Integrated Service Network Design for Rail Freight Transportation

Endong Zhu
Teodor Gabriel Crainic
Michel Gendreau

November 2009

CIRRELT-2009-45

Integrated Service Network Design for Rail Freight Transportation

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

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

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

[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, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. The research aims to produce a good operating plan at the tactical level for freight rail transportation. The service network design problem is studied, as we consider the scheduled service selection, blocking policy, train make-up policy and car distribution together. A 2-layer time-space network is proposed to model the car flow as well as the decisions on both blocks and services. The mixed-integer programming model is difficult and a tabu search heuristic is developed to provide good feasible solutions within reasonable solving effort. Numerical results show the proposed method is robust, and capable to provide near-optimal solutions for rather large instances.

Keywords. Service network design, freight transportation, rail application, capacitated multi-commodity network design.

Acknowledgements. 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 and by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec.

[†] This paper received the first prize in the 2009 Student Research Paper Contest of Railway Applications Section (RAS), a subdivision of INFORMS.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2009

1 Introduction

As one of the major components of the logistic chain, freight rail transportation is distinguished by its economics and mass volume. It also facilitates international trade and economic growth in most countries.

To distribute all kinds of commodities, railways need to provide services so that transportation demands from customers can be met. A *service* is carried out by a train which is characterized by its origin, destination, route, intermediate stops, speed, capacity, and schedule. Services without any intermediate stops are called *direct services* or *non-stop services*. A *service plan* maintains a list of services to be provided. Generally, a service plan is based on a planning horizon (usually 1 or 2 weeks), and is cyclic over a certain period of time (4 to 6 months) with the probability of slight occasional changes.

In order to ensure the proper implementation of the proposed services, extra policies and configurations are adopted to manage internal operations, which have great impacts on the services' performance. The service plan and interior policies thus form an overall *operating plan*. The operating plan is vital to each rail carrier because it conducts the operating efficiency, balances customer satisfaction and rail profits. According to the planning horizon, rail operation planning is grouped into three levels: strategic (long term), tactical (medium term) and operational (short term). Refer to Crainic (1999) for hierarchical details.

Guiding the daily operating process, tactical planning is particularly important and interesting for rails. When planning the medium-term operations, scheduling is unavoidable because the operating cost and time might be considerably affected by congestions and delays. In convention, planning and scheduling are separately addressed because of the complexity of each subject. Few works have been developed to analyze the relations between them, except in very simplified settings.

Aiming to combine operation planning together with scheduling at the tactical level, we study the service network design problem. In the problem, we simultaneously consider a mixture of operations. The operations considered are performed in different facilities and may require conflicting resources. The service network design problem analyzes the interactions among the processes and minimizes the total operating cost while meeting the customers' service expectations, avoiding congestions, and obeying various capacities from different resources.

As an extension of the classic network design, a service network design problem generally shows complex formulation. For real-size instances, a service network design aggregating many operations may puzzle any existing solution methods. A simpler case is studied where only direct services are featured. Direct services contribute to a rather large portion of the service set if it is not the only setting as in some small rails. A good direct service design forms the backbone, and can be later used to generate the final service plan.

The major contribution of this paper is twofold. The first aspect is a comprehensive modeling approach that brings the key tactical decisions together in the scheduling context. To analyze the various operations on different aspects as well as the temporal effects, a 2-layer time-space network is constructed, in which we are able to explicitly address the shipment in different formats in a time-dependent manner. Second, we propose an algorithm to solve this formulation at dimensions that may be of interest in practice. The heuristic method is developed on a cycle-based neighborhood, and is proven to be efficient to provide good solutions in reasonable time for large-scale instances. We launch equal endeavors on both modeling and solution method in order to keep the model meaningful in practice as well as successfully solved in the end.

This paper is organized as follows. After the introduction of freight rail operations, we review the previous research works, and detail our problem. A MIP model is then constructed based on a special structure. This model is very complicated, and no optimization technique is known to be efficient on real-sized instances. A tabu search algorithm is developed to find acceptable solutions. The results of random experiments are then presented and analyzed. In the end, we summarize our work and propose extensions.

2 Rail Transportation

The rail network consists of stations and yards, which are connected by rail tracks. Working on the rail network, railways receive transportation demands from customers for shipping cars of commodities from their origin station to their destination station.

Trains are provided on rail tracks. A train is composed of one or more engines providing power, as well as a series of cars. A train assembles at its origin and disassembles at its destination. During the journey on a sequence of rail tracks, the train picks up/unloads some cars at some intermediate stops. Because of the expensive crew charge and locomotive depreciation cost, in general, dedicated trains are not provided for each customer unless the demand is regular and with high volume. To share some common trains on their journey, shipments from different customers have to be consolidated. The consolidation is implemented in *yards* interspersing in rail network. Feeder trains are used to provide transportation between the customer's station and the yard nearby, and main-line trains carry cars through the yards.

A *service* is carried out by a train. Conventionally, a service is characterized by a train route and addressed by frequency. When *schedule* is concerned, which is the timetable depicting the departing/arrival time at each stop on the train route, each service refers to a train with a predefined timetable.

To benefit from the economy-of-scale, cars are not handled individually, and *blocks* are built. A block consists of a group of cars with different origins and destinations, and the cars in a block will be processed as a unit and share a common trip from the block origin to the block destination. Blocks are built at particular yards, called *classification yards*, where many parallel tracks called *classification tracks* exist. To build blocks, cars are sorted by being slipped into (in the case of hump yards) or hauled into (in the case of flat yards) classification tracks at block origin. One classification track must be exclusively assigned to the block for a certain time. The cars are accumulated in the classification track until enough cars are gathered and the block is formed up. The classification process requires a considerable amount of resources, and on average, each re-classification results approximately in a one-day delay for the shipments, making it a major source of expenses and delays. At its destination yard, the block is broken down and its component cars are delivered to the consignees by feeder service if it's their final yard, or are re-classified and go through another block. Clearly, each car may go through one or several blocks before reaching its destination yard.

Services transport blocks on tracks. After a block is built up in its origin, a service, either departing from or passing the classification yard, then takes the block to another yard. If it's not the destination of the block, the block is unloaded onto a *transfer track* and will later be *transferred* (or *switched*) to another service. The transfer process usually causes delays due to the connection between services. Each block is therefore shipped by one service or more. We notice, it's possible that each block only uses several sub-services (service portion between two unnecessarily consecutive stops on a route).

The loaded cars are cleared at their destination. When returned to rails, empty cars need to be repositioned, which means they will be moved from areas with a surplus to areas with a deficit in order to fulfill the future demands.

Managing the complex daily operations, the operating plan at the tactical level has quite complicated setup, and comprises many policies affecting different operations. The most common policies are introduced below. Given a set of potential services, *service selection* determines which services to provide in order to form the service plan. The *blocking policy* is probably the most essential internal regulation, which concerns the decisions on building blocks. Through the *train make-up policy*, one figures out which service takes which block. *Traffic distribution* gives the assignment of cars on blocks, and specifies the itinerary for each demand. The movements of empty cars are managed by the *empty reposition policy*. All these policies have network-wide impacts and are strongly and complexly linked both in economic terms and in their time-space dimension. To study the trade-offs, one need to consider these intertwined issues in one model, and determines those policies concurrently to identify the most efficient way of delivering all shipments while satisfying a set of technological constraints from block, train, track and yard capacity.

