CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Scheduled Service Network Design for Freight Rail Transportation

Endong Zhu
Teodor Gabriel Crainic
Michel Gendreau

December 2013

CIRRELT-2013-88

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

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Scheduled Service Network Design for Freight Rail Transportation

Endong Zhu[1], Teodor Gabriel Crainic[2,3,*], Michel Gendreau[2,4]

[1] App Annie, Suite 2802, Tower D, Sanlintun Soho 8 Gongrentiychang Beilu, Chaoyang, Beijing, China 100027

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

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

[4] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. The paper addresses the scheduled service network design problem for freight rail transportation. The proposed model integrates service selection and scheduling, car classification and blocking, train make up, and routing of time-dependent customer shipments, based on a cyclic three-layer space-time network representation of the associated operations and decisions, their relations and time dimensions. The paper also proposes a matheuristic solution methodology integrating slope scaling, a dynamic block-generation mechanism, long-term memory-based perturbation strategies, and ellipsoidal search, a new intensification mechanism to thoroughly explore very large neighborhoods of elite solutions restricted using information from the history of the search. Experimental results show that the proposed solution method is efficient and robust, yielding high-quality solutions for realistically-sized problem instances.

Keywords: Scheduled service network design, rail freight transportation, capacitated multi-commodity network design, slope scaling, ellipsoidal search.

Acknowledgements. While working on this project, Endong Zhu was doctoral student with the Department of Computer Science and Operations Research, Université de Montréal, while T.G. Crainic was the Natural Sciences and Engineering Research Council of Canada (NSERC) Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, by the partners of the Chair, CN, Rona, Alimentation Couche-Tard, the Ministry of Transportation of Québec, and by the Fonds de recherche du Québec - Nature et technologies (FRQNT).

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

Freight is moved for a large part by *consolidation-based* carriers: railroads, less-than-truckload motor carriers, container ships, regular and express-carrier services, etc. The fundamental idea of consolidation-based transportation is to group loads from different shippers, with possibly different origins and destinations, and to load them into the same vehicles for efficient long-haul transportation. The performance and profitability of such a system depend for a large part on efficient and coordinated terminal and long-haul transport operations, as well as an offer of services meeting the cost and quality criteria of its potential customers. The *service network design problem* is at the core of the tactical planning process addressing these issues and producing the transportation plan of the carrier for the contemplated planning period (Ahuja et al. 2005, Cordeau et al. 1998, Crainic 2000, Crainic and Kim 2007).

Rail carriers generally implement a double consolidation policy: loaded and empty cars are grouped into so-called blocks, which are then grouped again to make up trains. Cars with different origins and destinations being present simultaneously in the same terminal are thus sorted and grouped into a *block*, which is moved as *a single unit* by a series of trains until its destination, where it is broken down, the cars being either delivered to their final consignees or sorted for inclusion into new blocks. The associated operations and policies are denoted car *classification* and *blocking*, block *transfer* (from one train to another), and train *make up*. Tactical planning for freight rail carriers then aims to select the train services to operate over the contemplated schedule length (e.g., the week), together with their frequencies or schedules (timetables), the blocks that will make up each train, the blocks to be built in each terminal, and the routing of the cars loaded with the customers' freight using these services and blocks (empty-car movements are also considered). The corresponding service network design problem aims to build this plan making the most efficient use of the railroad's assets to achieve its economic and customer-service performance objectives.

A rich literature exists on models and methods addressing these issues, as reviewed in the surveys indicated above. Most of them, however, either address a single or a limited number of issues, or make significant simplifying hypotheses. To the best knowledge of the authors, in fact, no model currently available in the literature addresses in an integrated, comprehensive formulation all the tactical planning issues. Our goal is to answer this challenge and present an *integrated scheduled service network design* methodology for rail freight transportation.

The major contribution of this paper is twofold. We propose what we believe to be the first comprehensive modeling framework for key tactical planning decisions, integrating service selection and scheduling, car classification and blocking, train make up, and routing of time-dependent customer shipments. The framework is based on a three-layer space-time network representation of the associated operations and decisions, their

relations and time dimensions. Second, we propose an innovative solution methodology to address the large-size mixed-integer programming (MIP) formulation, which integrates exact and meta-heuristic principles. The proposed matheuristic is based on *slope scaling*, long-term memory-based perturbation strategies, and a new intensification mechanism, named *ellipsoidal search*, which thoroughly explores very large neighborhoods of elite solutions defined using information from the history of the search. The form of this restricted space inspired the name of the new procedure. Two variants of the matheuristic are presented: in the first, all potential blocks to be considered must be specified by the decision-maker, but the second includes a dynamic block-generation mechanism, thus allowing one to only consider a limited number of blocks to start with, since useful blocks will be generated as needed. Experimental results show the matheuristic to be efficient and robust, as it yields high-quality solutions for realistically-sized problem instances.

It must be emphasized that the interest of these contributions goes well beyond rail transportation. Our methodological framework encompasses the general field of service network design for consolidation-based transportation. It also opens interesting avenues for tackling multi-layer design problems found in logistics (e.g., in City Logistics), production and telecommunication applications. The proposed solution approach could thus serve as a basis for developing effective solution methods for such problems.

This paper is organized as follows. Section 2 describes the problem setting and reviews the related literature. The three-layer space-time network representation for the scheduled service network design problem is introduced in Section 3, while the MIP formulation is presented in Section 4. The proposed solution method is introduced in Section 5, experimental results are analyzed in Section 6, and we conclude in Section 7.

2 Problem Setting and Literature Review

We initiate the section with a brief description of the freight rail transportation system, its main components, operations, and tactical planning challenges, concluding by stating the particular problem addressed in this paper. The second part of the section is dedicated to a review of related literature.

Railroads are complex transportation systems where several major components interact and compete for resources. The infrastructure of the system is made up of a large number of terminals and rail tracks linking them. Most of these terminals are stations where demand originates and terminates. A much smaller number are denoted *yards* and are specially equipped to handle large quantities of cars, sorting and grouping them for long-haul transportation, as well as to make up and disassemble trains. The term *classification* (or *marshalling*) yard is also often used to emphasize the major car-handling role of these facilities.

Customer demand takes the form of a number of cars, of a type appropriate to the commodity that needs to be moved, to be shipped from an origin station to a destination one. The appropriate number of empty cars is delivered for loading to the customer (assuming it is connected to the rail network, or to a designated station, otherwise). Empty cars are generally delivered from a designated yard and the loaded cars are moved back to the same yard or to a different one as appropriate for the long-haul movement toward the destination that follows. Most origin and destination stations are on secondary lines. Even when this is not the case, and the station is located on a main trunk line, the movement of loaded and empty cars between stations and their respective designated yards is generally performed by local trains (also called feeder trains) whose scheduling is usually not within the scope of the tactical planning process designing the long-haul service network. We follow this practice in this paper and assume that all demands are specified at the appropriate origin and destination yards.

Movements of freight on the rail network are performed by train services. A *train* is composed of one or more engines providing power and a series of cars. Each train has a particular origin yard where it is made up and a destination yard where it completes its journey, delivers all the cars currently hauled, and liberates the engines. A number of intermediary stops may be included in its route, the train delivering and picking up blocks at each one.

When the volume shipped between two stations in the systems is significant (i.e., the equivalent of at least a full train) and regular, railroads run dedicated trains, non-stop from the origin to the destination of the freight. To take advantage of economies of scale, however, most shipments are handled according to consolidation principles, trains hauling groups of cars of different customers, with different origins and destinations. Then, schematically, cars at their origin yard are sorted, *classified* is the term generally used, and grouped with other cars with potentially different origins and destinations into particular blocks. The *block* is then handled as a single unit from its origin yard, where it is formed, to its destination yard, where it is taken apart, its cars being then either delivered to the respective consignees (the cars are at their final destination) or reclassified and blocked for the next part of their trip. Trains are thus made up of blocks and, when appropriate, it is blocks that are picked up and delivered at intermediate stops. Blocks may thus be *transferred* (*switched*) from one train to another. This double consolidation organization reduces car handling activities at yards.

Operations are constrained by the physical characteristics of the infrastructure and the operational policies of the railroad. Thus, for a given time period, classification-yard operations are limited by the yard capacity in terms of numbers of cars that may be classified, blocks that may be built, trains that may be made up or stop, and so on. Similarly, the capacity of the rail tracks limits operations with respect to the number of trains that may operate "simultaneously" on a given track segment, as well as to the total weight (length also, sometimes) a train may haul on the track segment. The length

and weight of trains are thus limited and lower and upper limits may be imposed on blocks.

Service network design addresses the integrated planning of these operations, and aims for an allocation of resources to activities that fulfills the economic and customer-service objectives of the railroad. Several decisions are intertwined in this process, which yields an operating (the terms "load" and "transportation" are also used) plan and schedule:

- Service selection and scheduling. A *service* corresponds to a train route and type that may be operated, where the route specifies the origin, destination, sequence of yards visited, and physical route among them, while the type includes the planned speed, capacity, and priority of the train. Scheduling is performed for a given *schedule length* (e.g., one week) to be repeated for a certain *planning horizon* (e.g., a season). Schedules may be indicative, e.g., the frequency is only specified assuming a more or less uniform distribution of departures over the schedule length. Alternatively, a timetable may be specified for each service indicating arrival and departure times for each yard on the service route.
- *Blocking* (and classification) specifying the car classification policies at each yard and the corresponding blocks to be built.
- *Train make up* giving the blocks the service hauls out of its origin yard and, eventually, the blocks it drops and picks up at intermediary stops.
- *Freight routing* specifying for each particular demand the itinerary used to transport its cars from origin to destination: the sequence of blocks and train services, and the corresponding yard operations. In this context, *demand* refers both to forecast customer requirements and to an estimation of the associated needs for empty car repositioning.

Service network design is thus a complex problem in most cases. It involves many selection (services, blocks, schedules) and routing (demand loads) decisions, each with network-wide impacts and all strongly and complexly linked in their economic, time, and space dimensions. An integrated model is required to address this problem in a comprehensive way. Most contributions in the literature, however, address a limited number of issues only.

Early contributions include the train routing and make up model of Assad (1980) and the train routing and scheduling model of Morlok and Peterson (1970). Service selection is often addressed in two steps, service routes and frequencies are first determined (e.g., Marín and Salmerón 1996, Goossens et al. 2004), train timetables being then constructed based on the resulting routing pattern (e.g., Brännlund et al. 1998, Caprara et al. 2002, 2006). Newman and Yano (2000) proposed a train scheduling model, which also assigned

containers to trains, for the rail portion of intermodal shipments. Huntley et al. (1995) developed a computerized routing and scheduling system for CSX Transportation, while Ireland et al. (2004) developed a planning system for Canadian Pacific Railway that brought together several separate procedures without building a comprehensive model.

Blocking has often been addressed as a separate problem, assuming that train services were already selected. Bodin et al. (1980) proposed one of the first such models, a non-linear MIP formulation. Newton et al. (1998) and Barnhart et al. (2000) formulated the blocking problem as a network design problem, with nodes and arcs representing classification yards and candidate blocks, respectively. No fixed costs were associated to blocks, but several capacity restrictions were considered to limit the number of blocks and the total volume of freight processed at each yard. A path-formulation and a branch-and-price solution approach (Barnhart et al. 1998) was proposed in the former paper, while a dual-based Lagrangian relaxation was used in the latter to decompose the problem into easier-to-address subproblems: a continuous multi-commodity flow problem and an integer block formulation that selected blocks satisfying yard capacity constraints (addressed by branch-and-cut). Ahuja et al. (2007) proposed a large neighborhood search algorithm for their mixed-integer blocking formulation aimed at addressing large problem instances.

Crainic et al. (1984) presented what was probably the first service network design model addressing simultaneously the selection of services and their frequencies, car classification and blocking, train make up, and freight routing. Crainic and Rousseau (1986) generalized the model for the tactical planning of consolidation-based multi-commodity multi-mode freight transportation systems. The model took the form of a path-based, nonlinear network design formulation accounting for congestion phenomena in yards and on rail tracks, as well as for trade-offs between operating and time-related costs. A heuristic solution method was used to address realistically-dimensioned problem instances derived from the case of a large North-American railroad. It was a "static" formulation, however, assuming a uniform distribution of departures and indirectly selecting blocks through nonlinear accumulation delay functions.

Haghani (1989) attempted to combine train routing and scheduling, make-up, as well as empty car distribution on a space-time network with fixed travel times and pre-specified traffic rules. A heuristic was used to address a somewhat simplified version of the model and illustrate the interest of integrated planning. The model proposed by Keaton (1989, 1992) aimed to determine which pairs of yards to connect by direct services, and whether to offer more than one train a day, as well as the routing of freight and the blocking of rail cars. The service network was made up of one network for each pair of yards in the system with positive demand. Links represented trains and connections in yards, as well as *a priori* determined blocking alternatives. Blocking was not a direct decision, however, integer train connections and continuous car flows representing the decision variables. Solutions were obtained by using a combination of

Lagrangian relaxation and various heuristics based on particular operation rules. Results were mixed. While the model was used to perform a number of analyses on relatively small systems, convergence difficulties were also reported. Gorman (1998a) proposed a model aimed at the design of a scheduled operating plan that followed the particular operation rules of a given railroad. Model simplifications had to be introduced to achieve a comprehensive mathematical network design formulation, but the solution method was innovative. A hybrid meta-heuristic, a tabu-enhanced genetic search, was used to generate candidate train schedules, which were evaluated on their economic, service, and operational performances. On relatively small but realistic problems, the meta-heuristic performed well and was used for strategic scenario analysis for a major North-American railroad (Gorman 1998b). All the previous contributions modeled the blocking process through classification costs, with no explicit blocking decision variables.

Andersen et al. (2009a,b), Pedersen and Crainic (2007), and Pedersen et al. (2009) added a new dimension to railroad tactical planning by introducing asset-management considerations into service network design models. The latter two contributions focused on the management of one asset type, engines, namely. A larger palette of issues was addressed in the first two papers, which also included scheduling aspects, cyclic schedules being constructed for services and assets. Classification and blocking issues were not addressed, however.

None of the previous contributions integrates all the major tactical planning issues into a single scheduled service network design formulation. Our objective is to address this challenge. The next two sections detail our modeling-framework proposal, which takes the form of a mixed-integer network design formulation on a cyclic space-time network that explicitly integrates decisions on service selection and scheduling, car classification and blocking, train make up, and traffic routing.

