06 for configuration 4. The configuration 6, with a score of 2.41, proves to be somewhat better than the first five. It is interesting to note that the only difference between this configuration and the initially selected one is a shorter tabu-list length. This is an indication that the initial set of problem instances used for calibration was probably too small and we misjudged somewhat the appropriate level for this parameter.

The scores indicate, nevertheless, that the Tabu Search meta-heuristic is robust with respect to the parameter selection. High-quality results are obtained over a range of parameter values for quite a diverse set of problem instances. This claim is further supported by investigating the standard deviations on the values of the best solutions obtained with each of the six parameter configurations. These results are displayed in the last columns of Tables 8 and 9. Only 10 out of the 74 instances display a standard deviation larger than 2%, the highest being 7,26% for instance C30,700,400,F,L. The standard deviation values are thus quite small, particularly considering the very different characteristics of the 78 problem instances, supporting the claim of a robust algorithm with respect to parameter settings and problem-instance characteristics.

5. Conclusions and Perspectives

This paper focused on a generic model for transportation service network design with asset management considerations. Traditionally, in many instances, fleet management issues were not directly included into tactical planning and processes aimed at designing the transportation plan. This situation evolves rapidly and addressing asset management issues jointly with the design of the service network and schedule increasingly appears as the most appropriate approach. This trend is readily apparent in rail transportation, intermodal services, in particular, but may also be observed in some motor-carrier operations.

In this paper, we model the asset positioning and utilization through constraints on asset availability at terminals. We denote these relations as “design-balance constraints” and focus on the design-balanced capacitated multi-commodity network design model (DBCMND), a generalization of the capacitated multi-commodity network design model (CMND) generally used in service network design applications. We analyze the problem and propose generic arc and cycle-based formulations for the DBCMND.

The paper also proposes a Tabu Search meta-heuristic framework to address the arc-based formulation of the DBCMND. The algorithm is based on two local-search phases. The first explores the design-defined solution space though a hybrid of an add/drop procedure and single path-based neighborhoods. Infeasible solutions with respect to the design-balance constraints are admissible in this phase. The second phase then aims to identify good feasible solutions. The Tabu Search meta-heuristic was applied on a wide range of network design problem instances from the literature and its behavior was analyzed through comparisons to the exact solution method offered by the Xpress-MP MIP-solver, as well as to the solutions of an independent multi-search procedure

Very good results were obtained in terms of computational efficiency and solution quality. The proposed method performed well compared to the exact MIP-solver over the entire set of test problem instances, with even better performance for the largest problem instances. The multi-search experiment showed that some improvement may be further achieved. It also indicated the proposed method is robust with respect to parameter setting and problem-instance characteristics.