3 Literature Review and Problem Description

Service selection, blocking policy, train make-up policy, traffic distribution and empty reposition policy form the fundamentals of a rail operating plan. A large number of articles for planning freight rail operations exist, focusing on different aspects. In this section, we limit ourselves to the works on service selection and blocking policy, as well as some attempts to combine those two aspects. Complete reviews on former rail models can be found in Cordeau et al.

(1998) and Ahuja et al. (2005).

Service selection is typically developed and solved in two steps. Service routing models (e.g. Marín and Salmerón, 1996; Goossens et al., 2004) first determine the routing and frequency of services, and the train timetable is then addressed by scheduling models (e.g. Brännlund et al., 1998; Caprara et al., 2002, 2006) based on the routing pattern. However, the congestion during the train transition suggests modifications to the routing plan. Morlok and Peterson (1970) first determined the train routing and train scheduling in a single optimization model. Huntley et al. (1995) developed a computerized routing and scheduling system to help planners at CSX Transportation where each stop-to-stop link in the routes is assumed to follow an established path. Newman and Yano (2000) proposed a model within the context of scheduling trains and assigning containers for each trains in the rail portion of intermodal shipment.

Bodin et al. (1980) worked on the blocking policy and developed a nonlinear mixed-integer programming model. The blocking model is extended by Newton et al. (1998), and the researchers considered demands with different priorities. A formulation similar to the aforementioned one is used by Barnhart et al. (2000), and they applied a dual-based Lagrangian relaxation approach to solve the problem. Another extension of the blocking model is studied by Ahuja et al. (2007). A special neighborhood search algorithm is developed to heuristically improve the current solution. The algorithm provides blocking policies efficiently and has a real potential to be applied to rails.

Only few efforts have been made to address both service selection and blocking policy simultaneously. Compared with the previous models, these compound models are more complex and much harder to solve. Crainic et al. (1984) defined a feasible journey for each shipment as an itinerary. An itinerary includes the service path followed and the operations performed at each intermediate stop. By selecting the best itinerary for each demand, the model not only solves the traffic routing problem but also determines the blocking and make-up strategies, as well as the distribution of classification workload among yards. A good operating strategy balancing the service cost and quality is obtained. However, blocking decisions are indirect and no service schedule is provided. Crainic and Rousseau (1986) presented a general model of multi-commodity multi-mode freight transportation and explained the solution methodology in greater detail.

Haghani (1989) attempted to combine train routing and scheduling, make-up, as well as empty car distribution problems based on a time-space network with fixed travel times. Keaton (1989, 1992) examined the problem of deciding which pairs of yards are provided with direct service and the frequency, as well as car routing and make-up policy. The proposed model has a linear formulation since neither yard delay nor train transit time is determined endogenously. The objective is to minimize the total cost which includes train costs, car time costs, and classification costs in yards. A similar formulation is solved by Gorman (1998a) with genetic and tabu search and the model is applied in Santa Fe Railway (Gorman, 1998b). All these researches tried to model the blocking process by including classification costs while planning scheduled services. However, no explicit blocking decisions are addressed.

Planning processes have been applied to generate the holistic operating plan for railways, such as Ireland et al. (2004) for Canadian Pacific Railway. These solutions track the entire operating plan problem by aggregating many separate algorithms within a systematic frame. However, the aggregating method failed to analyze the detailed relations among different policies, and further improvements on the operating plan may be available if we are able to consider the various operations together.

Our review of many previous works reveals an opportunity for developing a new framework to link decisions from different aspects of the rail operation. Since no prior work takes the service schedule into account while making the blocking policy, it might be difficult to find a feasible train timetable to accommodate all proposed blocks. On the contrary, if we fix the scheduled service plan first, the outcome blocking policy may violate the block building capacities in yards. Therefore, explicit blocking decisions in scheduled service network design, which are absent in reports, will coordinate the decisions on blocks and services, and synchronize the yard operations.

Our objective is to produce a good operating plan consisting of the essential policies to help rails work in a smooth, rational and cost-efficient way. First, we are interested in the scheduled service design and blocking policy these two vital decision makings. Moreover, to allocate the blocks built to the services offered, we need a train make-up policy. By achieving all these, cars are distributed and a time-dependent itinerary is determined for every traffic demand. The empty reposition policy is assumed to be given so that empty demands come with the demand pattern, and car flows in our model are the mixed flow of loaded and empty cars.

In order to simultaneously address the scheduled service design, blocking policy, make-up policy and traffic distribution, we study a time-dependent service network design problem where the most substantial operations at the tactical level and additional operations at the operational level are integrated. By ensuring the proper delivery for customers, we minimize the total operating cost and provide decisions for both services and blocks.

4 Service Network Design Model

The working procedure in rails can be briefly reviewed as follows. Cars (loaded or empty) received from customers are first put into receiving tracks. After a possible waiting time, the cars are classified and moved into classification tracks where the cars are held until enough cars are gathered and blocks are formed. After that, the blocks are loaded onto services and transported on rail tracks. At a midway stop, some blocks may be unloaded onto transfer tracks in yards and later transferred to another service after connection delays. Thus, shipments are generally processed in the form of cars and blocks, respectively. For cars, shipments are either waiting in receiving tracks, or classified, or accumulated in classification tracks. For blocks, shipments are transported by services, transferred, or delayed in transfer tracks for connection.

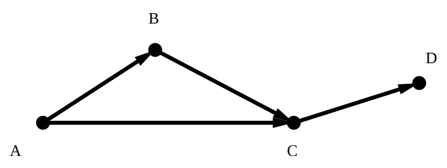

Figure 1: A Simple Physical Network.

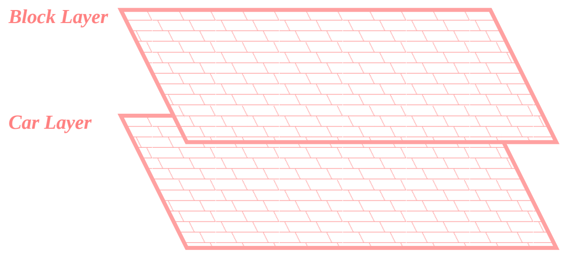

Figure 2: 2-Layer Structure.

4.1 2-Layer Time-Space Network

Trains run on a physical network $G = (V, E)$, where each vertex $v \in V$ represents a yard and each link $e \in E$ stands for a rail track section. Planning at the tactical level, minor stations are aggregated and only major yards handling a large number of traffic, and main-line tracks connecting yards are considered. One simple physical network with 4 yards and 4 directed tracks is illustrated in Figure 1.

Apparently, the physical network, which often appears in earlier works, is insufficient to picture the temporal information and the flows in different formats. To describe the various operations on different objects, we delaminate the physical network into layers. Based on a physical network, a 2-layer structure is constructed (shown in Figure 2). The structure has two parallel layers: the top layer concerns the flow of blocks and is denoted as the *block layer*, and the bottom layer, called the *car layer*, describes the flow of cars.

Moreover, considering the service schedule as well as the delays, temporal information must be addressed. In each layer, a time dimension is attached. A unified granularity is adopted for both layers, where the time horizon is divided into $\mathbf{T}$ time periods by $\mathbf{T}$ time points, denoted by $t \in \{0, \cdots, \mathbf{T}-1\}$. Because of the continuity of the operating plan, we adopt a cyclic time dimension, that is, the next time point of $\mathbf{T}-1$ is back to 0.