3 Three-Layer Space-Time Network

This section is dedicated to the presentation of the three-layer space-time network structure that we propose to describe the operations and tactical decisions of a rail freight transportation system. This structure is built starting from the rail network on which operations are performed, represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertices $v \in \mathcal{V}$ representing yards and directed links $e \in \mathcal{E}$ rail track sections.

A number of measures associated to the vertices and links of $\mathcal{G}$ represent limitations on the operations of the system. The volume of activities a classification yard may perform during any time period is limited by its physical characteristics, layout, number of tracks, engine or gravity-based sorting, etc., as well as available resources in terms of workforce, yard engines, and so on. To represent these limitations, we define the

classification capacity, u_v^{C}, as the number of cars that may be classified at yard $v \in \mathcal{V}$, and the *blocking capacity*, u_v^{B}, as the number of blocks that may be built at the yard. Assuming, for simplicity of presentation, that a blocking track is dedicated to each block being built, the latter measure is given by the number of blocking tracks of the yard. Similarly, the track section *service capacity* u_e represents the maximum number of trains that may be present on link $e \in \mathcal{E}$.

Time-dependent issues in network design are generally addressed through space-time networks representing the occurrences of objects, events, and decisions at particular time moments. We follow this approach in this paper, and propose a three-layer structure to represent adequately the double consolidation activities proper to railroad transportation, cars into blocks and blocks into trains, as well as the three types of rail flow movements, namely, train services, blocks, and cars. The structure is illustrated in Figure 1.

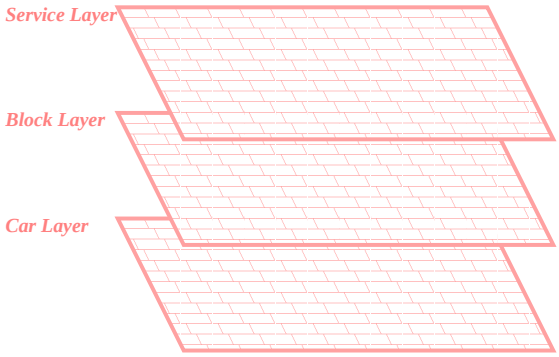

Figure 1: Three-Layer Space-Time Network Structure

The time dimension is the same in all layers. The schedule length is divided into $\mathbf{T}$ equal-length time *periods* starting at time $t = 0, \ldots, \mathbf{T} - 1$. The schedules are cyclic, that is, Period 1 follows Period $\mathbf{T}$. Yards are represented at each period through two nodes: an IN node where cars, blocks, and services arrive at the yard and, symmetrically, an OUT node from where they leave the yard. Consequently, the total number of nodes in each layer is equal to 2 × number of yards × number of periods. Denote $\mathcal{N}^{\mathrm{S}}$, $\mathcal{N}^{\mathrm{B}}$, and $\mathcal{N}^{\mathrm{C}}$ the sets of nodes in the service, block, and car layers, respectively.

The link set of the space-time network is the union of the link sets in each layer, plus the vertical links connecting them. Yard operations in all layers are represented through various instantiations of *handling* links between IN and OUT nodes, and *holding* links between two consecutive IN or OUT nodes. The service layer also includes links representing inter-yard train movements, defined between OUT and IN nodes of different yards at different periods. The attributes of nodes and links depend upon the layer they belong to and are specified later in the corresponding sections. The *temporal length* (or, simply, *length*) definition is, however, the same for all links: the number of time periods covered by the link. Vertical links have no duration. The up and down links

between two consecutive layers represent the result of the yard car $\leftrightarrow$ block and block $\leftrightarrow$ train consolidation/de-consolidation activities. The three layers are described in the following subsections, which also introduce the definitions and notation used to formulate the proposed tactical planning model. Concepts are illustrated on a simple rail network with four yards and four directed track sections shown in Figure 2.

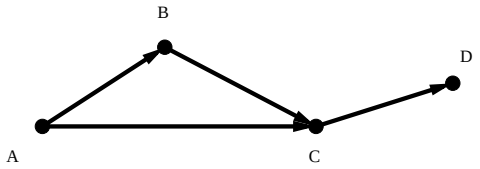

Figure 2: A Simple Rail Network

3.1 Service layer

Links in the *service layer* represent train operations and include *moving* and *stop* links. A moving link $a \in \mathcal{A}^{\mathrm{SM}}$ stands for a direct train movement from the OUT node of its origin yard at departure time to the IN node of its destination yard at arrival time (constant travel times are assumed for simplicity of presentation). The set $\mathcal{E}(a)$ includes the rail tracks defining the associated physical route. Figure 3 illustrates service-layer links defined on the rail network of Figure 2. Trains may move at different speeds on the same physical routes, which yields links of different lengths, e.g., the links (i_5, i_6) and (i_5, i_7) between yards B and C in Figure 3. Parallel moving links, e.g., with the same length between the same origin and destination nodes, represent train movements following different physical routes between the two yards with the same departure and transit times. Links a_1 and a_2 from node i_1 to node i_2 illustrate this case, where the former follows the route A $\rightarrow$ B $\rightarrow$ C, while the letter goes directly from A to C. A stop link $a \in \mathcal{A}^{\mathrm{SR}}$ connects an IN node to the OUT node of the same yard at the next period, representing the time spent by a train at an intermediate yard to drop and pickup blocks and prepare for departure, e.g., links (i_4, i_5) and (i_7, i_8) in Figure 3.

Let $\mathcal{S}$ be the set of possible services among which the selection is to be performed. A *service* $s \in \mathcal{S}$ is then defined as a path from $o(s) \in \mathcal{N}^{\mathrm{S}}$, the OUT node of its origin yard, to $d(s) \in \mathcal{N}^{\mathrm{S}}$, the IN node of its destination yard, through a set of moving, $\mathcal{A}^{\mathrm{SM}}(s) \subset \mathcal{A}^{\mathrm{SM}}$, and stop links. Service s_1 of Figure 3, which leaves yard A with destination D, with stops at B and C, and is made up of 3 moving and 2 stop links, $i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_7 \rightarrow i_8 \rightarrow i_9$, illustrates the definition. Each service is also characterized by a departure period, a (fixed) travel time, and a "fixed" cost c_s^{F} representing the cost of supplying the service (power, crew, and so on), as well as making up and disassembling the train. The capacity of the service s, u_s, indicates the maximum number of cars that may be hauled by a train.

Let $\mathcal{L}(s)$ be the set of sections of service s, each *section* $l \in \mathcal{L}(s)$ defining the service

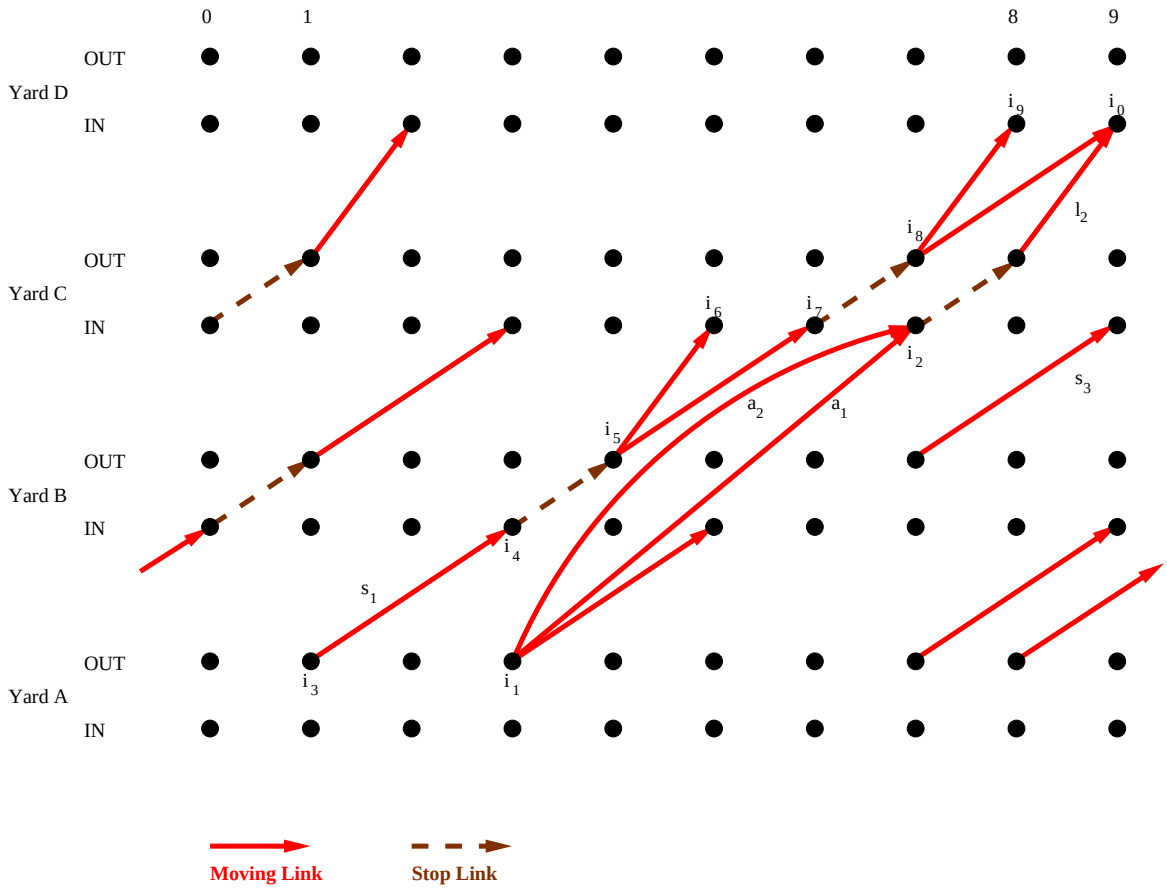

Figure 3: Service Layer

that s provides between two consecutive, but not necessarily adjacent, yards on its route. There are thus 6 sections for service s_1 in Figure 3: $(i_3 \rightarrow i_4)$, $(i_5 \rightarrow i_7)$, $(i_8 \rightarrow i_9)$, $(i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_7)$, $(i_5 \rightarrow i_7 \rightarrow i_8 \rightarrow i_9)$, and the service itself. For each section $l \in \mathcal{L}(s),\ s \in \mathcal{S}$, let $o(l), d(l) \in \mathcal{N}^{\mathrm{S}}$ be its origin and destination yards, respectively, $\mathcal{A}^{\mathrm{SM}}(l) \subseteq \mathcal{A}^{\mathrm{SM}}(s(l))$ its set of moving links, and $s(l)$ the indicator function giving the service it belongs to.

3.2 Block layer

Let $\mathcal{B}$ denote the set of potential blocks. A block $b \in \mathcal{B}$ is built at its origin yard $o(b) \in \mathcal{N}^{\mathrm{B}}$ and is moved until its destination yard $d(b) \in \mathcal{N}^{\mathrm{B}}$ by one or more services, service-to-service block transfers being performed at the intermediary yards on its route in the latter case. A fixed cost c_b^{F} is associated to building block b at its origin yard and performing the block transfers (if any). The length of the blocking track assigned to block b yields its capacity u_b in terms of the number of cars it may carry.

Block yard operations are represented through *transfer* and *transfer-delay* links in the *block layer*, collected in sets $\mathcal{A}^{\mathrm{BT}}$ and $\mathcal{A}^{\mathrm{BH}}$, respectively. A transfer link is a handling link representing the actual transfer procedure, i.e., taking the block off a service and attaching it to another one, and thus connects the IN and OUT nodes of the transfer yard with a length of one period. Transfer-delay links are holding links representing the time the blocking track is assigned at the origin yard - given as $h(b)$ periods (transfer-delay links) - as well as the time required to connect to the next service, to be determined by the model and algorithm. Arcs (i_7, i_8) and (i_6, i_7) in Figure 4 illustrate a transfer and a one-period transfer-delay link, respectively.

Block movements occur on sections in the service layer, the set $\mathcal{L}(b) \subset \mathcal{L}$ containing the sections used by block b. Figure 4 displays the projection of some of these links, e.g., set $\{(i_3, i_6), (i_8, i_9)\}$ representing the two sections of a block formed at yard A and departing from it at period 1 on one service, transferred and delayed one period at yard C, and arriving at destination yard D on a different service at time 8. The yard and long-haul block operations are connected by inter-block links, illustrated by the vertical links of Figure 5, which represent attaching/detaching blocks to/from services. An upward *attach-block* link connects an OUT node in the block layer to the corresponding OUT node in the service layer and represents the result of adding the block to the service at the given period. Symmetrically, a downward *detach-block* link connects IN nodes of the service and block layers, standing for taking off the block from its current service and making it available either for transfer or dismantling (if at destination). The route of a block is thus defined as a path in the block and service layers made up of $h(b)$ transfer-delay links at the origin yard, followed by a sequence of attach-block, service section, detach-block, transfer, and transfer-delay links describing its journey and the associated handling operations.