Instance	XpressMP	TS	Parallel TS	Parallel imp.	TS avg.	St. dev.
R13,F01,C1	147349	147837	147.349	-0.33%	147.768	0.14%
R13,F05,C1	281283	281668	281.668	0.00%	284.719	0.92%
R13,F10,C1	404045	404434	400.656	-0.94%	405.197	0.70%
R13,F01,C2	155887	159852	156 585	-2.09%	158.638	0.87%
R13,F05,C2	295180	311209	307.180	-1.31%	310.148	0.53%
R13,F10,C2	443831	470034	437.396	-7.46%	454.649	2.83%
R13,F01,C8	218787	225339	223.541	-0.80%	225.043	0.52%
R13,F05,C8	502811	512027	510.887	-0.22%	520.720	1.49%
R13,F10,C8	812606	875984	839.174	-4.39%	870.713	2.43%
R14,F01,C1	422709	431562	427.872	-0.86%	430.679	0.47%
R14,F05,C1	835597	811102	811.102	0.00%	829.272	1.38%
R14,F10,C1	1259890	1193950	1.157.500	-3.15%	1.179.155	1.79%
R14,F01,C2	452591	465762	458.240	-1.64%	463.558	0.76%
R14,F05,C2	912189	942678	917.832	-2.71%	934.399	1.45%
R14,F10,C2	1397100	1401880	1.356.910	-3.31%	1.384.428	1.28%
R14,F01,C8	704719	720882	720.494	-0.05%	726.171	0.74%
R14,F05,C8	1696780	1795650	1.795.650	0.00%	1.838 053	1.80%
R14,F10,C8	2874660	2997290	2.997.290	0.00%	3.075.418	2.04%
R15,F01,C1	1042790	1039440	1.032.640	-0.66%	1.045.135	0.89%
R15,F05,C1	2297560	2170310	2.082.990	-4.19%	2.127.448	1.44%
R15,F10,C1	3304180	3194270	3.116.770	-2.49%	3.180.830	1.27%
R15,F01,C2	1176860	1205790	1.191.440	-1.20%	1.202.992	0.60%
R15,F05,C2	2723740	2698680	2.698.680	0.00%	2.725.057	1.02%
R15,F10,C2	4349910	4447950	4.310.340	-3.19%	4.397.752	1.26%
R15,F01,C8	2402800	2472860	2.465.650	-0.29%	2 473.680	0.44%
R15,F05,C8	5807050	6067350	5.969.370	-1.64%	6.044.925	1.06%
R15,F10,C8	9169890	10263600	9.304.650	-10.31%	9.840.033	4.61%
R16,F01,C1	140082	142692	140.149	-1.81%	141.531	0.70%
R16,F05,C1	259840	261775	261.775	0.00%	263.875	0.98%
R16,F10,C1	364786	374819	360.884	-3.86%	366.544	1.59%
R16,F01,C2	142381	145266	143.921	-0.93%	145.350	0.92%
R16,F05,C2	275626	277307	273.024	-1.57%	276.821	1.20%
R16,F10,C2	396966	391386	387.601	-0.98%	391.869	0.86%
R16,F01,C8	180199	187176	185.397	-0.96%	189.834	1.83%
R16,F05,C8	396721	423320	419.945	-0.80%	429.169	1.73%
R16,F10,C8	637944	649121	647.212	-0.29%	662.796	1.99%
R17,F01,C1	364784	374016	365.913	-2.21%	368.587	0.81%
R17,F05,C1	730195	718135	702.957	-2.16%	717.800	1.40%
R17,F10,C1	1150630	1041450	1.026.040	-1.50%	1.036.430	0.73%
R17,F01,C2	382593	393608	389.249	-1.12%	392.201	0.42%
R17.F05,C2	761041	786198	786.198	0.00%	800.117	1.23%
R17,F10,C2	1195710	1162290	1.159.440	-0.25%	1.172.777	1.08%
R17,F01,C8	531791	539817	539.817	0.00%	549.197	1.14%
R17,F05,C8	1284720	1348750	1.323.330	-1.92%	1.345.490	1.00%
R17,F10,C8	2047390	2227780	2.207.590	-0.91%	2.246.002	2.03%
R18,F01,C1	869263	864425	864.425	0.00%	879.355	0.88%
R18,F05,C1	1869230	1640200	1.627.700	-0.77%	1.668.745	1.73%
R18,F10,C1	2390740	2399230	2.366.280	-1.39%	2.425.505	2.95%
R18,F01,C2	980178	962402	962.402	0.00%	971.682	0.78%
R18,F05,C2	2146670	1958160	1.958.160	0.00%	1.987,065	1.59%
R18,F10,C2	n/a	2986000	2.986.000	0.00%	3.012.392	0.90%
R18,F01,C8	1560790	1617320	1.613.790	-0.22%	1.619.775	0.35%
R18,F05,C8	4230970	4268580	4.268.580	0.00%	4.421.490	2.75%
R18,F10,C8	n/a	7440780	7.194.120	-3.43%	7.610.975	4.53%

Table 9. Sequential versus Multi-search Performance Comparisons, R problems

A number of interesting research directions may be identified. Considering the proposed Tabu Search framework, a first area where to improve algorithm performance is using a dynamic adjustment of the parameter configuration. This should provide the means to guide the algorithm according to the characteristics of the particular instance and, hopefully, improve the performance. A second direction concerns the enhancement of the method through additional phases targeting particular problem or search characteristics. Thus, for example, we observed that the feasibility phase of the current implementation seems to struggle on network instances with a sparse arc structure. This is mainly due to the fact that when the network is sparse, paths to open are hard to find, while paths to close often result in infeasible minimum cost network flow sub-problems. A scarcity-detection mechanism and a phase directed toward such instances could help. This potential 3rd phase could use partial instead of full paths in defining neighborhoods and, thus, diversify the imbalance structure by shifting the imbalances between nodes. This brings us to the possibility of introducing additional, so-called long-term memory mechanisms and associated intensification and diversification schemes. The former would make sure that the solution space is fully explored around good solutions, while the latter would provide the means to direct the search toward previously unexplored regions. Different directions of research concern alternative formulations, e.g., the cycle-based model, and solution approaches, including exact algorithms. We have initiated work on some of these research directions and plan to report results in the near future.

Acknowledgements

This work was supported by the Virtual Centre for Logistics and Freight Transport (CLG) in Denmark, the Danish Technical Research Council, and Atkins Denmark. The Natural Sciences and Engineering Research Council of Canada (NSERC) offered additional support through its Discovery Grant program.

The authors would like to take this opportunity to thank two anonymous referees for their detailed and insightful comments, which helped us write a better paper.