To further specify the yard operations, at each time point, each yard is divided into two nodes: the IN node and the OUT node. The IN node indicates that the objects (cars or blocks as in car layer or block layer) are received in the yard at the time point, and the OUT node says the objects are ready to ship out. With all these considerations, a special network structure is constructed, and the total number of nodes in each layer is equal to,

$$2 \times \text{number of yards} \times \text{number of time periods.}$$

Links in the time-space network include the links in each layer, and the vertical links connecting the two layers. The definition of the *temporal length* (or *length*) of a link is the total time periods this link covers.

In the block layer, links are defined to represent the operations applied on blocks, including movement on service, transfer and delay for connection. Corresponding to the physical network in Figure 1, the block layer is shown in Figure 3.

Each *service* is represented by a *direct service link*, denoted as s, which is from an OUT node of one yard to an IN node of another yard. The physical route passed is addressed by a set of tracks $E(s)$. For each service, we are therefore aware of its origin, destination, departure time, fixed running time and service route. A fixed cost $c^f(s)$, which stands for the cost of supplying locomotives and crew, is appended if the service is provided. The flow cost of a direct service link is the fuel cost which is linear with the train load. Direct service links are enumerated by departing each combination of one rail route and one possible speed from each time point. On a pre-arranged route, the service with shorter transit time represents the train running on greater speed, and using lower speed trains causes a longer service time. We notice parallel direct service links, which have the same origin and destination nodes as well as the same transit time, may exist. One example is service s_1 and s_2 in Figure 3. They both depart from node i_1 and end at node i_2. Parallel direct service links distinguish each other by their track sequence, that is, they represent train movements between two yards following different physical routes. In this instance, s_1 and s_2 follow route $(A \rightarrow B \rightarrow C)$ and route $(A \rightarrow C)$ respectively. S is the set of all direct service links.

A *transfer link* connects an IN node to the OUT node of the same yard at the next time point, e.g. (i_4, i_5), standing for the transfer procedure. The flow cost on a transfer link is the cost to unload and later load blocks onto another service. One *transfer delay link* attaches two consecutive IN nodes of a same yard, e.g. (i_6, i_7), representing a one-period delay for catching the next service. A^i is the set of all inter-service transfer links, and A^d the set of transfer delay links.

With direct service links, transfer links and transfer delay links describing the operations on blocks, the journey of a block can be represented by a path. Despite of the building process, we define such a path as a *block path* (or *block*). A block is formed by a series of direct service links which are connected by transfer delay links and transfer links. For each block b, a sequence of direct service links $S(b) \subset S$ is kept to depict the movements on services. The block flow cost is the sum of the flow costs on its component links. Moreover, when a block b is built, we attach a fixed cost $c^f(b)$ to represent the classification track occupancy during the building process. The estimated building time for

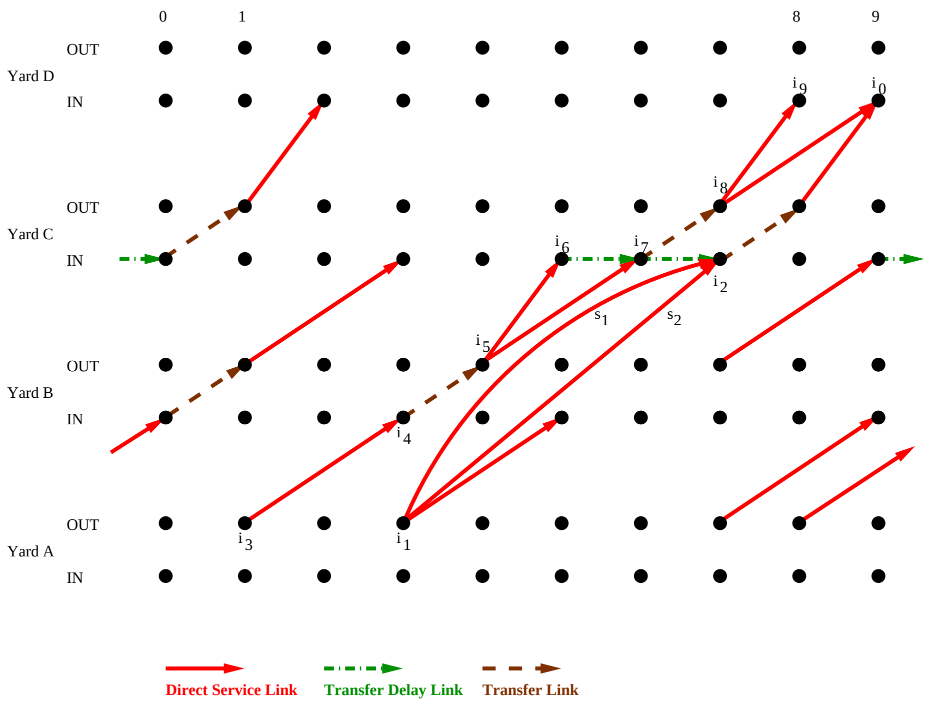

Figure 3: Block Layer.

forming the block in its origin yard is denoted as $h(b)$. Given the links in Figure 3, a number of potential blocks can be generated. For example, $(i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_6)$ marks one block, which is taken by a service departing from yard A at time 1, and transferred at yard B, then shipped by another service (i_5, i_6) and terminates at yard C. $(i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_7)$ describes a similar block, which takes a slower train (i_5, i_7) from yard B to yard C. Let B denote the set of all potential blocks we may generate.

The bottom car layer, as shown in Figure 4, has the same node set as the top layer. Links in the car layer stand for the in-yard operations on cars, including *car waiting links*, *classification links*, and *car holding links*. Each car waiting link joins two continuous IN nodes of a yard, representing cars waiting in receiving tracks to be classified. The classification link connects an IN node to the next OUT node of the same yard, describing the classification process that cars are moved from receiving tracks into classification tracks. Next, a car holding link is between two neighboring OUT nodes of a same yard, and pictures one period of accumulation delay. We associate a flow cost to each link in the car layer. For classification links, the flow cost is equal to the cost of transiting a car from the receiving track to the classification track. The car waiting and car holding costs represent a one-period occupancy expense in receiving tracks and classification

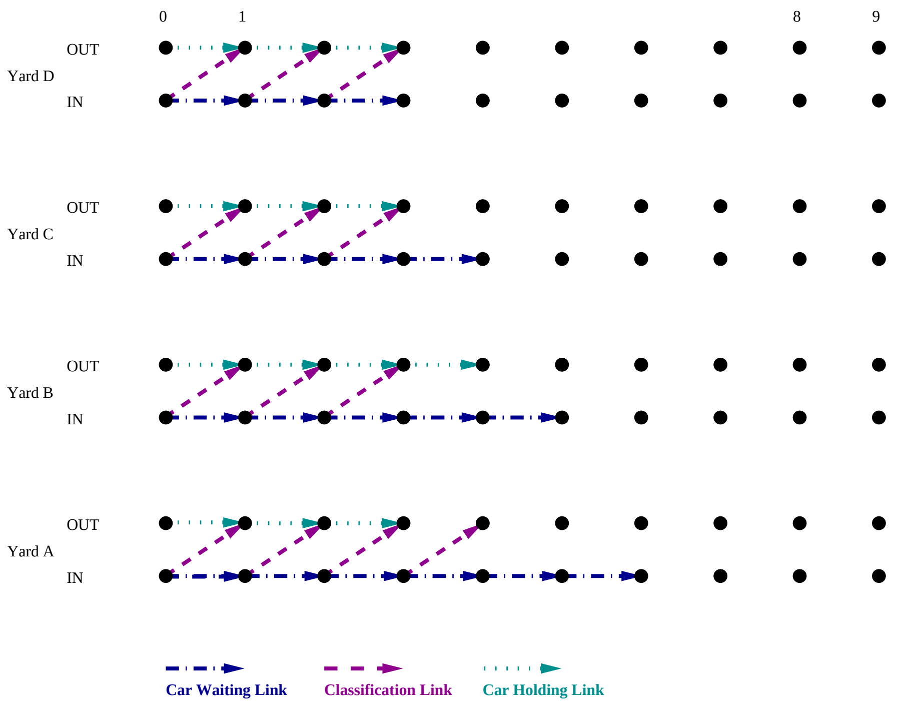

Figure 4: Car Layer.

tracks, respectively. A^w is the set of car waiting links, A^c the set of classification links, and A^h the set of car holding links.