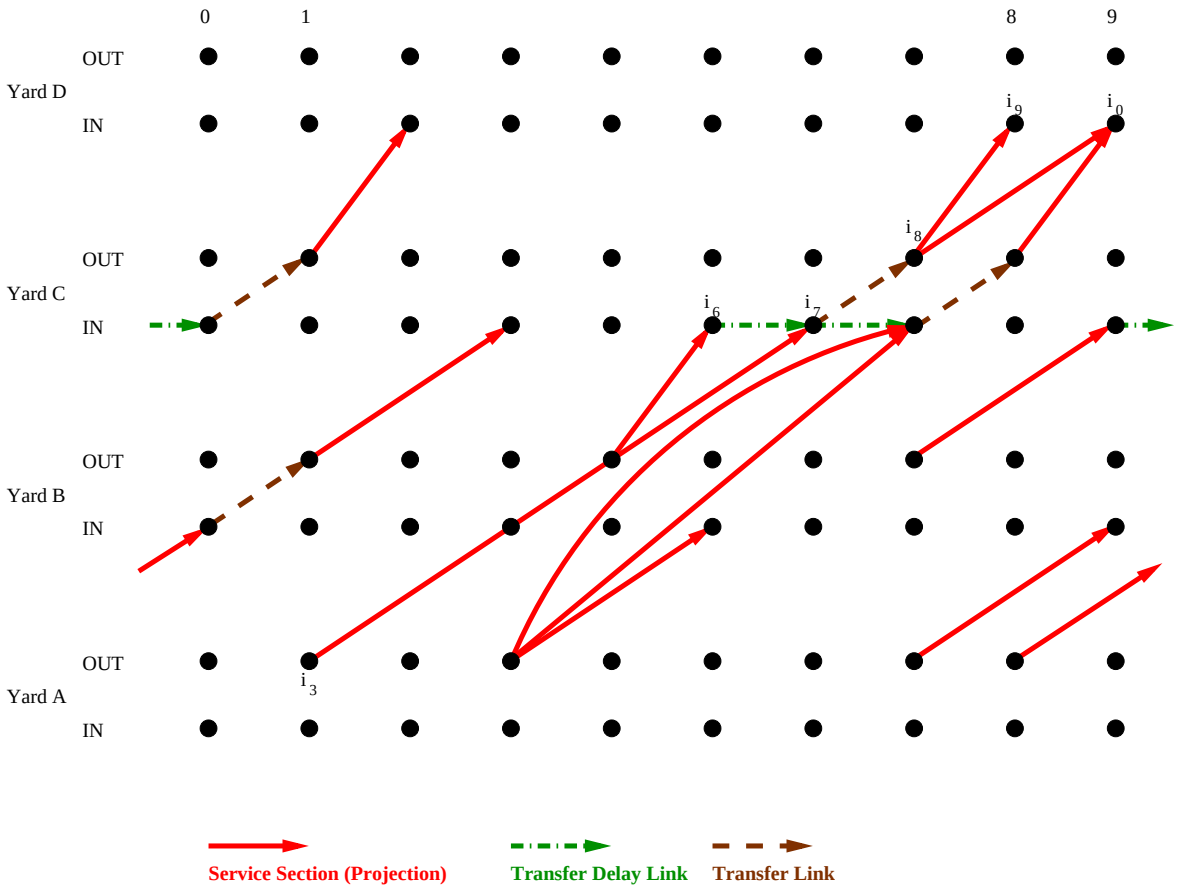

Figure 4: Block Layer

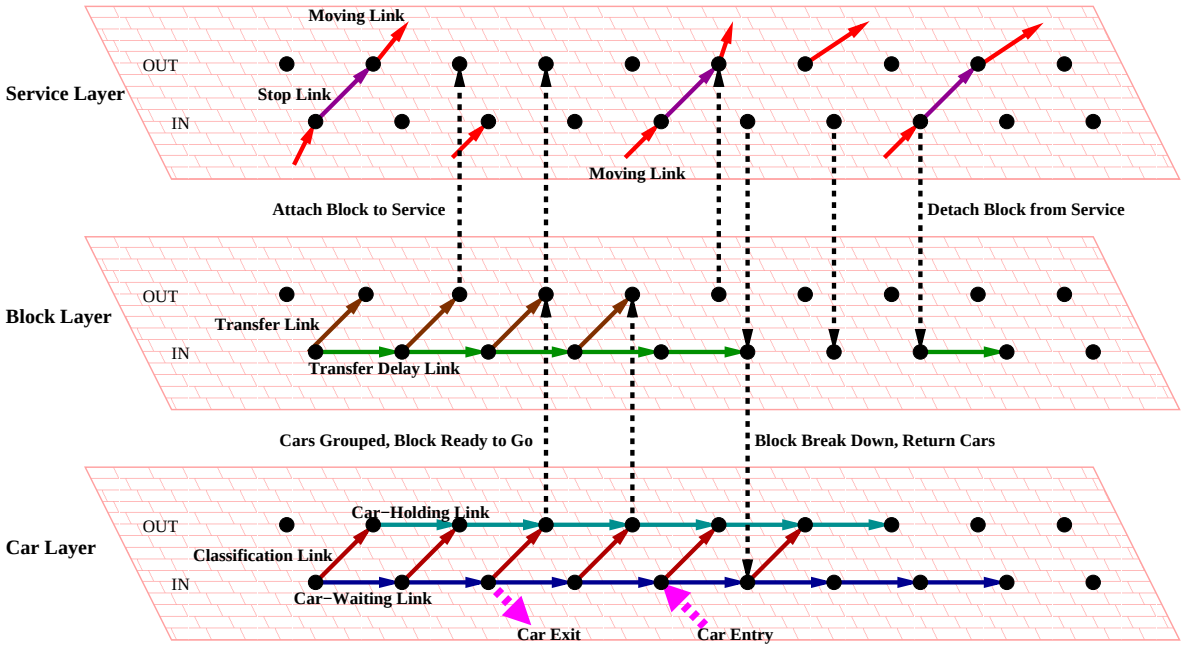

Figure 5: Main Link Types of the Three-Layer Space-Time Network

3.3 Car layer and itineraries

Similar concepts are defined for the *car layer*, illustrated in Figure 6, the links in this layer standing for the operations performed on cars in yards. There are two holding-type link sets, representing waiting in the receiving tracks to be classified and in the classification tracks for a sufficient number of cars to accumulate to build the block, respectively, and a handling one for the car classification operation. The three corresponding car *waiting*, *holding*, and *classification* link sets are denoted $\mathcal{A}^{\mathrm{CW}}$, $\mathcal{A}^{\mathrm{CH}}$, and $\mathcal{A}^{\mathrm{CC}}$, respectively. A waiting link joins two successive IN nodes of a yard, a classification link connects an IN node to the OUT node of the same yard at the next period, and a holding link connects two successive OUT nodes of a yard, as illustrated by links (i_1, i_2), (i_2, i_4), and (i_4, i_5), respectively. All these links have a length equal to one time period. The classification capacity u_v^{C} of a yard $v \in \mathcal{V}$ is associated in this representation to its classification links, i.e., $u_a = u_v^{\mathrm{C}}$ for $a \in \mathcal{A}^{\mathrm{CC}}$. The interplay between this limit and the flow of cars requiring classification at the yard determines for how long (i.e., the number of waiting links) cars will wait there.

Cars are moved in blocks in the block layer, Figure 6 illustrating the projection of some of these movement links. "Vertical" links thus connect the two layers to represent the result of the car $\leftrightarrow$ block activities at yards. Upward *group* links connect the OUT nodes in the car layer to their corresponding OUT nodes in the block layer and stand for the block formation activity. Symmetrically, a downward *break down* link is built from

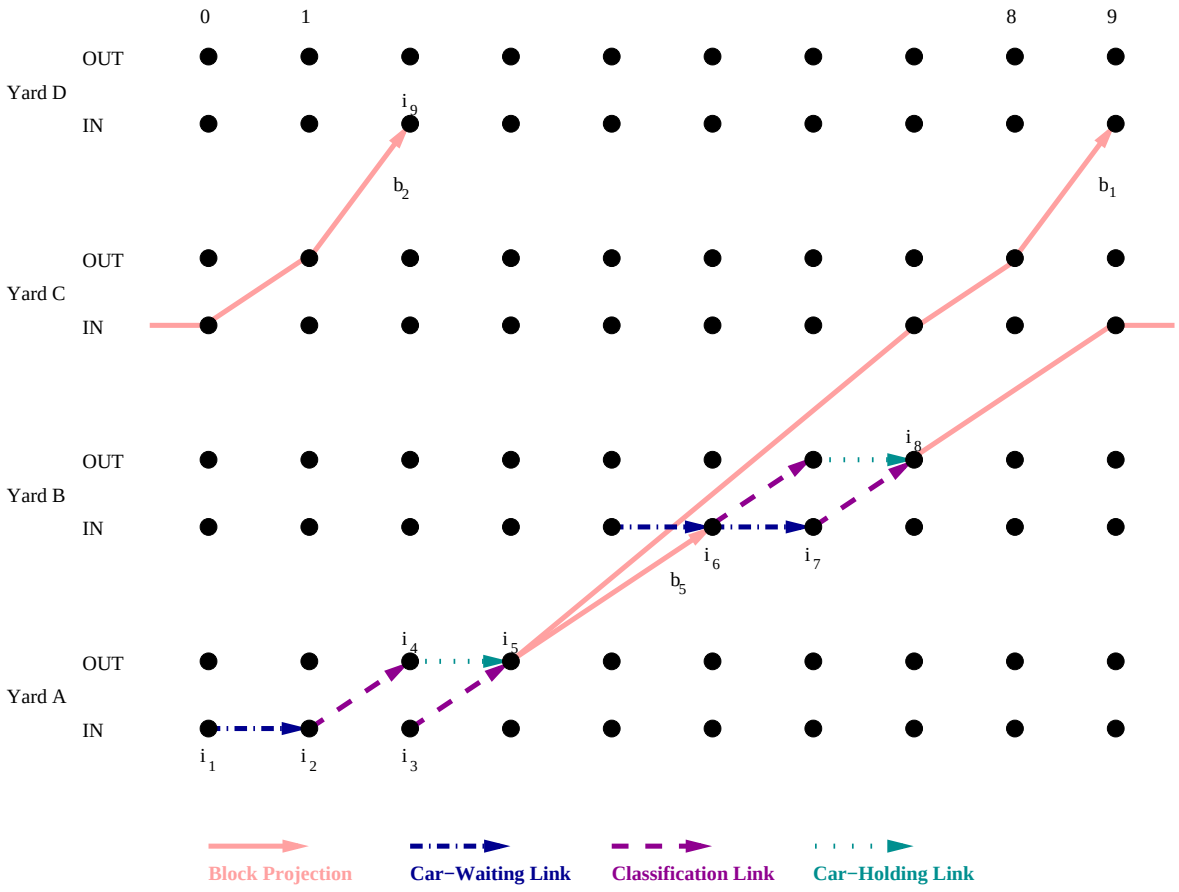

Figure 6: Car Layer

each IN node in the block layer to the corresponding IN node in the car layer to represent the activities related to blocks arriving at their destinations to be broken down into cars. These links are illustrated in Figure 5, together with those in and out of the IN nodes of the car layer representing the car reception from and delivery to customers.

An *itinerary*, i.e., the journey of a group of cars of a particular demand, is then a path in the three-layer space-time network between two IN nodes in the car layer representing their origin and destination yards. Once at the IN node of their origin yard, cars wait for classification in the waiting links, are classified through a classification link, and reach an OUT node, where they wait for accumulation on car holding links. Once the block is formed, the traffic goes up to the block layer, where the block journey to its destination proceeds as described earlier on. At destination, the block is unloaded and broken down, and the cars return to the car layer at the IN node of the associated yard and period. The journey ends if this node is the final yard of the demand. Otherwise, the cars go through classification and the block and service layers again until reaching their final destination.

The three-layer space-time network structure addresses the complete set of tactical planning activities and decisions. In particular, yard congestion phenomena, associated to higher workloads than the available capacity in a given period, are captured through the additional time spent by cars on car-waiting links (for yard classification congestion) and by blocks on transfer-delay links (for train congestion on line tracks). Selecting the best itinerary for each demand and the associated services, blocks, and schedule is the objective of the model presented in the next section.

4 Formulation

Let demand be specified by a set of *traffic classes* $\mathcal{P} = \{p\}$. Each traffic class $p \in \mathcal{P}$ is defined by an origin yard $o(p) \in \mathcal{V}$, a destination yard $d(p) \in \mathcal{V}$, a type of commodity $m(p)$, a quantity (number of cars) $w(p)$, a receiving time $r(p)$ at which cars become available at $o(p)$, and a due date by which cars must be received at $d(p)$. To simplify the notation on a cyclic network, we define the maximum *delivery time* $h(p)$ as the difference between the due date and the receiving time. We assume, as is usual in service network design settings, that $\mathcal{P}$ includes empty car repositioning requests.

Let $\mathcal{A}$ be the union of the links in all layers, and let c_{ap} represent the unit car cost for traffic class $p \in \mathcal{P}$ on link $a \in \mathcal{A}$. Notice that c_{ap} represents a hauling and time cost on inter-yard moving links, $\mathcal{A}^{\mathrm{SM}}$, a handling and time cost on car classification, $\mathcal{A}^{\mathrm{CC}}$, and block transfer links, $\mathcal{A}^{\mathrm{BT}}$, and a time cost on the service stop, $\mathcal{A}^{\mathrm{SR}}$, block transfer delay, $\mathcal{A}^{\mathrm{BH}}$, car waiting, $\mathcal{A}^{\mathrm{CW}}$, and car holding, $\mathcal{A}^{\mathrm{CH}}$, links.

We define three groups of decision variables:

- Service selection, $z_s = 1$ if service s is selected in the final design, 0 otherwise;
- Block selection, $y_b = 1$ if block b is built, 0 otherwise;
- Car flow distribution, x_{ap}, representing the number of cars of traffic class p on link $a \in \mathcal{A}$, with x_{bp} standing for the number of cars of traffic class p being part of block b.

Define the indicator function $\delta_{al}^{sb} = 1$ if the moving arc $a \in \mathcal{A}^{\mathrm{SM}}$ of service s belongs to section $l \in \mathcal{L}(b)$ of block b, and 0, otherwise. Define also $x_{asp} = \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}(b)} \delta_{al}^{sb} x_{bp}$, the total flow on moving link $a \in \mathcal{A}^{\mathrm{SM}}$ of service s, and $x_{sp} = \sum_{a\in\mathcal{A}^{\mathrm{SM}}} x_{asp}$, the total workload of service s with respect to the cars hauled on its moving links. The complete *Integrated Scheduled Service Network Design* formulation, *ISSND*, may then be written as follows:

$$\text{Minimize} \qquad \Phi = \sum_{s\in\mathcal{S}} c_s^{\mathrm{F}} z_s + \sum_{b\in\mathcal{B}} c_b^{\mathrm{F}} y_b + \sum_{p\in\mathcal{P}} \sum_{a\in\mathcal{A}} c_{ap} x_{ap} \tag{1}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}^+(n)} x_{ap} - \sum_{a\in\mathcal{A}^-(n)} x_{ap} = w_n^p \qquad \forall n \in \mathcal{N}^{\mathrm{C}}, \forall p \in \mathcal{P}; \tag{2}$$

$$\sum_{p\in\mathcal{P}} x_{bp} \le y_b u_b \qquad \forall b \in \mathcal{B}; \tag{3}$$

$$\sum_{p\in\mathcal{P}} x_{asp} \le z_s u_s \qquad \forall a \in \mathcal{A}^{\mathrm{SM}}, s \in \mathcal{S}; \tag{4}$$

$$y_b \le z_{s(l)} \qquad \forall b \in \mathcal{B}, l \in \mathcal{L}(b); \tag{5}$$

$$\sum_{p\in\mathcal{P}} x_{ap} \le u_a \qquad \forall a \in \mathcal{A}^{\mathrm{CC}}; \tag{6}$$

$$\sum_{b\in\mathcal{B}(v,t)} y_b \le u_v \qquad \forall v \in \mathcal{V}, \forall t \in \{1,\ldots,\mathbf{T}\}; \tag{7}$$

$$\sum_{s\in\mathcal{S}(e,t)} z_s \le u_e \qquad \forall e \in \mathcal{E}, \forall t \in \{1,\ldots,\mathbf{T}\}; \tag{8}$$

$$x_{ap} \ge 0 \qquad \forall a \in \mathcal{A}, \forall p \in \mathcal{P}; \tag{9}$$

$$y_b \in \{0,1\} \qquad \forall b \in \mathcal{B}; \tag{10}$$

$$z_s \in \{0,1\} \qquad \forall s \in \mathcal{S}. \tag{11}$$

The objective function (1) minimizes the total cost of the system for the schedule duration, that is, the total cost of selecting and operating services and blocks, waiting for operations, and moving the loaded and empty cars to satisfy demand.