References

Ahuja, R.K., Magnanti, T.J., Orlin, J.B., and Reddy, M.R. 1995. Applications of Network Optimization. *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, M. Ball, T.L. Magnanti, C.L. Monna, and G.L. Nemhauser (Eds.), pages 1-83, North-Holland, Amsterdam.

Andersen, J. and Christiansen, M. 2006. *Optimal Design of New Rail Freight Services*, Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology.

Armacost, A.P., Barnhart, C., and Ware, K.A. 2002. Composite Variable Formulations for the Express Shipment Service Network Design. *Transportation Science* 36(1):1-20.

Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. 1997. Network Design. *Annotated bibliographies in combinatorial optimization,* M. Dell'Amico, F. Maffioli, and S. Martello (Eds.), pages 311-334. John Wiley & Sons, New York, NY.

Bektaş, T and Crainic, T.G. 2007. A Brief Overview of Intermodal Transportation. *Logistics Engineering Handbook*, G.D. Taylor (Ed.), Taylor and Francis Group (forthcoming).

Cordeau, J.-F., Toth, P., and Vigo, D. 1998. A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science* 32(4):380-404.

Christiansen, M., Fagerholt, K., and Ronen, D. 2004. Ship Routing and Scheduling: Status and Perspectives, *Transportation Science* 38(1):1-18.

Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. 2007. Maritime Transportation, Chapter 4 in *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, C. Barnhart and G. Laporte (Eds.), pages 189-284, North-Holland, Amsterdam.

Crainic, T.G. 2005. Parallel computation, Co-operation, Tabu Search. *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. C. Rego and B. Alidaee (eds.), pages 283-302, Kluwer, Norwell, MA.

Crainic, T.G. 2003. Long-haul Freight Transportation, in *Handbook of Transportation Science*, R.W. Hall (Ed.), pages 451-516, Kluwer Academic Publishers, New York, 2nd Edition.

Crainic, T.G. 2000. Service Network Design in Freight Transportation. *European Journal of Operational Researc*h 122:272-288.

Crainic, T.G., Gendreau, M., Soriano, P. and Toulouse, M. 1993. A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research* 41:359-383.

Crainic, T.G., Gendreau, M., Farvolden, J.M. 2000. A Simplex-based Tabu Search Method for Capacitated Network Design. *INFORMS Journal on Computing* 12(3):223–236.

Crainic, T.G. and Kim, K.H. 2007. Intermodal Transportation, Chapter 8 in *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, C. Barnhart and G. Laporte (Eds.), pages 467-537, North-Holland, Amsterdam.

Crainic, T.G. and Laporte, G. 1997. Planning Models for Freight Transportation, *European Journal of Operational Research* 97(3): 409-438.

Crainic, T.G., Nourredine, H. 2005. Parallel Meta-heuristics Applications. *Parallel Metaheuristics*. E. Alba (Ed.), pages 447-494, John Wiley & Sons, Hoboken, NJ.

Cung, V.-D., Martins, S.L., Ribeiro, C.C., and Roucairol, C. 2002. Strategies for the Parallel Implementations of Metaheuristics, *Essays and Surveys in Metaheuristics*, C.C. Ribeiro and P. Hansen (Eds.), pages 263-308, Kluwer Academic Publishers, Norwell, MA.

Ghamlouche, I., Crainic, T.G., Gendreau, M. 2003. Cycle-based Neighbourhoods for Fixed-charge Capacitated Multicommodity Network Design. *Operations Research* 51(4):655-667.

Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research* 1(3):533-549.

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

Jørgensen, R.M. and Madsen, O.B.G. 2004. *Shortest Paths and Real Road Networks*. Centre for Traffic and Transport, Technical University of Denmark.

Koskosidis, Y.A., Powell W.B., and Solomon, M.M. 1992. An Optimization-based Heuristic for Vehicle Routing and Scheduling with Soft Time Windows Constraints. *Transportation Science* 26:69–85.

Magnanti, T.L. and Wong, R.T. 1984. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science* 18(1):1-55.

Minoux, M 1986. Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. *Networks* 19:313-360.

Montgomery, D.C. 2000. *Design and Analysis of Experiments*, 5th edition. Wiley.

Pedersen, M.B. 2006. *Optimization Models and Solution Methods for Intermodal Transportation*. PhD thesis, Centre for Traffic and Transport, Technical University of Denmark.

Pedersen, M.B. and Crainic, T.G. 2007, Optimization of Intermodal Freight Service on Train Canals, Publication, CIRRELT – Center for Research on Transportation, Université de Montréal.

Powell, W.B. 1986. A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks. *Transportation Science* 20(4):246–357.

Smilowitz KR, Atamtürk A, and Daganzo CF. 2003. Deferred Item and Vehicle Routing within Integrated Networks. *Transportation Research Part E: Logistics and Transportation* 39:305-323