Vertical links are introduced to unite the two isolated layers. An upward link is made from each OUT node in the car layer to the upstairs OUT node in the block layer which represents the same yard at the same time point. This link is called *load link* and depicts the operation that blocks are formed and loaded onto a service. On the contrary, a downward *unload link* is built from each IN node in the block layer to the IN node underneath, and shows the procedure where blocks are unloaded and broken down into cars.

Links in the 2-layer time-space network represent the operations on shipments during the transportation process from origin to destination. Railways exchange cars with customers at the IN nodes in the car layer, as they either receive from or deliver to customers. From the origin car-IN node which represents the receiving at the origin yard, the cars first wait in car waiting links. After being classified through a classification link, cars reach an OUT node, and are delayed for accumulation on car holding links. When the block is formed and loaded onto a service, the traffic goes up to the block layer through a load link. Then, the block is either shipped by direct services, or transferred to another service, or delayed for transfer. At the destination, the block is unloaded

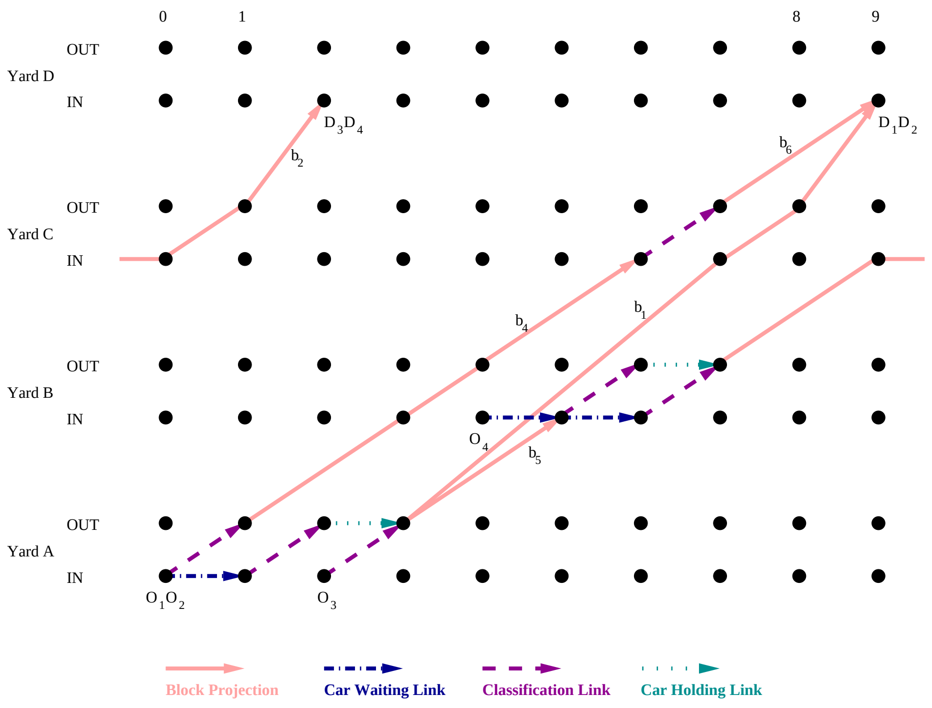

Figure 5: Flows in 2-Layer Time-Space Network.

and broken down, and the traffic returns back to the car layer. The cars are delivered at a car-IN node and the journey ends if it is the final yard of the shipment. Otherwise, the cars will go through the block layer again until reaching its final yard.

With the structure above, it is straightforward to describe a traffic itinerary as a path between two IN nodes in the car layer. An instance is given in Figure 5, which presents the vertical projection of the 2-layer network, where only blocks in the block layer and links in the car layer are shown. Four demands, which are specified by their Origin-Destination (O-D) pairs, must be satisfied. Demands d_1 and d_2 have the same origin and destination, and two itineraries are presented for those demands. The first one passes two blocks, b_4 and b_6, which means the cars will be classified twice in yards A and C. On the second one, traffic is first delayed, but only block b_1 is used. Our job is to choose the best itinerary for each demand, and by doing so, we not only solve the traffic distribution problem, but also provide a block design as well as a service design.

4.2 Mathematical Formulation

To specify the characteristics of different traffic demands, we first define *traffic class*. Each traffic class represents a shipment, which is distinguished by the origin yard o and destination yard d, type of commodity m, the receiving time r when cars are available at the origin yard, as well as the due date. However, working with cyclic time dimension, it is complicated to count the exact date. The due transit hour h, which is the due date minus the receiving time, is used to express the maximal transit time allowed till the final yard. Precisely, a traffic class is defined as,

$$p = (o, d, m, r, h).$$

Let P be the set of all traffic classes.

Our model includes three types of variables. Let A be the union of all car layer links and all blocks in the block layer. The first type, denoted as x_a^p, is the continuous variable representing car flow of traffic class p on each link $a \in A$. A binary decision variable is adopted to model the building decision on each block, which is defined as $y_b = 1$ if block b is built, $y_b = 0$ otherwise. The last type is another binary design variable addressing the service selection, $z_s = 1$ if we include the direct service s in the final design, $z_s = 0$ otherwise.

The objective is to minimize the total operating cost in both layers, which includes the fixed cost on open services, the fixed cost on open blocks, as well as the flow cost on all the links/blocks. The objective function is written as (1),

$$\min \Phi = \sum_{p \in P} \sum_{a \in A} c(p,a) \cdot x_a^p + \sum_{b \in B} c^f(b) \cdot y_b + \sum_{s \in S} c^f(s) \cdot z_s \tag{1}$$

where $c(p, a)$ is the unit flow cost on car layer links/blocks a for traffic class p.

The first constraint comes from the car flow conservation, which guarantees the proper delivery for each traffic demand. Let d^p denote the demand for traffic class p. Since empty flows are present as well, car number is used to address demand. In equation (2), $A^+(n)$ and $A^-(n)$ are the set of outward and inward links (including classification links, car waiting links, car holding links and blocks) of car-layer node n, and w_n^p is the absolute demand for traffic class p on node n. If n is the origin of traffic class p, $w_n^p = d^p$; if n is the destination of p, $w_n^p = -d^p$; otherwise $w_n^p = 0$.

$$\sum_{a \in A^+(n)} x_a^p - \sum_{a \in A^-(n)} x_a^p = w_n^p \qquad \forall n \in N, \forall p \in P. \tag{2}$$

Next, we have two linking constraints (3) and (4) which explicitly specify the relations between the flow of cars and the flow of blocks, as well as the flow of blocks and the flow of services.

$$\sum_{p \in P} x_b^p \leq y_b u_b \qquad \forall b \in B; \tag{3}$$

$$\sum_{b \in B | s \in S(b)} y_b \leq z_s u_s \qquad \forall s \in S; \tag{4}$$

where u_s is the maximum cars that a train can haul on direct service link $s \in S$, and u_b is the car flow capacity on block b.