Let $\mathcal{A}^+(n)$ and $\mathcal{A}^-(n)$ represent the sets of outward and inward links of node $n \in \mathcal{N}^{\mathrm{C}}$. Constraints (2) then enforce the car flow conservation in the car layer and, thus,

satisfaction of demand, where $w_n^p = w(p)$ if $n = o(p)$, $w_n^p = -w(p)$ if $n = d(p)$, and $w_n^p = 0$ otherwise.

Relations (3) and (4) are linking constraints enforcing the block and service moving-link car capacities, respectively, provided the corresponding blocks and services have been selected. Relations (5) are linking constraints as well, specifying that a block cannot exist unless all services providing the sections on its path have been selected.

Constraints (6) and (7) limit the yard car-classification and block-building workload at each period. Because building a block occupies a track for several ($h(b)$) periods, $\mathcal{B}(v,t)$ is the set of blocks being build simultaneously in yard v at period t. Similarly, constraints (8) enforce the limit on the number of trains that may simultaneously run on a link (track segment), $\mathcal{S}(e,t)$ standing for the set of such services for link $e \in \mathcal{E}$ in period t. Finally, constraints (9) - (11) specify the domain of each set of decision variables.

5 Solution Method

The model proposed above yields a mixed-integer formulation of large dimensions, both in the number of integer variables and the number of constraints. The problems belongs to the general class of Capacitated Multi-commodity Network Design (CMND) problems, which is known to be NP-hard (Magnanti and Wong 1984). Previous experience with CMND formulations showed that exact methods cannot efficiently address instances of meaningful dimensions and that meta-heuristics integrating large-scale neighborhoods are required (Ghamlouche et al. 2003, 2004). They also showed, however, the interest of combining heuristics and methods taking advantage of the properties of the mathematical formulation of the problem (Hewitt et al. 2010).

We introduce such a matheuristic for the ISSND starting from the slope-scaling (SS) idea of Kim and Pardalos (1999), strengthened by Crainic et al. (2004) and Kim et al. (2006). Applied to the CMND, SS iteratively solves a linear approximation of the formulation, the resulting flow distribution being then used to enhance the current approximation of the fixed costs. One thus approaches the optimum solution of the CMND by solving successive linear approximation problems, the terms of the objective function being updated recursively. The perturbation of the linearization factors based on long-term search memories helps the procedure move out of local optima of rather mediocre quality.

The challenge in addressing the ISSND is compounded by the two-layer structure of its design. The problem and formulation display two sets of design decisions and variables, respectively, locked in a double-sided combinatorial relation, a train being made up of a series of blocks, while each block corresponds to a sequence of train sections. The

main issues in defining a slope-scaling strategy for such a formulation are the selection of the set of design variables to linearize, the resolution of the resulting approximation problem, and the determination of feasible solutions to ISSND based on the solutions of the linearized approximation. The intuitive approach to the first issue is to apply the fundamental principle of SS and linearize both sets of design variables. We detail this basic SS strategy for ISSND in Section 5.1.

An alternative approach linearizes one set of design variables only. While less intuitive, as the resulting approximation problem is still a mixed-integer CMND formulation, this approach takes advantage of the fact that the high design layer – trains on services – makes up the backbone of the tactical plan, while the second design layer – blocks – may be viewed and generated as sequences of train sections. It thus linearizes the service design variables only, and applies a meta-heuristic generating blocks dynamically to address the resulting block-CMND problem. Section 5.2 details this dynamic SS procedure for ISSND.

Two procedures complete the proposed algorithm, a mechanism to perturb the linearization factors when the SS search stalls (Section 5.5), and a new intensification method, named *ellipsoidal search*, detailed in Section 5.4. The latter thoroughly explores the solution space, suitably restricted based on the characteristics of elite solutions and the long-term memories, around good solutions identified during the search and collected in a *reference set*.

At a glance, the overall procedure, displayed in Algorithm 1, starts by defining *R-ISSND*, a relaxed problem obtained by removing the capacity constraints on the design variables to be linearized. It then proceeds through search phases separated by perturbations of the linearization factors Λ. In each phase, a sequence of SS iterations finds a local best solution to the linearized formulation AP(Λ), possibly generating new blocks, extracts a feasible solution to R-ISSND by selecting the design arcs with positive flows, and gradually updates the linearization factors. The sequence stops when no further improvement is obtained, i.e., when the flow (and block in the dynamic-SS case) patterns of two consecutive iterations are the same, indicating a local optimum has been reached, or no improvement is observed in the value of the R-ISSND feasible solution for $I_{max}^{nonimprove}$ consecutive iterations (even though different solutions might have been obtained for the AP problem). When required, a feasibility-restoration procedure yields feasible ISSND solutions by solving a series of so-called *sliding problems* (Section 5.3). The ellipsoidal search is implemented before the perturbation of the linearizarion factors is triggered. The procedure repeats until it reaches either the maximum computing time T_{max} or the maximum number of SS iterations I_{max}.

Algorithm 1 Proposed matheuristic

Set initial values for linearization factors Λ;
Reference set = $\emptyset$;
Define R-ISSND;
while Stopping criteria not met **do**
  repeat
    Generate an approximation problem AP(Λ);
    Address AP(Λ):
      For Basic_SS: Multi-commodity capacitated network flow problem;
      For Dynamic_SS: Block-CMND by tabu search, dynamically generating new blocks;
    Extract a feasible solution to R-ISSND;
    Update Λ;
  until No further improvement is obtained
  Generate a series of feasible solution to ISSND by solving sliding problems (updating the reference set);
  Intensify the search based on the reference set through the ellipsoidal search;
  Perturb linear factors Λ based on long-term memories.
end while

5.1 Basic Slope Scaling

Define the relaxed problem, *R-ISSND*, by removing the block-building (7) and service-on-link (8) constraints of the ISSND formulation involving design variables only. Let β_s, β_{as}, $\alpha_b \in \Lambda$ be the linearization factors associated to each service s, moving link $a \in \mathcal{A}^{\mathrm{SM}}$ of service s (where the fixed cost of s is distributed over its moving links, i.e., each moving link gets a $c_s^{\mathrm{F}}/|\mathcal{A}^{\mathrm{SM}}(s)|$ cost), and block b, respectively. The approximation problem may then be defined as

$$\mathrm{AP}(\alpha,\beta) = \text{Minimize} \sum_{p\in\mathcal{P}}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}^{\mathrm{SM}}} \beta_{as}x_{asp} + \sum_{p\in\mathcal{P}}\sum_{b\in\mathcal{B}} \alpha_b x_{bp} + \sum_{p\in\mathcal{P}}\sum_{a\in\mathcal{A}} c_{ap}x_{ap} \qquad (12)$$

subject to car flow conservation (2) and handling (6) constraints, as well as the capacity restrictions on the numbers of cars hauled by services on their moving links (13) and grouped into blocks (14).

$$\sum_{p\in\mathcal{P}} x_{asp} \le u_s \qquad \forall a \in \mathcal{A}^{\mathrm{SM}}, s \in \mathcal{S}; \qquad (13)$$

$$\sum_{p\in\mathcal{P}} x_{bp} \le u_b \qquad \forall b \in \mathcal{B}. \qquad (14)$$

AP(α,β) is a multi-commodity capacitated network flow problem, which can be solved by the Simplex method. Yet, because AP(α,β) has to be solved repeatedly and given the

usually large dimension of the instances addressed, this approach may require excessive computing time (and memory). We therefore propose a heuristic based on the shortest augmenting-path algorithm. Define the residual capacity of a traffic-class itinerary as the minimum of the residual capacities of its classification links, blocks, and sections. Flows are then assigned to the lowest cost itinerary with positive residual capacity. Preliminary experiments (Section 6.2) showed that the heuristic generates very good solutions within much shorter computing times compared with a state-of-the-art solver.

Addressing AP(α, β) yields a flow pattern $\tilde{x}$, leading to a feasible design for R-ISSNF by selecting only blocks $\tilde{y}$ and services $\tilde{z}$ with positive car flows. In order for the approximation to adequately reflect the total cost, its linearized term $\sum_{p\in\mathcal{P}}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}^{\mathrm{SM}}}\beta_{as}x_{asp}+\sum_{p\in\mathcal{P}}\sum_{b\in\mathcal{B}}\alpha_b x_{bp}$ should equal the corresponding ISSND design cost $\sum_{b\in\mathcal{B}|\tilde{y}_b=1}c_b^{\mathrm{F}}+\sum_{s\in\mathcal{S}|\tilde{z}_s=1}c_s^{\mathrm{F}}$. We therefore update the linearization factors for the selected services, sections, and blocks according to (15) and (16). Initial values for the linearization factors are computed according to the fixed-cost/capacity ratio, often used to obtain initial solutions to network design problems (Kim and Pardalos 1999): $\alpha_b = c_b^{\mathrm{F}}/u_b,\ \forall b \in \mathcal{B}$ and $\beta_{as} = c_s^{\mathrm{F}}/|\mathcal{A}^{\mathrm{SM}}(s)|u_s,\ \forall a \in \mathcal{A}^{\mathrm{SM}},\ s \in \mathcal{S}$.

$$\alpha_b = c_b^{\mathrm{F}} / \sum_{p\in\mathcal{P}} \tilde{x}_{bp} \qquad \text{if } \tilde{y}_b = 1; \tag{15}$$

$$\beta_{as} = c_s^{\mathrm{F}} / \sum_{p\in\mathcal{P}} \tilde{x}_{asp} \qquad \text{if } \tilde{z}_s = 1. \tag{16}$$

Long-term memories are updated following each slope-scaling iteration (Section 5.5) and are used to guide the perturbation and ellipsoidal-search phases, as well as to adjust penalties for yard or track capacity violations. These memories track four service and block utilization measures continuously over all iterations; no re-initialization is performed following perturbation and ellipsoidal-search phases. The first is the *open frequency*, $f_i,\ i = \{s, b\}$, recording the number of iterations for which the service or block was selected, respectively. The other three measures are based on the flow carried by services and blocks: *average car flow*, $\bar{x}_i$, *peak car flow*, $\hat{x}_i$, and *flow variability ratio* $\mu_i = \bar{x}_i/\hat{x}_i$ ($\mu_i = 0$ when $\hat{x}_i = 0$), for $i = \{s, b\}$. The latter reflects the consistency of the loads a service or block carried, a value close to 1 indicating that, historically, car flows were close to the respective capacity, while the variability of the flow carried by the respective service or block increases significantly as the ratio approaches 0. Notice that the service car flow measures are computed using the total workload x_{sp} of the service.

5.2 Block-dynamic Slope Scaling

Only service design variables are linearized in this version, defining R-ISSND by removing the service-on-link (8) constraints of the ISSND formulation, and setting $\Lambda = \{\beta_s\}$. The

approximation problem then becomes (the initial set $\mathcal{B}$ is populated by building a block for each service section)

$$\text{AP}(\beta) = \text{Minimize} \sum_{b \in \mathcal{B}} c_b^{\text{F}} \cdot y_b + \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}} x_{sp} \beta_s + \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} c_{ap} \cdot x_{ap} \tag{17}$$

subject to constraints on car-flow conservation (2), car-to-block linking/capacity (3), car-classification capacity (6), and block-building at yards (7), as well as the train load capacity (18).

$$\sum_{p \in \mathcal{P}} x_{asp} \leq u_s \qquad \forall a \in \mathcal{A}^{\text{SM}}, s \in \mathcal{S}. \tag{18}$$

Define c'_{bp}, the surrogate flow cost for traffic class p and block b, as the sum of the unit flow costs $c_{ap} + \beta_s$ over the (moving) links a of the service sections making up the block b. The objective (17) then becomes

$$\text{AP}(\beta) = \text{Minimize} \sum_{b \in \mathcal{B}} c_b^{\text{F}} \cdot y_b + \sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} c'_{bp} x_{bp} + \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}^{\text{CW}} \cup \mathcal{A}^{\text{CC}} \cup \mathcal{A}^{\text{CH}}} c_{ap} \cdot x_{ap} \tag{19}$$

and AP(β) is a *block network design* CMND formulation. Addressing AP(β) yields a feasible flow distribution $\tilde{x}$ and a block design $\tilde{y}$. A feasible solution to R-ISSND may then be obtained by selecting services s used by blocks selected in $\tilde{y}$, i.e., $\tilde{z}_s = 1$ when one of the service sections of s is used by b and $\tilde{y}_b = 1$, $\tilde{z}_s = 0$, otherwise.

The linearization factors β_s are updated according to Equation (20) in order to satisfy condition (21) for the selected services. As for the long-term memory updates, the procedure proceeds as for the basic version (with the exception of f_b, of course).

$$\beta_s = c_s^{\text{F}} / \sum_{p \in \mathcal{P}} \tilde{x}_{sp} \tag{20}$$

$$\sum_{p \in \mathcal{P}} \tilde{x}_{sp} \beta_s = c_s^{\text{F}} \tilde{z}_s \tag{21}$$

We propose a cycle-based (Ghamlouche et al. 2003) tabu search (TS) heuristic with a dynamic block-generation mechanism to address the *block-CMND* formulation. The TS heuristic is based on the structure proposed by Zhu et al. (2009) for the direct service network design problem (see also Zhu 2011), enhanced to increase performance within a slope-scaling environment and to include the dynamic block generation capability.

The main idea of the cycle-based neighborhood is to improve a given solution by "closing" and "opening" sets of design arcs by redirecting flow on promising (in terms

of reducing both fixed and variable costs) cycles. In applying this idea to the AP(β), we integrate the generation of promising blocks into the neighborhood evaluation, that is, into the computation of such cycles. We now describe the main steps of the TS heuristic, which are summarized in Algorithm 2.

Algorithm 2 Tabu search heuristic with block generation

Require: An AP(β) solution $\tilde{x}$ and $\tilde{y}$;
Require: Current block set $\mathcal{B}$;
  while Stopping criteria not met **do**
    Set *MinCycle* = 0;
    for Each open block b in $\tilde{y}$ **do**
      Compute the flow value γ;
      Generate blocks between eligible OUT-IN node pairs through shortest-path algorithm on the (β, γ)-block layer;
      Add new blocks to set $\mathcal{B}$;
      Construct the γ-residual network;
      Identify the lowest-cost cycle for block b and update *MinCycle*
    end for
    Deviate flow on the *MinCycle*, update the block design and the long-term memory, and assign a tabu tag to blocks that changed their status (opened or closed);
    Distribute the car flow on the new block design, and restore feasibility (when needed);
    if Current solution is better than or close to the best overall AP(β) solution **then**
      Intensify the search;
    else
      if Improvement on current solution is negligible **then**
        Diversify the search;
      end if
    end if
  end while

Let $\mathcal{B}$ be the set of blocks available at the current iteration. The search space of the TS heuristic is made up of block designs in $\mathcal{B}$, and corresponding car flows, feasible for the approximated AP(β) problem. The neighborhood of a solution in this search space contains all solutions (block designs) that may be reached by closing a currently open block b, as well as closing and opening other blocks arrayed in a cycle including b. Clearly, one desires to identify the cycle that offers the best prospect to decrease the objective value of AP(β). Then, given the flow $\gamma = \sum_{p \in \mathcal{P}} x_{bp} > 0$ carried by b in the current block set $\mathcal{B}$, any cycle that permits to close b must offer the possibility to redirect at least γ units of flow. Moreover, among all such cycles that may be defined for block b, we are interested in the one that yields the largest cost decrease. An extended Bellman-Ford algorithm (Zhu et al. 2009) applied on a suitably defined γ-*residual network* searches this very large neighborhood and identifies least-cost candidate cycles.

The γ-residual network is based on the car-layer projection made up of block links

corresponding to blocks in $\mathcal{B}$, as well as car classification, waiting, and holding links. A block from an OUT-node i to an IN-node j yields possibly two block links:

- a *forward* arc $(i,j)^+$ if the residual capacity of the block, with respect to block and service capacities, is larger than γ; the cost of the arc equals the transportation cost for γ cars on the block, plus the fixed cost of the block when the block is currently closed; and
- a *backward* arc $(j,i)^-$ if a block is open and with a total flow larger or equal to γ; the negative cost (savings) of the arc equals the transportation cost for γ cars on the block, plus the fixed cost of the block if this is to be emptied by the move (i.e., the current flow $= \gamma$).

Forward arcs are built for all car waiting and holding links (these are uncapacitated), while backward links are built when the flow is at least γ cars. Classification arcs with positive flows are grouped and forward/backward arcs are added between each IN-OUT node pair to permit adding/subtracting flows on the in-yard paths of the corresponding station. The construction of the γ-residual network is illustrated in Figure 7.

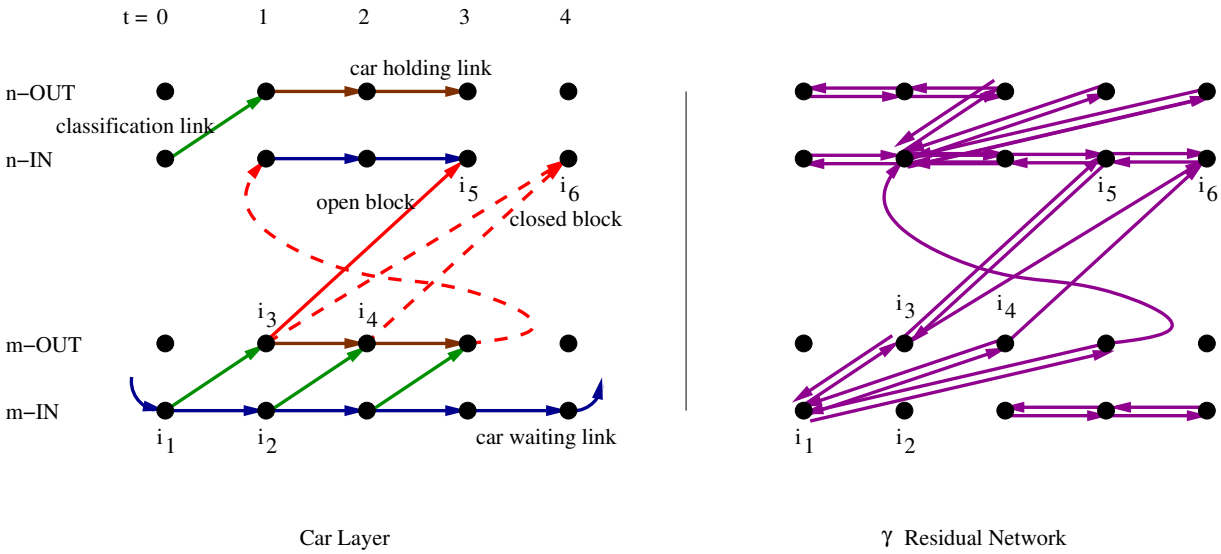

Figure 7: Construction of γ-Residual Network

The shortest-path algorithm identifies the lowest-cost cycle for block b, accounting for the temporal bounds imposed by the car flow conservation constraint enforcing timely delivery of demand. The cycle with the lowest cost, hopefully negative, among those of all blocks with positive flow is selected, and car flows are shifted along this cycle yielding a new flow distribution $\tilde{x}$ (an illustration of the move is given in the Electronic Companion). The block design $\tilde{y}$ is modified accordingly, yielding a new AP(β) solution.

New blocks are generated and added to $\mathcal{B}$ to increase the probability of identifying improving cycles. New blocks have therefore to be low-cost and able to carry at least γ units of flow. Because blocks are moved by trains on service sections, blocks are generated on a (β, γ)-block network with links corresponding to service sections with at least γ residual capacity, plus block-transfer and transfer-delay links. The unit costs of these links are computed as averages over the set of commodities and are given by Equations (22), (23) with the block fixed costs linearized by γ, and (24), respectively.

$$c'_l = \sum_{a \in \mathcal{A}^{\mathrm{SM}}(l)} \Big(\frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} + \beta_{s(l)} \Big) + \sum_{a \in \mathcal{A}^{\mathrm{SR}}(l)} \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} \qquad \forall l \in \mathcal{L}; \tag{22}$$

$$c'_a = \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} + \frac{c^F_a}{\gamma} \qquad \forall a \in \mathcal{A}^{\mathrm{BT}}; \tag{23}$$

$$c'_a = \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} \qquad \forall a \in \mathcal{A}^{\mathrm{BH}}. \tag{24}$$

A block is represented in the (β, γ)-block network by a path consisting of service sections connected by transfer and transfer-delay links. The set of such paths connecting an *eligible* OUT-IN node pair, i.e., the OUT and IN nodes are from different yards at different time periods, makes up a *block kernel* (or *parallel block set*). We then generate the least-cost path in each block kernel and add the corresponding blocks to set $\mathcal{B}$.

The lowest-cost cycle over all currently open blocks, *MinCycle* in Algorithm 2, identifies the neighboring solutions to which the algorithm moves by deviating γ units of aggregated flow along the cycle, updating the block design, and re-optimizing the flow distribution over the new design. The shortest augmenting path heuristic introduced in Section 5.1 is used in this case as well (this is different from Zhu et al. 2009, where the minimum cost flow problem was solved to optimality). Particular OD demands that cannot be feasibly delivered are sent on artificial links connecting the corresponding origin and destination nodes, and a restoration phase is implemented to regain feasibility. Restoration follows the same cycle-based idea, where cycles are constructed for artificial links with positive flow on graphs with γ values equal to the corresponding flow volumes.

Blocks that changed their status (from open to close or vice-versa) receive a tabu tag forbidding the reversal of this status for a tabu tenure randomly selected in a predefined integer range. In addition to this classical utilization of tabu memories to prevent cycling, we use them to prevent the search from exploring unfeasible regions. Each time a minimum-cost cycle is calculated, the blocks violating the block-building constraints are tagged with a tabu tenure, and we calculate the next non-tabu cycle. A long-term memory is also updated recording the frequency of blocks open in feasible solutions obtained by the tabu search.

Two processes are included to direct the search into/out of specific domains. *Intensification* is triggered each time a new best AP(β) solution is found or when the value

of the new current solution is close to the best AP(β) solution. Intensification proceeds similarly to the main tabu search but with a commodity-specific neighborhood, i.e., instead of working with aggregated flows, cycles of each commodity present on the block are evaluated with an assumption that the flows from other commodities are fixed. *Diversification* is called upon when there were no or only minor improvements in the last moves. To diversify the search, the long-term memory is exploited and a small number (e.g., 15%) of frequently-open blocks are forced to close. The flows are then redistributed on the new design, feasibility is restored, when required, as previously described, and the tabu search continues from this diverse solution.

To conclude this section, notice that the tabu search procedure with dynamic block generation offers a number of benefits. From a computational efficiency point of view, it facilitates a "continuous" resolution of the series of approximation problems, as the search at the current SS iteration may start from that of the previous iteration. Moreover, from an application point of view, the block generation procedure provides the means to consider application-specific restrictions, e.g., resource availability or characteristics of particular facilities, without the need to explicitly integrate them into the model formulation.

5.3 Restoring feasibility

A feasible solution to the R-ISSND may not be feasible for the original ISSND with respect to the block-building (7) and service-on-link (8) capacity constraints for the basic SS version, or only with respect to the latter for the block-dynamic version. The repair procedure we propose is based on the idea of "sliding" a number of selected services and blocks forward or backward in time, to distribute the workload in excess among "copies" of those services and blocks, which are assumed to be "good" because part of the current local optimum.

Define the *parallel neighbors* of a service (block) as the services (blocks) with the same route and transit time, but departing at the previous and the subsequent period. The *sliding problem* corresponding to a given solution is then defined as the ISSND model restricted to the services (and blocks for the basic version) selected in the current solution, plus their respective parallel neighbors.

The sliding-restricted problem has a relatively small number of integer variables and may be addressed efficiently by a MIP solver. We therefore continue performing the procedure once a feasible solution is obtained to further improve it. When the sliding procedure produces no feasible solution, we penalize the services and blocks violating the capacity constraints and continue the slope-scaling procedure. The penalization proceeds by increasing the corresponding linearization factors proportionally to the flow variability ratios, i.e., $\beta_{as} = \beta_{as}(1 + \mu_s)$ and $\alpha_b = \alpha_b(1 + \mu_b)$, respectively. The sliding procedure is

schematically displayed in Algorithm 3.

Algorithm 3 Sliding repair procedure

Require: Solution to R-ISSND;
  Define the corresponding sliding problem and solve exactly
  if Solution is feasible **then**
    Define the sliding problem corresponding to the new solution and solve;
    If the resulting solution has a lower total cost, **repeat** until no further improvement is obtained.
  else
    Penalize the services (and blocks) creating the infeasibility and continue the SS procedure.
  end if

5.4 Ellipsoidal search

The feasible ISSND solutions yielded either directly by addressing the approximation problem or, most often, as a result of the sliding repair procedure make up the *reference set*. We propose a new matheuristic, called *ellipsoidal search*, to intensify the search around elite solutions in this set, with a narrow focus on promising regions suggested by the history of the search and elite-solution attributes.

Ellipsoidal search selects two elite solutions, named *initial* and *guiding*, and uses them to define a restricted ISSND problem to be solved exactly by a MIP solver. The initial solution is then discarded from the reference set, but the pair of initial and guiding solutions is recorded in an *ellipsoidal-search memory* to avoid exploring solution regions already visited. When the MIP solver yields a solution different from the guiding solution, the new solution is added to the reference set, and the ellipsoidal search restarts; it stops, otherwise.

The selection of the two elite solutions is performed based on Path Relinking ideas (Glover et al. 2000) as applied to the CMND by Ghamlouche et al. (2004). We thus select the guiding solution as the best solution in the reference set, and the initial solution as the solution with the maximum Hamming distance, noted k, from the guiding one.

An *elite problem* is then defined as an ISSND with a k-distance cut (or local branching cut, Fischetti and Lodi 2003), and is solved exactly starting from the initial solution. This is equivalent to implicitly exploring the k-opt neighborhood of the initial solution defined as all solutions with at most k design variables with a different status compared to the initial solution. This neighborhood includes the guiding solution; it is illustrated in Figure 8.

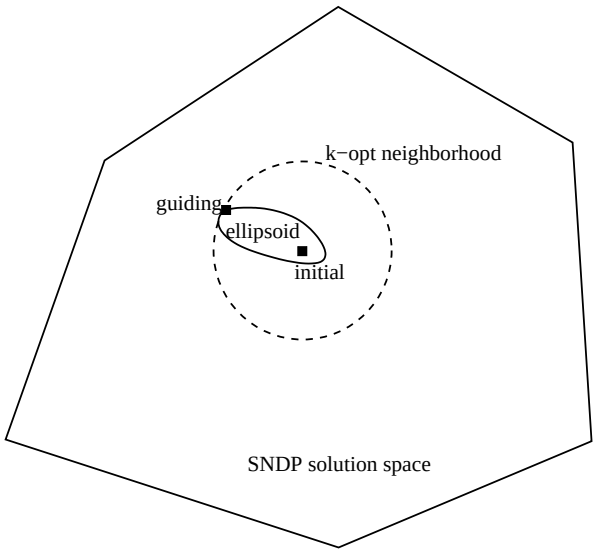

Figure 8: Ellipsoid within k-opt neighborhood

Given the large number of design variables proper to ISSND formulations, the k-opt neighborhood could still be quite large. We therefore further reduce the search space to a suitable ellipsoid, depicted in Figure 8, through variable selection and fixing based on the properties of the two elite solutions and the long-term search history.