In the operating process, we also meet many restrictions from diverse resources of the rail system, such as the physical characteristics of the tracks, the yard construction and configuration, equipment allocation and crew assignment policy, etc. All those restrictions could be very close to each instance, and generally, at the tactical/operational level, we have the following forcing constraints.

The *yard handling constraint* marks the maximum classification workload for a yard in each time period. With the constraint, yard congestion is avoided and excess classification work will be delayed and then performed in the following time periods. Let u_a be the handling capacity on classification link $a \in A^c$ in terms of cars. Generally, this capacity comes from the yard structure and configuration, e.g. number of switch engines and flow capacity on each engine. The car handling constraint can be expressed as (5),

$$\sum_{p \in P} x_a^p \leq u_a \qquad \forall a \in A^c. \tag{5}$$

More locomotives on a train can increase its hauling capacity. However, due to the physical condition of the tracks chosen and yards passed, there is still an upper bound on the size of train. We have the *train load constraint* as (6),

$$\sum_{p \in P} \sum_{b \in B | s \in S(b)} x_b^p \leq z_s u_s \qquad \forall s \in S. \tag{6}$$

Based on the assumption that each classification track can be assigned to only one block at a time, the number of blocks being built in one yard should be less than the number of classification tracks in the yard. The constraint holds, although it is possible that several classification tracks are assigned to build one block. With $B(a)$ being a set of blocks which starts at the yard that car holding link a represents, and link a is in $h(b)$ time periods ahead of the block departing node $o(b)$. Therefore, we have inequality (7) as the *block building constraint*,

$$\sum_{b \in B(a)} y_b \leq u_{v(a)} \qquad \forall a \in A^h \tag{7}$$

where $u_{v(a)}$ stands for the maximum blocks being built in the yard at the time, represented by classification link a.

The *train running constraint* comes from the fact that in each time period each rail track can accommodate a restricted number of trains. In general, train running capacity depends on the physical track condition, track mile, the speed and direction of each train, as well as time interval between adjacent departures, etc. Working at the tactical level, however, there is no need to specify the detailed on-track delay and we assume the train running capacity only depends on track condition. Let $S(e, t)$ be the set of services running on physical track $e \in E$ in time period t. For each track e in rail network G, u_e is the capacity of trains simultaneously running on e. Consequently, we have the train running constraint (8),

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

With the proper train running constraints in formulation, we prevent the on-track congestion and ensure the on-time transportation of trains.

The complete formulation is summarized below.

$$\begin{aligned}
\min \Phi = \sum_{p\in P}\sum_{a\in A} c(p,a)\cdot x_a^p + \sum_{b\in B} c^f(b)\cdot y_b + \sum_{s\in S} c^f(s)\cdot z_s & \\
\text{s.t.} \sum_{a\in A^+(n)} x_a^p - \sum_{a\in A^-(n)} x_a^p = w_n^p & \qquad \forall n\in N, \forall p\in P;\\
\sum_{p\in P} x_b^p \le y_b u_b & \qquad \forall b\in B;\\
\sum_{b\in B|s\in S(b)} y_b \le z_s u_s & \qquad \forall s\in S;\\
\sum_{p\in P} x_a^p \le u_a & \qquad \forall a\in A^c;\\
\sum_{p\in P}\sum_{b\in B|s\in S(b)} x_b^p \le z_s u_s & \qquad \forall s\in S;\\
\sum_{b\in B(a)} y_b \le u_{v(a)} & \qquad \forall a\in A^h;\\
\sum_{s\in S(e,t)} z_s \le u_e & \qquad \forall e\in E, \forall t\in\{1,\cdots,\mathbf{T}\};\\
x_a^p \ge 0 & \qquad \forall a\in A, \forall p\in P;\\
y_b \in\{0,1\} & \qquad \forall b\in B;\\
z_s \in\{0,1\} & \qquad \forall s\in S.
\end{aligned}$$

The service network design model generates a linear mixed-integer formulation. By choosing z_s for all $s\in S$, we characterize the service routing, schedule, and speed (by the length of direct service link). The variable y_b decides which block should be built, and the make-up policy is addressed by the according $S(b)$. With the variable x_a^p for all the $a\in A$ and $p\in P$, we determine the routing of cars and the processes cars should go through at each yard. Moreover, our model gives an approximated schedule for yard operations (classification and transfer) by x_a^p.

This model takes into account the most essential elements of tactical/operational planning in rails. It is probably the first effort to integrate service selection, blocking policy, train make-up policy, and traffic distribution altogether in a time-space structure. By analyzing the relations and effects among these tangled issues, we believe promising results exist if the problem can be decently solved.

5 Tabu Search Algorithm

The service network design model introduced above takes a special form of the Fixed-cost Multi-commodity Capacitated Network Design (FMCND) formulation, which belongs to the NP-hard complexity class. Moreover, our model is much more complicated than the general FMCND because of the additional variables introduced and constraints considered. For instances of interesting sizes, we therefore believe only specially tailored heuristics can discover good solutions in a rational computing time.

Several heuristic methods have been developed for the general FMCND problem. The simplest one is the add/drop procedure (e.g. Powell, 1986) based on reduced-cost calculations, that is, in each iteration, one or several arcs are included or excluded in the current design. The add/drop idea is pretty simple but is proven as inefficient on large instances. Subgradient (Farvolden and Powell, 1994) and slope scaling (Kim and Pardalos, 1999) methods have also been developed. These heuristics fail to explore beyond the local optimum. Even guided by long-term memory (Crainic et al., 2004; Kim et al., 2006), the search could still terminate far from optimum. A simplex-based tabu search on path-based formulation was presented by Crainic et al. (2000). The algorithm derives from applying column generation in the simplex method, and is capable to identify good solutions for pretty large instances. However, the process requires a fitting linearization of the objective function which may be inapplicable here because of double sets of integer variables in the model. The cycle-based neighborhood by Ghamlouche et al. (2003) is another structure that allows meta-heuristics to find good solutions on the network design problem. The cycle-based neighborhood defines moves that may explicitly consider the impact on the total design cost of potential modifications to the flow distribution of several commodities simultaneously. The fundamental idea is to explore the space of the design variables by redirecting flow around cycles, closing and opening design arcs accordingly. Thus, compared with link-based neighborhoods, move evaluations are more comprehensive since all commodities on a cycle are considered, plus the range of moves is broader because flow deviations are no longer restricted to paths linking origins and destinations of actual commodities. A tabu search is developed and computational experiments on problems of various sizes (up to 700 arcs and 400 commodities) show that the heuristic works quite well on the cycle-based neighborhood. The tabu search algorithm is later strengthened by the path relinking method (Ghamlouche et al., 2004).

The cycle-based neighborhood idea is applied here due to the promising performance on FMCND. As in a minimization problem, we notice in our model, services which are not used by any open block should not appear in the final design, and whenever a block is open, all its component services must also be open. Each block pattern thus has its optimal direct service pattern. The advantage gives us the privilege to focus on the block design, and only implicitly consider the service decisions during the solution procedure. Based on the observation, we define a neighborhood focusing on changing the status of blocks in the means of deviating traffic flows within a cycle in the time-space network.