All service design variables are selected for the ellipsoid definition, because each service may impact many blocks. With respect to blocks, we start with $\mathcal{B}_{ini} \cup \mathcal{B}_{gui}$, the blocks open in the initial and guiding solutions, respectively. We then add a restricted number (to keep the size of the formulation small) of other blocks with promising characteristics. The long-term memory guides this selection through the $\bar{x}_b/\hat{x}_b$ ratio, comparing the historical average and the maximal flow on block b, which gives a measure of the block showing up steadily in the solutions of previous SS iterations. Two strategies are defined and experimentally compared in Section 6.2. The first selects all blocks with a ratio superior to a given threshold λ. The second considers a maximum number of blocks to add, $|\mathcal{B}_{gui}| \times \varphi$, for a parameter φ, and selects them in decreasing order of the ratio. A number of the selected variables are then fixed to preserve attributes of current elite solutions. Thus, all blocks in $\mathcal{B}_{int} \cap \mathcal{B}_{gui}$ are fixed to open ($= 1$). Furthermore, flows common to both reference solutions are also fixed.

5.5 Long-term memory-based perturbation

The goal of the perturbation phase is to take the solution method out of the current local optimum and into a different region of the search space where slope scaling can again proceed.

Perturbation is performed by increasing or reducing the linearization factors according to a *diversification-intensification* strategy based on the search information stored in the long-term memories. Diversification follows the well-known meta-heuristic principle of moving the search into a not-yet explored region of the search space. In the present context, this translates into making less/more "interesting" a number of services (and blocks for the basic SS version) that currently are much/little used (with high/low open frequency), respectively. Intensification, on the other hand, aims to focus the search "around" good solutions, e.g., integrating components that appear consistently during previous iterations. It is thus performed by fixing services (and blocks) with high utilization (open) frequency.

To decide which blocks and services are to be involved, let $\bar{f}_{\text{ind}}$ and σ_{ind} stand for the mean and standard deviation of the opening frequencies for services (ind $= \mathcal{S}$) and blocks (ind $= \mathcal{B}$). Define $0 \leq \theta^{+}_{\text{ind}},\ \theta^{-}_{\text{ind}} \leq 1$, the thresholds indicating high and low frequency values, respectively, computed as $\theta^{+}_{\text{ind}} = \bar{f}_{\text{ind}} + \omega^{+}\sigma_{\text{ind}}$ and $\theta^{-}_{\text{ind}} = \bar{f}_{\text{ind}} - \omega^{-}\sigma_{\text{ind}}$, for given ω^{+} and ω^{-} parameters. A service (block) is then said to be frequently used when $f_{\text{ind}} \geq \theta^{+}_{\text{ind}}$, and rarely used when $f_{\text{ind}} \leq \theta^{-}_{\text{ind}}$.

The rewards or penalties used to perturb the linearization factors are governed by the flow variability ratios. Diversification is performed by adding high/low penalties to services and blocks with high/low ratios, which yields

- Services: $\beta_{sa} = \beta_{sa}(1 + \mu_s)$ if $f_s \geq \theta^{+}_{\mathcal{S}}$, and $\beta_{sa} = \beta_{sa}\mu_s$ if $f_s < \theta^{-}_{\mathcal{S}}$, $a \in \mathcal{A}^{\text{SM}}$, $s \in \mathcal{S}$;
- Blocks: $\alpha_b = \alpha_b(1 + \mu_b)$ if $f_b \geq \theta^{+}_{\mathcal{B}}$, and $\alpha_b = \alpha_b\mu_b$ if $f_b < \theta^{-}_{\mathcal{B}}$, $b \in \mathcal{B}$.

Symmetrically, to intensify, high/low rewards are given to services and blocks with high/low ratios:

- Services: $\beta_{sa} = \beta_{sa}(1 - \mu_s)$ if $f_s \geq \theta^{+}_{\mathcal{S}}$, and $\beta_{sa} = \beta_{sa}(2 - \mu_s)$ if $f_s < \theta^{-}_{\mathcal{S}}$, $a \in \mathcal{A}^{\text{SM}}$, $s \in \mathcal{S}$;
- Blocks: $\alpha_b = \alpha_b(1 - \mu_b)$ if $f_b \geq \theta^{+}_{\mathcal{B}}$, and $\alpha_b = \alpha_b(2 - \mu_b)$ if $f_b < \theta^{-}_{\mathcal{B}}$, $b \in \mathcal{B}$.

Too many consecutive intensifications may focus the search on few regions only and yield a poor exploration of the search space, while too many diversifications could result in continuously moving among distant solutions without a proper exploration of the regions visited. Two parameters, I^{inten}_{max} and I^{diver}_{max} are thus defined to guide the procedure and limit the number of consecutive intensification and diversification perturbation phases, respectively. When the new feasible solution for R-ISSND is better than (or equal to) the current best, intensification is called unless I^{inten}_{max} consecutive intensification moves were already performed, in which case, diversification is performed. On the other hand,

when that new solution is not better than the current one, diversification is called upon, unless I_{max}^{diver} is reached, in which case, intensification is called.

6 Computational Experiments

The performance of the proposed algorithms was computationally evaluated on a set of small to medium randomly-generated instances, designed to reflect actual application settings. Algorithms were coded in C++, and experiments were run on 2.4Hz CPU workstations with 16 GB of RAM, operating under Linux. We first describe the instances and the generation procedure, followed by the calibration of the various algorithms. Computational results with the best set of parameters are displayed and analyzed in the last subsection.

6.1 Instance generation

The instances were inspired by the setting of the main-line network of a major North-American railroad, but without including specific details to ensure the generality of the analysis.

Instances include a description of the physical network, a number of transportation demands, and all candidate services and blocks. To enumerate candidate services, service routes are combined with possible speeds and stops. The complete service list is obtained by instantiating all promising route-stop-speed combinations at each period. Sections from the service list are then connected by transfer links and transfer delay links in the block layer to generate potential blocks. Notice that, when methods such the one we propose are used in practice, the initial set of candidate services, and blocks for the Basic_SS method, is significantly more restricted, which speeds up the computations. In such cases, planning is generally started from the transportation plan used "the last time", enhanced with a number of potential new services reflecting changes in policy, demand, technology, regulation, and so on. For generality sake, we generated full potential service networks, but enforced a number of "realistic" conditions, e.g., no services and blocks with over-zigzagging routes, and no high-speed services with a high number of stops.

Two sets of fifteen instances each were generated. Both sets include instances with $5-10$ yards and $14-60$ track sections, but differ in the number of periods, services, and blocks. Set **S** instances have 7 periods, between 300 and 2674 services, and from 1855 to 279,230 blocks; the respective figures for set **L** are 10, 550 - 3050, and 8900 - 326,550. Instance parameters are listed in the Electronic Companion 8.2. The figures

in these tables emphasize the very large dimensions the formulation may reach in terms of service and block design variables. The number of flow variables and constraints also grows rapidly. To illustrate, consider that for 10 yards and 10 time periods, instances easily display over 300 million flow variables.

6.2 Calibration

Calibration aims to determine values for the major parameters of the proposed method such that it performs well over a broad range of instances. Calibration was performed on a set of random instances, called **C**, generated similarly to the other two, but different from them. The instances are described in the Electronic Compagnon 8.2, while detailed results supporting the decisions we report in this section are included in the Electronic Compagnon 8.3.

Three categories of parameters are important for the long-term memory-based perturbation phase ruling when to call on the procedure, $I_{max}^{nonimprove}$, the number of consecutive intensifications, I_{max}^{inten}, and diversifications, I_{max}^{diver}, and selection range of blocks and services for perturbation, α and φ. They were fixed first with the goal of striking a balance between search length in unpromising territory (high parameter values) and moving away too soon and missing an interesting solution (low values). Three values, 8, 10, and 15 were tested for $I_{max}^{nonimprove}$. Regarding I_{max}^{diver} and I_{max}^{inten}, setting them to 1 means alternating between intensification and diversification irrespective of possible improvement in the R-ISSND solution. Small values would provide a reasonable number of repetitions, and we tested 2 and 3. The selection range is managed by parameters ω^+ and ω^-. We choose to keep the low-frequency thresholds equal to the average opening frequencies $\bar{n}_{\mathcal{B}}$ and $\bar{n}_{\mathcal{S}}$, which are rather small due to the large number of potential blocks and services, and set $\omega^- = 0$. Two values were tested for the high-frequency thresholds, $\omega^+ = 0$, setting the thresholds to the average opening frequencies, and $\omega^+ = 1$, which restricts the range by adding the standard deviation to the previous value.

All 12 parameter combinations were evaluated and compared over the instances of set **C**. The results pointed to the combination ($I_{max}^{nonimprove}$, I_{max}^{diver}, I_{max}^{inten}, ω^+) = (10, 2, 2, 1), which stood out among all others and was used in all following experiments.

The next step consisted in evaluating the performance of the shortest augmenting-path heuristic in addressing the approximation problem generated during slope scaling. Experimental results indicated that, in this context, the heuristic outperforms the Simplex method, being more efficient in producing feasible solutions and enabling more slope-scaling iterations in a limited computing time. Flow distributions obtained by the augmenting path heuristic were not necessarily optimal, but they were sufficiently good to provide adequate starting points for generating better feasible ISSND solutions.

The last calibration step addressed the parameters of the ellipsoidal search. Two strategies were tested. The first, named s_1 in the following, is based on a threshold λ on the regularity of the flow on the blocks. Too small a threshold yields a large elite problem requiring a long solution time, while a large threshold avoids this, but may also miss many interesting solutions. We compared three values for λ, 15%, 25%, and 50%. The second strategy, s_2, limits the number of block variables to be considered into the ellipsoid through parameter φ. A value of 2 for φ generally corresponds to the $\mathcal{B}_{ini} \cup \mathcal{B}_{gui}$ set. We therefore tested 2, 5, and 10. Comparing the performance of the 6 settings, the second strategy outperformed the first. We therefore selected the second strategy with $\varphi = 5$, which achieved the highest score. Only a small portion of blocks are selected with this setting. For example, for instance c08, less than 200 block variables out of 75,000 were picked up in each elite problem.

The ellipsoidal-search phase plays an important role in the solution procedure, as shown by the results of an initial experimental phase performed with the Basic_SS procedure (see the Electronic Companion 9). The results show that in almost all cases (29 instances out of 30), including the ellipsoidal-search yielded better solutions than the basic slope scaling mechanism within the same solution time. The average improvement was 1.55% and 2.20% for the two instance sets, with a maximum improvement of over 6%.

6.3 Result analysis

We analyze the behavior of the methods we propose by comparing the performances of the block-dynamic slope scaling algorithm, identified as *DSS* in the following tables, the basic SS method, *BSS*, and the standard CPLEX MIP algorithm (version 12), which is also used as MIP solver embedded into the matheuristics. All methods were allowed to run for ten (10) hours of CPU time, except when the maximum number of SS iterations $I_{\max}$ was attained for the slope-scaling matheuristics. The initialization terminated after 300 iterations or 5 hours, whichever happened first.

Tables 1 and 2 display the computational results on instance sets **S** and **L**, respectively. CPLEX results are rounded to integer values for the sake of display clarity. The first column gives the instance name, while the CPLEX and OptGap columns display the objective value obtained by CPLEX and the corresponding optimality gap, respectively. Column BSS2CPLEX Gap displays the gap of the basic slope-scaling matheuristic relative to the CPLEX solution (negative figures indicate performance improvement), while the last column, DSS2BSS Gap, displays the gap of the block-dynamic SS method relative to the basic one.

The first observation is that, except for a few small instances, CPLEX is unable to find the optimal solution within the 10-hour time limit. Moreover, the efficiency decreases

Inst	CPLEX	OptGap	BSS2CPLEX Gap	DSS2BSS Gap
p01	75157	0.00%	0.00%	0.00%
p02	72403	0.00%	0.11%	0.13%
p03	78629	0.00%	0.13%	0.02%
p04	82784	0.00%	0.00%	0.00%
p05	98705	0.00%	0.00%	0.06%
p06	107663	2.78%	-0.08%	0.19%
p07	256530	35.29%	-28.46%	1.55%
p08	227342	36.07%	-26.86%	1.10%
p09	167011	26.76%	-20.30%	0.07%
p10	228857	36.23%	-30.75%	0.15%
p11	140270	10.69%	-5.38%	1.46%
p12	187967	17.61%	-10.93%	1.66%
p13	×	-	-	-0.07%
p14	×	-	-	-4.88%
p15	×	-	-	-2.06%

Table 1: Performance Results on Instance Set S

Inst	CPLEX	OptGap	BSS2CPLEX Gap	DSS2BSS Gap
p16	94777	1.76%	-0.11%	0.27%
p17	90010	2.54%	-0.51%	0.64%
p18	128097	0.61%	0.00%	0.32%
p19	79414	0.00%	0.00%	0.19%
p20	129305	0.00%	0.00%	0.00%
p21	103180	3.60%	-1.48%	1.51%
p22	217485	49.59%	-34.58%	1.26%
p23	218377	36.25%	-27.58%	0.36%
p24	×	-	-	1.23%
p25	205420	32.00%	-24.10%	0.20%
p26	×	-	-	1.52%
p27	246209	34.47%	-26.75%	1.35%
p28	×	-	-	-3.60%
p29	×	-	-	-1.79%
p30	×	-	-	-3.08%

Table 2: Performance Results on Instance Set L

dramatically with instance size, resulting in considerable optimal gaps for medium-sized instances and the impossibility to even find a feasible solution for larger instances (e.g., p13 with 10 yards and 60 track sections).

This performance further emphasizes the impressive behavior of the proposed methods compared to CPLEX, particularly as the instance dimensions grow. For the small instances for which CPLEX found the optimum, basic slope scaling with ellipsoidal search finds 5 optima out of 7 and solutions within 0.13% of the optimum for the others. In general, the proposed Basic_SS method obtains better solutions than CPLEX for 28 out of 30 instances tested, with a 23.57% average improvement for instances with 7 yards.

The Dynamic_SS matheuristic also performs very well. It also outperforms CPLEX, both on solution quality and computing times. Compared with the basic version, the block-dynamic procedure finds very close solutions within the same computing effort. Recall the additional effort required by the block-dynamic generation feature, which addresses a MIP formulation at each slope-scaling iteration. Thus, not surprisingly, the dynamic version performs somewhat less well on small and medium instances for the same computing time, with an average gap of 0.28% and 0.99% on the two instance sets. The dynamic version starts to outperform the basic one when the instance dimensions grow, improvements ranging from 0.07% for instance p13 to 4.88% for p14.