First we define a *cycle* as a closed chain consisting of car-layer links and blocks, and the links/blocks in the chain may follow different directions. A *block design* is a vector $\bar{y}$ as well as the inherent optimal service design vector $\bar{z}$. If $\bar{y}$ honors the block building constraints everywhere and the associated $\bar{z}$ satisfies the train running constraints on all tracks in each time period, we say the block design $\bar{y}(\bar{z})$ is *eligible.* For any solution with an eligible block design, the neighbors of current block design are defined as all the eligible block designs we can obtain through the steps below.

1. Choose a block. Starting from the block, identify a cycle.
2. Deviate the total car flow through the cycle, from one path to the other, so that at least one block status changes.
3. On the cycle, open all the blocks with flow, and close all other blocks.

The neighborhood specified by the above procedure can be very large if we take all the blocks in consideration, and an enumeration of all potential cycles is impractical. To achieve good feasible solutions in short computing time, we restrict the neighborhood scope in each move.

Given a block design $\bar{y}(\bar{z})$, the distribution of traffic is given by a capacitated multi-commodity minimum-cost flow problem in the car layer. The associated car flow problem is defined as Traffic Routing Sub-Problem (TRSP).

$$\begin{aligned}
\min \quad & \phi'(\bar{y}(\bar{z})) = \sum_{p\in P}\sum_{a\in A} c(p,a)\cdot x_a^p && && (9)\\
\text{s.t.} \quad & \sum_{a\in A^+(n)} x_a^p - \sum_{a\in A^-(n)} x_a^p = w_n^p && \forall n\in N, \forall p\in P;\\
& \sum_{p\in P} x_a^p \le u_a && \forall a\in A^c;\\
& \sum_{p\in P}\sum_{b\in B|s\in S_b} x_b^p \le \bar{z}_s u_s && \forall s\in S;\\
& \sum_{p\in P} x_b^p \le \bar{y}_b u_b && \forall b\in B;\\
& x_a^p \ge 0 && \forall a\in A, \forall p\in P.
\end{aligned}$$

The objective of TRSP, denoted $\phi\big(\bar{y}(\bar{z})\big)$, includes the travel costs on all car-layer links and blocks, but does not have the fixed design costs of the block design. The value $\Phi\big(\bar{y}(\bar{z})\big)$ on block design $\bar{y}(\bar{z})$ is then obtained as $\phi\big(\bar{y}(\bar{z})\big)$ plus the design costs, which is $\sum_{b\in B} c^f(b)\cdot \bar{y}_b + \sum_{s\in S} c^f(s)\cdot \bar{z}_s$.

Our tabu search algorithm starts with an initialization phase, which provides an eligible block design. Then, the algorithm iteratively improves the current block design by implementing a sequence of *cycle-based local searches* until the *stop criteria* are met. In each local search, the existing solution is improved by replacing the current block design with one of its neighbors, and then the traffic distribution is determined by the associated TRSP. When we reach a promising area, an *intensification* phase is called for further improvement. In case the improvement of a local search is insignificant or no good neighbor shows up, *diversification* is launched to escape from the local optimum. Schematically, an outline of the tabu search algorithm is written as,

1. *Initialization.*
2. Repeat until *stop criteria* are met,
    - *cycle-based local search*;
    - if the current solution is better than or close to the best overall, *intensification*;
    - else if the improvement is negligible, *diversification.*
3. Stop.

In the following, we describe the main algorithmic choices and detail each procedure, except the stop criteria which will be set within computational experiments.

5.1 Initialization

The initialization phase provides us an initial eligible block design. In order to satisfy the customers' demands to the maximum extent, it is preferable to open all potential blocks in the very beginning. However, the train running capacity and the block building capacity prevent us from opening all the blocks.

An integer programming formulation is solved to generate an eligible block design with a maximal number of blocks. The problem has two sets of binary decision variables on blocks and services correspondingly. The objective (10) is to maximize the number of open blocks, subject to block building constraints and train running constraints, as well as the linking constraints between blocks and services. The IP problem is then solved by the branch-and-bound method. As there is little need to find the optimal solution for the initial problem, the solving procedure stops when a near-optimal solution is obtained in a short time.

$$
\begin{aligned}
\max \quad & \sum_{b \in B} y_b && && (10)\\
\text{s.t.} \quad & \sum_{b \in B | s \in S(b)} y_b \leq z_s u_s && \forall s \in S;\\
& \sum_{b \in B(a)} y_b \leq u_{v(a)} && \forall a \in A^h;\\
& \sum_{s \in S(e,t)} z_s \leq u_e && \forall e \in E, \forall t \in \{1, \cdots, \mathbf{T}\};\\
& y_b \in \{0,1\} && \forall b \in B;\\
& z_s \in \{0,1\} && \forall s \in S.
\end{aligned}
$$

By opening as many blocks as possible, the initial block design leads to preposterously large design costs on both blocks and services. Nevertheless, the extra high fixed cost cannot guarantee the feasible flow of cars. This is due to the additional constraints considered in the model, coming from the car handling capacity restricting the number of cars that can be classified at each yard, as well as the car flow capacity on each block. The infeasibility of the according TRSP is avoided by introducing the *artificial links* (or *super itineraries*) in the car layer. For each traffic class, an artificial link is appended, from the IN node that represents the receiving of the demand to the IN node that corresponds to the destination. The length of the artificial link equals to the due transit hour of the traffic class. Each artificial link receives the same flow capacity as the demand of the associated traffic, and an arbitrarily high flow cost ensures that the artificial link will bear positive flow only if the distribution is infeasible. With the additional artificial links, the associated TRSP always has a solution irrespective of the current block design. Our algorithm will start with the initial eligible block design, regardless if the artificial links are used or not, and try to reach a feasible distribution before meeting the stop criteria.

5.2 Residual Network and Local Search

Starting from an initial block design, we progress to improve the current solution by running a series of *local searches.* The intention of local search is to divert flows following a cycle, so that the status of one or several blocks is modified, which leads to significant modification of traffic flows. Keeping the idea in mind, we are particularly interested in the cycles bearing enough flow and after deviation flows on some blocks are empty so that those blocks are free to close.

First, we define *residual value* the flow volume one can deviate around the cycle. Consequently, we are interested in such cycles with their residual value equal to the flow on one open block. On a block design $\bar{y}(\bar{z})$, we denote Γ as the set of flow volumes on all open blocks,

$$\Gamma(\bar{y}) = \{\sum_{p \in P} x_b^p > 0, b \in B \text{ and } \bar{y}_b = 1\}.$$

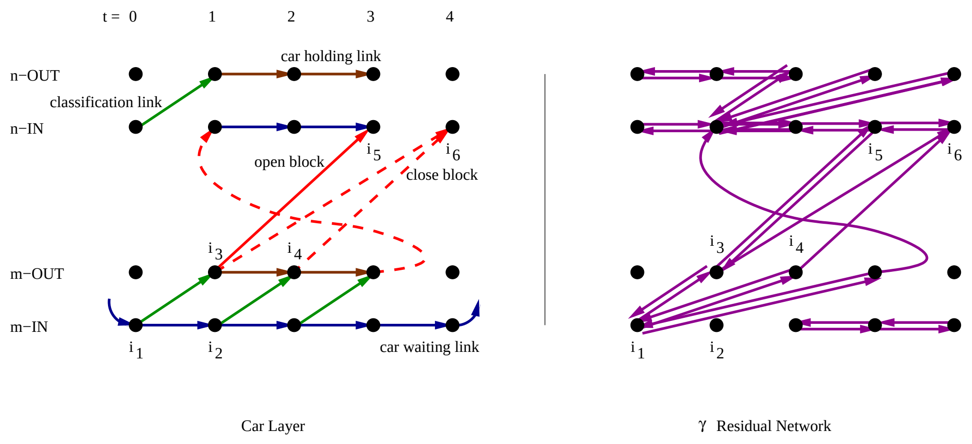

Figure 6: Construction of γ-Residual Network.

For each value $\gamma \in \Gamma$, we identify the cycles that support the variation of γ units of flow, and an accessorial γ-*residual network* is constructed.

The γ-*residual network* has the same nodes as the car layer, and the conceptual arcs in the residual network represent the changing of γ flow, either increasing or decreasing. The γ-residual network uses a positive-cost arc to represent an increase of γ flow on the link/block when there is enough flow capacity remaining. And flow decrease on the link/block is described by an arc with negative cost if extra γ flow exists.

On different links/blocks, we have different regulations for introducing residual arcs. First, for each block b from nodes i to j, we introduce a forward arc $(i,j)^+$ if an additional γ cars of flow can pass on block $b = (i,j)$. That is, the residual capacity on b is greater than or equal to γ. A positive cost c_{ij}^+ is associated with the arc $(i,j)^+$, approximates the cost for routing extra γ cars of average commodity on the block, plus the possible design cost if the block, as well as the direct services underneath, are currently closed. Symmetrically, we include a backward arc $(j,i)^-$ if the total traffic flow on block $b = (i,j)$ is no less than γ. The cost on arc $(j,i)^-$ is the saving value for reduction of γ flow on block b. The possible design cost of block b, including the fixed cost on the block and the potential fixed costs on some of the direct services if applicable, is also subtracted once the reduction leaves the block and services empty. The construction of the γ-residual network is illustrated in Figure 6. For the open block (i_3, i_5), we may have arc $(i_3, i_5)^+$ if there is enough flow capacity left and arc $(i_5, i_3)^-$ if the current flow on the block is over γ. Corresponding to the close block (i_3, i_6), we include arc $(i_3, i_6)^+$ if the residual capacity on the block is over γ.

On each car waiting link, e.g. (i_1, i_2), or car holding link, e.g. (i_3, i_4), both the $(i_1, i_2)^+$ and $(i_2, i_1)^-$, as well as the $(i_3, i_4)^+$ and $(i_4, i_3)^-$ can be defined following the same principle above. Particularly, we always have arc $(i_1, i_2)^+$ and arc $(i_3, i_4)^+$ since there is no flow capacity applicable on both car waiting links and car holding links. Focusing our attention on design decisions, instead of constructing residual arcs for each classification link, we aggregate the classification flows and build residual arcs to represent adding/subtracting flows on in-yard paths made of classification links, car waiting links and car holding links between each IN-OUT node pair. For example, an arc $(i_1, i_4)^+$ is used to represent that extra γ flow is added between i_1 and i_4, precisely on paths $(i_1 \rightarrow i_2 \rightarrow i_4)$ and $(i_1 \rightarrow i_3 \rightarrow i_4)$. If the total flow on these paths exceeds γ, an arc $(i_4, i_1)^-$ is included on which we associate a negative cost to represent the saving for the decline of γ flow on the in-yard paths between i_1 and i_4.

Given an open block b in the current solution, we therefore generate a residual network with a residual value equal to the car flow γ_b on the block. Starting from the residual arc representing clearing the flow on block b, enumeration of all cycles passing through the arc in the γ-residual network is very expensive. We here are particularly interested in the lowest-cost cycle only, which implies the maximal savings following the deviation. The idea of forming such lowest-cost cycles is to find the lowest-cost path from j to i in the residual network to complete the cycle for an arc (i, j). To find the path with the lowest cost from j to i, an *extended Bellman-Ford algorithm* is developed. The algorithm keeps a vector at each node, labeling the temporal length of the lowest-cost path. The length vectors eliminate over-length paths since by our assumption each demand must be finished in its due transit hour. Furthermore, examinations are performed to avoid looping in lowest-cost paths because of negative arcs present in the residual network. From j to i, the algorithm is proven efficient to find a series of lowest-cost paths with different temporal lengths. We pick the one with the overall lowest cost, together with the origin arc, a *lowest-cost cycle* is then obtained.

The pseudo code of the extended Bellman-Ford algorithm is described as follows.

```
1. Define temporal length upper bound $\tau$.
2. For each node $n \in N$ and each $t \in \{1, \cdots, \tau\}$ do,
    • $distance(n, t) = \infty$ and $prev\_node(n, t) = \varnothing$.
3. For each $t \in \{1, \cdots, \tau\}$ do,
    • $distance(j, t) = 0$.
4. $LIST = \{j\}$.
5. While $LIST \neq \varnothing$ do,
    • remove an element $u$ from $LIST$;
    • for each node $v \in N^+(u)$ and each $t \in \{1, \cdots, \tau\}$ do,
```

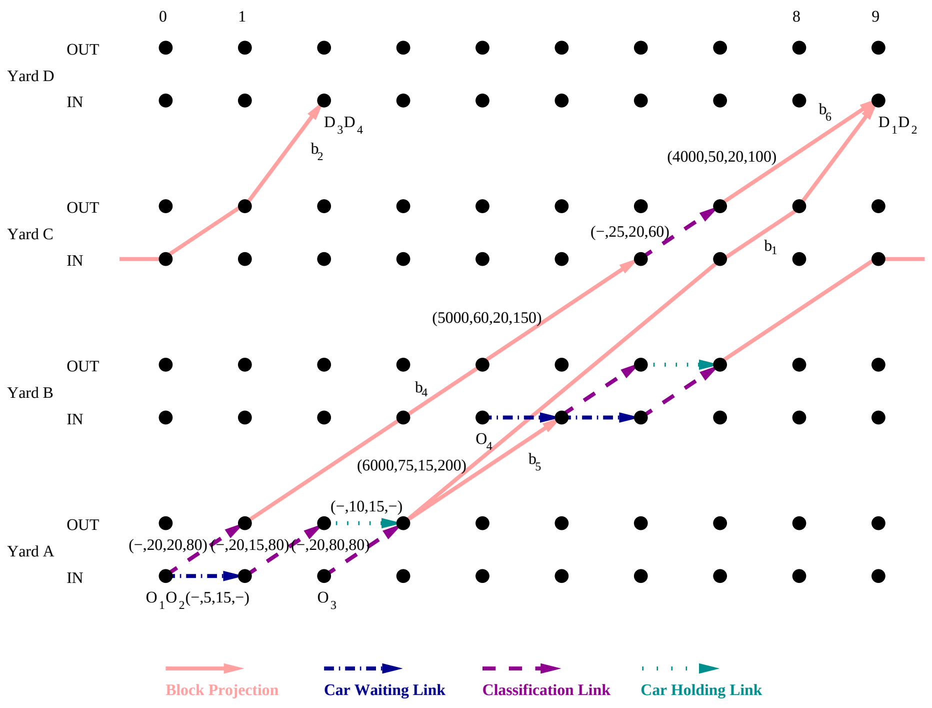

Figure 7: Solution in 2-Layer Time-Space Network (Projection).