We completed the experimental analysis of the dynamic version by letting it run on a set of very large instances with 15 to 30 yards, 14 to 60 track sections, 7 time periods, and 1000 commodities (instances described in the Electronic Companion 8.2). The size of the network forbids the current implementations of CPLEX and Basic_SS to provide meaningful answers. The goal of these experiments therefore was to investigate the volume of blocks generated by the dynamic feature of the proposed matheuristic. The experiment showed that a small number of blocks is generated by the procedure. The final number, even for the largest instances in the set, is not larger than the numbers generated a priori for the moderate-size instances of sets L and C. This indicates the robustness of the algorithm we propose with the instance dimensions.

The proposed methodology is thus efficient in providing high-quality solutions to the scheduled service network design model with double consolidation we introduced for the tactical planning of rail freight transportation. The method achieves these results in significantly less computing time than a state-of-the-art commercial MIP solver, and identifies even better ones given additional search time. Moreover, the method yields feasible solutions even when instance dimensions grow and the solver fails due to time or memory limitations.

7 Conclusions

We proposed a first comprehensive modeling framework for key tactical planning decisions for rail freight transportation systems, integrating service selection and scheduling, car classification and blocking, train make up, and routing of time-dependent customer shipments. The framework is based on a three-layer space-time network representation of the associated operations and decisions, with their space-time-activity-economic relations. It provides, in particular, a flexible and efficient approach to addressing the two-tiered consolidation nature of railways, grouping cars into blocks and blocks into trains.

Our framework, as presented in this paper, focuses on the core decisions that rail companies must make. As it stands now, a number of constraints that can be encountered in practice in rail tactical planning are not explicitly accounted for, since these depend heavily on the specific context of each railway. Adding such constraints in the framework would limit its generality and make its presentation much heavier. It is important to note, however, that many of these constraints can be dealt with quite elegantly when one generates the initial set of services and blocks by simply "filtering" services or blocks that do not satisfy them, or when dynamically generating blocks.

We also proposed a new matheuristic solution methodology, integrating exact and meta-heuristic principles, to address the resulting scheduled service network design model, which takes the form of a large-size mixed-integer programming formulation. The proposed matheuristic combines slope scaling, enhanced by long-term memory-based perturbation strategies, an effective mechanism for dynamically generating blocks, and ellipsoidal search. The latter is a new intensification mechanism we introduce to thoroughly explore a very large neighborhood of an elite solution restricted using information from the history of the search. Experimental results show the matheuristic to be efficient and yield good-quality solutions for realistically-sized problem instances. The proposed methodology outperforms a state-of-the-art commercial solver, the improvements becoming increasingly spectacular with the instance dimensions.

As mentioned in the Introduction, the interest of the methodological framework that we propose goes well beyond rail planning and encompasses the general field of service network design for consolidation-based transportation. It also opens interesting avenues for tackling complex multi-layer design problems found in logistics, production and telecommunication applications. These applications, modeling avenues, and associated methodological challenges propose a rich research field to which we intend to contribute with our future works.

Acknowledgments

While working on this project, the first author was doctoral student with the Département d'informatique et de recherche opérationnelle, Université de Montréal, while the second author was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec, and by the Fonds de recherche du Québec - Nature et technologies (FRQ_NT)

References

Ahuja, R. K., C. B. Cunha, G. Sahin. 2005. Network models in railroad planning and scheduling. *Tutorials in Operations Research* **1** 54–101.

Ahuja, R. K., K. C. Jha, J. Liu. 2007. Solving real-life railroad blocking problems. *Interfaces* **37** 404–419.

Andersen, J., T. G. Crainic, M. Christiansen. 2009a. Service network design with asset management: Formulations and comparative analyzes. *Transportation Res. C* **17**(2) 397–207.

Andersen, J., T. G. Crainic, M. Christiansen. 2009b. Service network design with management and coordination of multiple fleets. *Eur. J. Oper. Res.* **193**(2) 377–389.

Assad, A. A. 1980. Models of rail networks: Toward a routing/make-up model. *Transportation Res. B* **14B** 101–114.

Barnhart, C., H. Jin, P. H. Vance. 2000. Railroad blocking: A network design application. *Oper. Res.* **48**(4) 603–614.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46**(3) 316–329.

Bodin, L. D., B. L. Golden, A. D. Schuster, W. Romig. 1980. A model for the blocking of trains. *Transportation Res. B* **14**(1-2) 115–120.

Brännlund, U., P. O. Lindberg, A. Nou, J.-E. Nilsson. 1998. Railway timetabling using Lagrangian relaxation. *Transportation Sci.* **32**(4) 358–369.

Caprara, A., M. Fischetti, P. Toth. 2002. Modeling and solving the train timetabling problem. *Oper. Res.* **50**(5) 851–861.

Caprara, A., M. Monaci, P. Toth, P. L. Guida. 2006. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics* **154** 738–753.

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

Crainic, T. G. 2000. Service network design in freight transportation. *Eur. J. Oper. Res.* **122** 272–288.

Crainic, T. G., J. A. Ferland, J.-M. Rousseau. 1984. A tactical planning model for rail freight transportation. *Transportation Sci.* **18**(2) 165–184.

Crainic, T. G., B. Gendron, G. Hernu. 2004. A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics* **10**(5) 525–545.

Crainic, T. G., K. H. Kim. 2007. Intermodal Transportation. C. Barnhart, G. Laporte, eds., *Transportation*, *Handbooks in Operations Research and Management Science*, vol. 14, chap. 8. North-Holland, Amsterdam, 467–537.

Crainic, T. G., J.-M. Rousseau. 1986. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Res. B* **20B**(3) 225–242.

Fischetti, M., A. Lodi. 2003. Local branching. *Math. Programming, Ser. B* **98** 23–47.

Ghamlouche, I., T. G. Crainic, M. Gendreau. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Oper. Res.* **51**(4) 655–667.

Ghamlouche, I., T. G. Crainic, M. Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Oper. Res.* **131**(1-4) 109–133.

Glover, F., M. Lagunai, R. Marti. 2000. Fundamentals of scatter search and path relinking. *Control and Cybernetics* **29**(3) 653–684.

Goossens, J. W., S. van Hoesel, L. Kroon. 2004. A branch-and-cut approach for solving railway line-planning problems. *Transportation Sci.* **38**(3) 379–393.

Gorman, M. F. 1998a. An application of genetic and tabu searches to the freight railroad operating plan problem. *Annals of Oper. Res.* **78** 51–69.

Gorman, M. F. 1998b. Santa Fe Railway uses an operating plan model to improve its service design. *Interfaces* **28**(4) 1–12.

Haghani, A. E. 1989. Formulation and solution of combined train routing and makeup, and empty car distribution model. *Transportation Res. B* **23B**(6) 433–452.

Hewitt, M., G. L. Nemhauser, M. W. P. Savelsbergh. 2010. Combining exact and heuristic approaches for the capacitated fixed charge network flow problem. *INFORMS J. Comput.* **22**(2) 314–325.

Huntley, C. L., D. E. Brown, D. E. Sappington, B. P. Markowicz. 1995. Freight routing and scheduling at CSX transportation. *Interfaces* **25**(3) 58–71.

Ireland, P., R. Case, J. Fallis, C. Van Dyke, J. Kuehn, M. Meketon. 2004. The Canadian Pacific Railway transforms operations by using models to develop its operating plans. *Interfaces* **34**(1) 5–14.

Keaton, M. H. 1989. Designing optimal railroad operating plans: Lagrangian relaxation and heuristic approaches. *Transportation Res.* **23B** 415–431.

Keaton, M. H. 1992. Designing railroad operating plans: A dual adjustment method for implementing Lagrangian relaxation. *Transportation Sci.* **26**(4) 263–279.

Kim, D., X. Pan, P. M. Pardalos. 2006. An enhanced dynamic slope scaling procedure with tabu scheme for fixed charge network flow problem. *Computational Economics* **27** 273–293.

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

Magnanti, T. L., R. T. Wong. 1984. Network design and transportation planning: Models and algorithms. *Transportation Sci.* **18**(1) 1–55.

Marín, A., J. Salmerón. 1996. Tactical design of rail freight network. Part I: Exact and heuristic methods. *Eur. J. Oper. Res.* **90**(1) 26–44.

Morlok, E. K., R. B. Peterson. 1970. Final report on a development of a geographic transportation network generation and evaluation model. *Processing of the Eleventh Annual Meeting*. Transportation Research Forum, 71–105.

Newman, A. M., C. A. Yano. 2000. Centralized and decentralized train scheduling for intermodal operations. *IIE Transactions* **32** 743–754.

Newton, H. N., C. Barnhart, P. H. Vance. 1998. Constructing railroad blocking plans to minimize handling costs. *Transportation Sci.* **32**(4) 330–345.

Pedersen, M. B., T. G. Crainic. 2007. Optimization of intermodal freight service schedules on train canals. Research Report CIRRELT-2007-51, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Montréal, Canada.

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

Zhu, E. 2011. Scheduled service network design for integrated planning of rail freight transportation. Ph.D. thesis, Université de Montréal, Canada. Available as Publication CIRRELT-2011-11, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, Canada.

Zhu, E., T. G. Crainic, M. Gendreau. 2009. Integrated service network design for rail freight transportation. Publication CIRRELT-2009-45, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, Canada.

8 E-companion - Annexes

This Annex first illustrates a cycle-based move in the dynamic-block generation tabu search. It then displays the characteristics of the instance sets, followed by detailed results of the parameter-calibration experiments, and of the study of the impact of various algorithmic components on the performance of the proposed algorithm.

8.1 Cycle-base move illustration

An illustration of the neighborhood move is given below Figure 9 shows a very simple AP(β) solution on the car-layer projection. 2 demands indicated by their O-D pairs must be satisfied. The information on each link/block is explained as (fixed cost, surrogate unit flow cost, current flow, flow capacity).

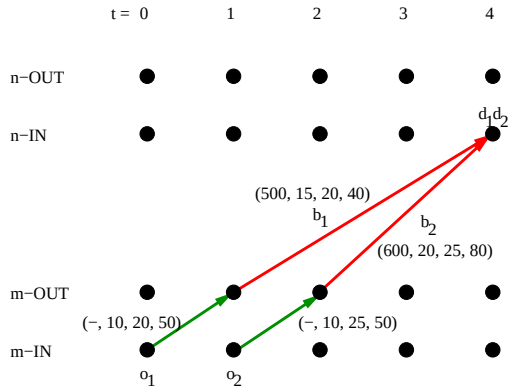

Figure 9: An AP(β) Solution on Car-Layer Projection

Residual values are collected in a residual value set, in this example, $\Gamma = \{20, 25\}$. For each residual value $\gamma \in \Gamma$, a γ-residual network is constructed. The lowest-cost cycles from each γ-residual network are compared, and the one with minimal cost is shown in Figure 10. The minimum-cost cycle has residual value $\gamma = 20$, and with total cost -380 (saving).

Following the minimum-cost cycle, the car flow on block b_1 is cleared by being moved to block b_2. The procedure reaches a neighbor block design where block b_1 is closed as shown in Figure 11.

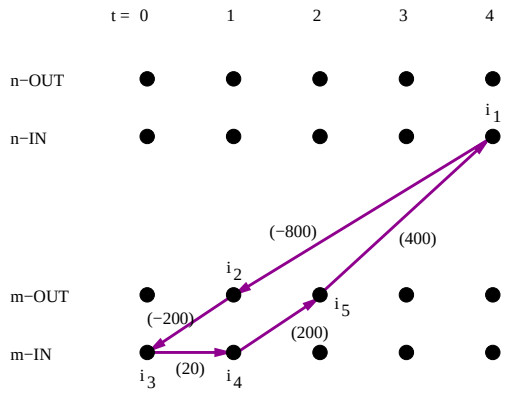

Figure 10: A Lowest-Cost Cycle in Residual Network

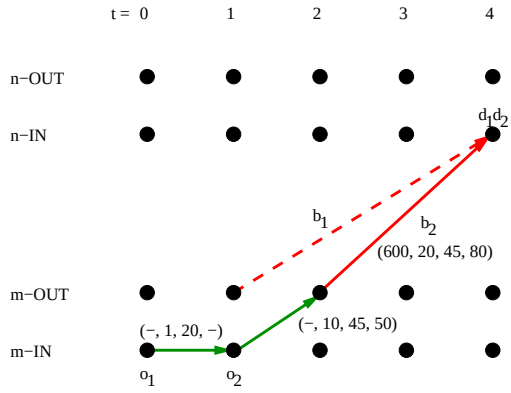

Figure 11: New AP(β) Solution on Neighbor Block Design

8.2 Problem Instances

Tables 3 and 4 display the characteristics of the instance sets S and L, respectively. Columns 2 to 4 display the numbers of blocks, services, yards, and track sections of the instance. Column "Time" gives the number of time periods, and the last column displays the number of commodities, i.e., origin-destination demands. Instances used for calibration are displayed in Table 5.

Table 6 display the characteristics of the large instances. The last three columns display the number of blocks generated initially and in total, and the value of the final solution of the Dynamic_SS procedure.

Inst	Block	Service	Yard	Track	Time	Demand
p01	1855	301	5	14	7	150
p02	2765	266	5	14	7	200
p03	2121	322	5	14	7	250
p04	3241	259	5	18	7	250
p05	1533	273	5	18	7	300
p06	7497	413	5	18	7	350
p07	39473	756	7	20	7	350
p08	29449	693	7	20	7	400
p09	19453	623	7	20	7	450
p10	18424	840	7	32	7	450
p11	9093	749	7	32	7	500
p12	11102	700	7	32	7	550
p13	140105	1834	10	60	7	600
p14	236628	2016	10	60	7	700
p15	279230	2674	10	60	7	800

Table 3: Instance Characteristics, Set S

8.3 Calibration

Table 7 displays the relative performance of the parameter combinations used to calibrate the long-term memory-based perturbation phase. Experiments were run on instance set **C**. Parameter combinations were weighted according to the solution quality, first place weighting 10 points, second 9, and so on until the last place weighted 1 point. Scores are then summed up over the problem instances and appear in Column 4. The first three columns present the corresponding parameter values.

Figures 12 and 13 illustrate the relative behavior of the shortest augmenting-path heuristic and the Simplex method when addressing the multi-commodity network flow formulation corresponding to the approximation problem generated by slope scaling. The figures display the evolution of the solution values R-ISSND and ISSND, respectively, in gray for the heuristic and in black for Simplex. The figures display results for instance c06 with parameter setting $I_{max}^{nonimprove} = 8$, $(I_{max}^{diver}, I_{max}^{inten}) = (2, 2)$ and $\omega^+ = 1$). The horizontal axis gives the computing time, and the vertical axis gives the solution value.

Table 8 displays the performance results of the parameter settings, three for each of the two strategies, for the ellipsoidal search. The same scoring system as above is used.

Inst	Block	Service	Yard	Track	Time	Demand
p16	8900	550	5	14	10	100
p17	4700	490	5	14	10	150
p18	3600	500	5	14	10	200
p19	1580	340	5	18	10	150
p20	2570	500	5	18	10	200
p21	8060	510	5	18	10	250
p22	46150	1230	7	20	10	200
p23	49920	1270	7	20	10	250
p24	50000	1080	7	20	10	300
p25	58930	1350	7	32	10	250
p26	55580	1140	7	32	10	300
p27	22900	1220	7	32	10	350
p28	491150	3080	10	60	10	500
p29	421370	3030	10	60	10	700
p30	326550	3050	10	60	10	900

Table 4: Instance Characteristics, Set L

Inst	Block	Service	Yard	Track	Time	Demand
c01	3350	450	4	10	10	90
c02	3340	460	4	10	10	120
c03	14970	1280	5	14	10	100
c04	9490	890	5	14	10	150
c05	27350	1200	5	14	10	200
c06	5790	810	5	18	10	120
c07	19970	1590	5	18	10	150
c08	75960	2550	5	18	10	200
c09	106310	2090	7	20	10	150

Table 5: Instance Characteristics, Set C

9 Performance of algorithmic components

This section of the Annex is dedicated to a brief study of the impact of the algorithmic components on the performance of the proposed matheuristic. The study was performed on an initial implementation of the Basic_SS procedure.

In the following tables the solution values for all procedures were rounded to integers for the sake of display. The column headers SS+LMP, Basic_SS, and CPLEX stand for the basic slope scaling mechanism of the Basic_SS procedure, the complete basic slope scaling with ellipsoid search, and the MIP solver of CPLEX, respectively.

Tables 9 and 10 display the computational results on instance sets **S** and **L**, respec-

Inst	Service	Yard	Track	Time	Demand	Block (ini)	Block (final)	DSS
x01	3654	15	60	7	1000	10031	92092	1003753
x02	4816	15	90	7	1000	13587	73717	868602
x03	4116	20	60	7	1000	10969	87514	1101111
x04	6048	20	80	7	1000	16212	62587	1071811
x05	9779	20	120	7	1000	30940	41538	1103703
x06	7434	25	90	7	1000	19978	45143	1155523
x07	13342	25	120	7	1000	38339	44366	1137879
x08	14896	25	150	7	1000	44653	49399	1089380
x09	14784	30	135	7	1000	41258	49595	1428480

Table 6: Instance Characteristics and Results, Set XL

$I_{max}^{nonimprove}$	$(I_{max}^{diver}, I_{max}^{inten})$	ω^+	Score
8	(2, 2)	0	33
8	(2, 2)	1	49
8	(3, 3)	0	34
8	(3, 3)	1	46
10	(2, 2)	0	36
10	(2, 2)	1	67
10	(3, 3)	0	45
10	(3, 3)	1	49
15	(2, 2)	0	51
15	(2, 2)	1	61
15	(3, 3)	0	39
15	(3, 3)	1	59

Table 7: Perturbation Parameter Calibration

tively, for CPLEX (10 hours CPU time) and the basic slope scaling mechanism after 1000 iterations and 10 hours CPU time. Solution times are displayed in CPU seconds. Characters $\times$ and "t" indicate no feasible solution found and time limit reached, respectively. Columns OptGap display the optimality gap of CPLEX, while columns CplGap display the relative gap of the proposed algorithm to the best solution of CPLEX, when available, or to its lower bound, otherwise.

The first observation is that, except for a few small instances, CPLEX is unable to find the optimal solution within the 10-hour time limit. Moreover, the efficiency decreases dramatically with instance size, resulting in considerable optimal gaps for medium-sized instances and the impossibility to even find a feasible solution for larger instances (e.g., p13 with 10 yards and 60 tracks).

The second is that even the simple slope scaling mechanism is very close to the optimum for small instances with an average optimality gap of 1.42% for p01 to p06 and an average of 50% reduction in computing time. On larger instances, the basic

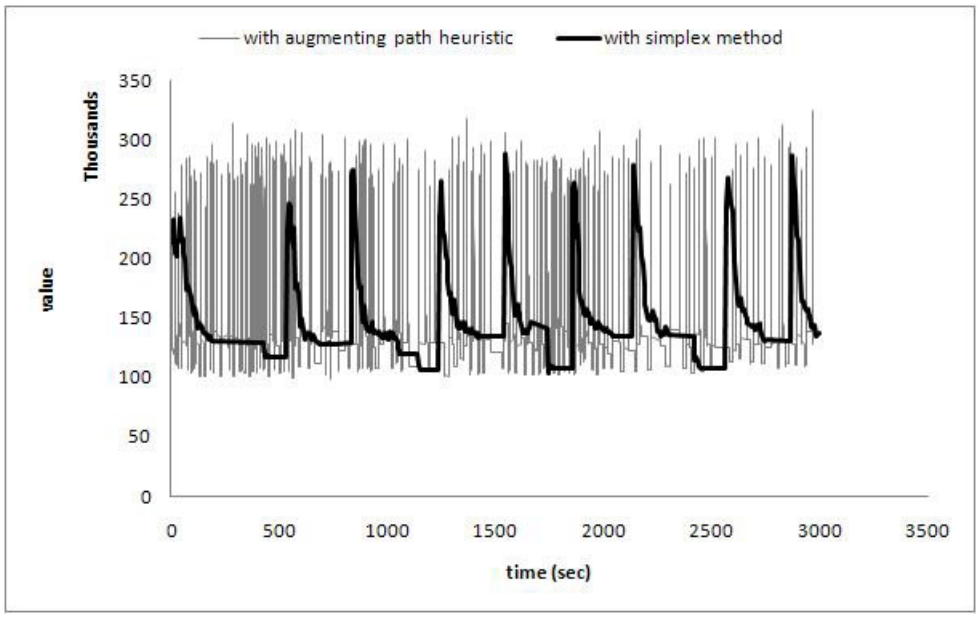

Figure 12: Comparison of R-SNDP evolutions

$s_1, \lambda = 15\%$	$s_1, \lambda = 25\%$	$s_1, \lambda = 50\%$	$s_2, \varphi = 2$	$s_2, \varphi = 5$	$s_2, \varphi = 10$
71	76	75	74	81	80

Table 8: Ellipsoidal Search Parameter Calibration

slope scaling mechanism performs impressively, outperforming CPLEX in both solution quality and time. Thus, for example, the mean improvement for instances p07 - p12 is 18.17% within 26.70% of the CPLEX computing time. Improvements of over 30% are observed for several instances and slope scaling is able to address larger instances than CPLEX. These improvements are obtained after 1000 iterations and are enhanced when longer computing times are allowed. A better solution was thus obtained for 16 out of 24 instances after 10 hours of computing. Then, compared to CPLEX solutions obtained with the same computational effort, slope scaling achieved an average gap of 0.88% on instances with 5 yards (p01-p06, p16-p21), and an average improvement of 21.58% on all the other, larger, instances.

Results of the complete Basic_SS matheuristic, integrating slope scaling and ellipsoidal search, for 10 hours of computing time, are displayed in Tables 11 and 12. The last two columns display the relative gap of the complete method to SS+LMP and CPLEX, respectively. These results underscore the important role of the ellipsoidal-search phase in the solution procedure. In almost all cases (29 instances out of 30), including the ellipsoidal-search yielded better solutions than slope scaling within the same solution time. The average improvement is 1.55% and 2.20% for the two instance sets, with a maximum improvement of over 6% for instance p24.

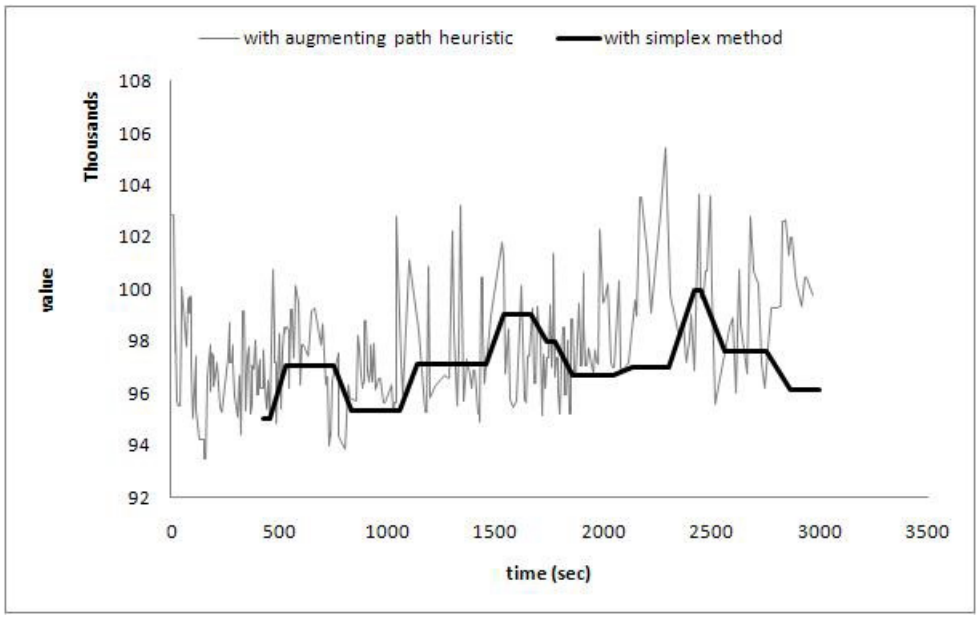

Figure 13: Comparison of ISSND evolutions

Inst	CPLEX	Time	OptGap	SS+LMP	CplGap	Time	TimeGap	SS+LMP	CplGap
		sec.		1000 iter.		sec.		10h	
p01	75157	371	0.00%	75667	0.68%	266	-28.39%	75347	0.25%
p02	72403	18281	0.00%	74228	2.52%	410	-97.76%	73446	1.44%
p03	78629	447	0.00%	79616	1.25%	345	-22.81%	79202	0.73%
p04	82784	9893	0.00%	83462	0.82%	688	-93.04%	83462	0.82%
p05	98705	703	0.00%	100173	1.49%	963	36.99%	99651	0.96%
p06	107663	t	2.78%	109542	1.74%	1042	-97.11%	109502	1.71%
p07	256530	t	35.29%	187930	-26.74%	4154	-88.46%	187930	-26.74%
p08	227342	t	36.07%	171912	-24.38%	4005	-88.88%	170525	-24.99%
p09	167011	t	26.76%	135343	-18.96%	7830	-78.25%	134790	-19.29%
p10	228857	t	36.23%	161141	-29.59%	2891	-91.97%	161141	-29.59%
p11	140270	t	10.69%	140845	0.41%	2782	-92.27%	136806	-2.47%
p12	187967	t	17.61%	169669	-9.73%	t	0.00%	169669	-9.73%
p13	×	-	-	216927	-	t	-	216927	-
p14	×	-	-	201368	-	t	-	201368	-
p15	×	-	-	229553	-	t	-	229553	-

Table 9: Basic Slope Scaling Results on Instance Set S

Inst	CPLEX	Time sec.	OptGap	SS+LMP 1000 iter.	CplGap	Time sec.	TimeGap	SS+LMP 10h	CplGap
p16	94777	t	1.76%	96345	1.65%	126	-99.65%	94944	0.18%
p17	90010	t	2.54%	92495	2.76%	203	-99.43%	91432	1.58%
p18	128097	t	0.61%	130037	1.51%	304	-99.16%	129256	0.90%
p19	79414	7862	0.00%	79566	0.19%	278	-96.46%	79563	0.19%
p20	129305	533	0.00%	131273	1.52%	343	-35.61%	130413	0.86%
p21	103180	t	3.60%	105144	1.90%	1006	-97.21%	104138	0.93%
p22	217485	t	49.59%	146270	-32.75%	1399	-96.11%	146270	-32.75%
p23	218377	t	36.25%	165325	-24.29%	1734	-95.18%	165325	-24.29%
p24	×	-	-	178798	-	3350	-	178798	-
p25	205420	t	32.00%	163591	-20.36%	3162	-91.22%	160778	-21.73%
p26	×	-	-	149691	-	3317	-	149691	-
p27	246209	t	34.47%	187950	-23.66%	1900	-94.72%	186722	-24.16%
p28	×	-	-	273842	-	t	-	273842	-
p29	×	-	-	264326	-	t	-	264326	-
p30	×	-	-	326381	-	t	-	326381	-

Table 10: Basic Slope Scaling Results on Instance Set L

Inst	Basic_SS	SS+LMP Gap	CplGap
p01	75157	-0.25%	0.00%
p02	72483	-1.31%	0.11%
p03	78731	-0.59%	0.13%
p04	82784	-0.81%	0.00%
p05	98705	-0.95%	0.00%
p06	107576	-1.76%	-0.08%
p07	183527	-2.34%	-28.46%
p08	166283	-2.49%	-26.86%
p09	133110	-1.25%	-20.30%
p10	158486	-1.65%	-30.75%
p11	132723	-2.98%	-5.38%
p12	167424	-1.32%	-10.93%
p13	212204	-2.18%	-
p14	195208	-3.06%	-
p15	228770	-0.34%	-
Avg		-1.55%	

Table 11: Basic_SS with Ellipsoidal Search Performance on Instance Set S

Inst	Basic_SS	SS+LMP Gap	CplGap
p16	94676	-0.28%	-0.11%
p17	89554	-2.05%	-0.51%
p18	128097	-0.90%	0.00%
p19	79414	-0.19%	0.00%
p20	129305	-0.85%	0.00%
p21	101656	-2.38%	-1.48%
p22	142283	-2.73%	-34.58%
p23	158157	-4.34%	-27.58%
p24	167934	-6.08%	-
p25	155919	-3.02%	-24.10%
p26	145052	-3.10%	-
p27	180356	-3.41%	-26.75%
p28	271156	-0.98%	-
p29	257230	-2.68%	-
p30	326575	0.06%	-
Avg		-2.20%	

Table 12: Basic_SS with Ellipsoidal Search Results on Instance Set L