- $t' = t + len(u, v)$;
- if $t' \leq \tau$ and $distance(v, t') > distance(u, t) + cost_{u,v}$ and v is not in the path from j to u do,
  - $distance(v, t') = distance(u, t) + cost_{u,v}$;
  - $prev\_node(v, t') = u$;
  - if $v \notin LIST$, then add v to $LIST$.

6. Stop.

In a local search, lowest-cost cycles deriving from all open blocks in the current solution are compared and the one with the minimum cost is selected. Furthermore, the minimum-cost cycle is accepted only if it is non-positive. A non-positive cost cycle accelerates the convergence, and has the highest probability to yield a better solution. We then move the flow following the minimum-cost cycle. After the deviation, the origin block of the cycle is cleared and closed. Some other blocks may be opened or closed accordingly to achieve the new design.

The instance shown in Figure 5 is further detailed in Figure 7, where the information on each link is illustrated as (fixed cost, unit flow cost, current flow, flow capacity). Notice that demands d_1 and d_2 have the same origin and destination node, but

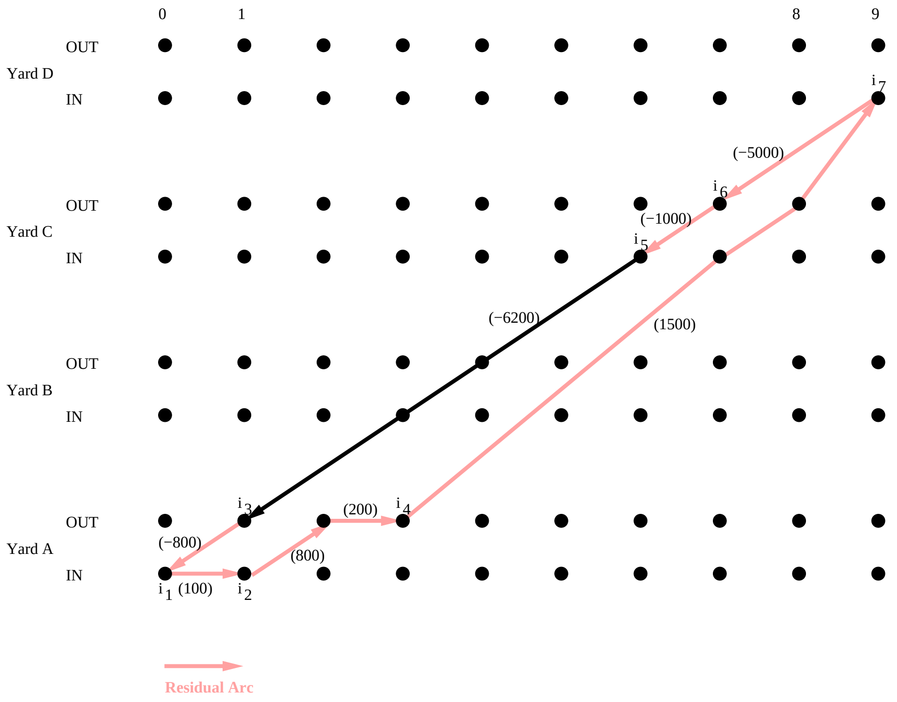

Figure 8: A Lowest-Cost Cycle in Residual Network.

they follow different itineraries in the current solution. d_1 goes through block b_4 and then block b_6, and d_2 uses only block b_1. Given the open block b_4 and its existing flow $\gamma = 20$, we generate a γ-residual network. As shown in Figure 8, emanating from the destination node of arc $(i_5, i_3)^-$, which represents eliminating γ flow on block b_4, we apply the extended Bellman-Ford algorithm. From i_3 to i_5, several lowest-cost paths with different lengths can be obtained, and the path with the overall lowest cost is selected. Together with arc $(i_5, i_3)^-$, a lowest-cost cycle is formed.

Suppose the cycle shown in Figure 8 is also the minimum-cost cycle, among the lowest-cost cycles from all the open blocks in the current design. Following the cycle, flows are moved and a neighbor block design is obtained (shown in Figure 9). On the new block design, demand d_1 now will be delayed and later go through block b_1 together with demand d_2, and blocks b_4 and b_6 are cleared and closed.

Two short-term memories, one for blocks and one for direct services, are used to prevent local search from looping or entering infeasible domains. The memories, called *block tabu list* and *service tabu list*, keep a tabu tenure for each block/direct service. A positive tenure means the block/direct service is prohibited to be opened or closed. When a new design is achieved, random tabu

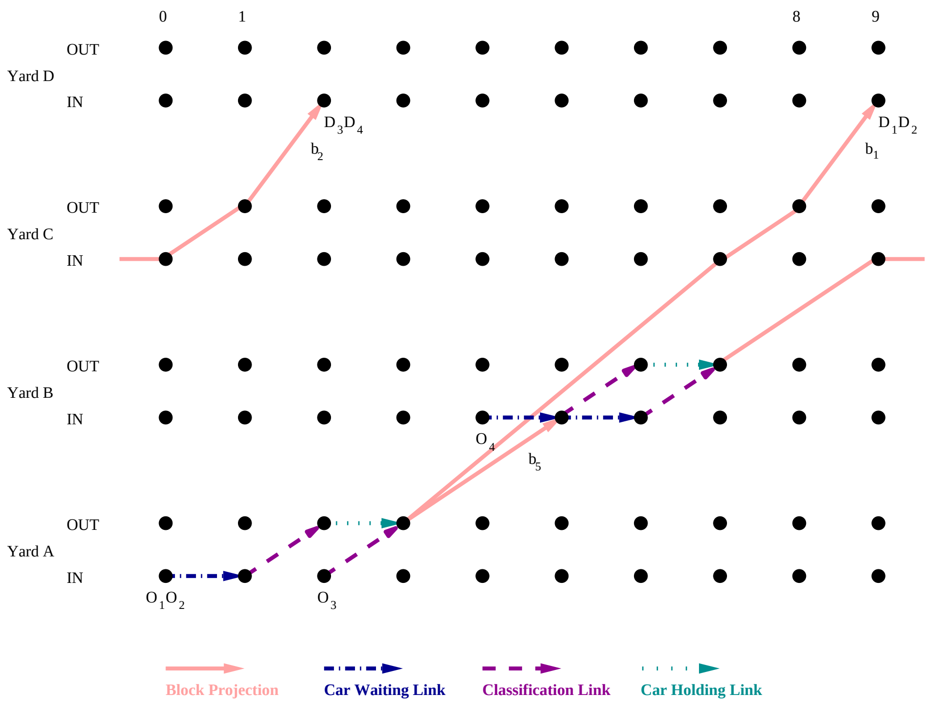

Figure 9: New Solution on Neighbor Block Design.

tenures are attached to all the blocks/direct services have just been modified. All positive tenures in both tabu lists descend by 1 each time we return to a feasible flow distribution on an eligible block design.

After a candidate neighbor block design is obtained through a non-positive cost cycle, a verification of the block building constraint and train running constraint is performed. The verification ensures the procedure remains on the eligible block design domain. If one of these constraints is violated, the last eligible block design is restored, and we tabu the blocks that disobey the block building constraint in the block tabu list, as well as the illegal direct services for the train running constraint in the service tabu list. The extended Bellman-Ford algorithm then restarts to find another minimum-cost cycle consisting of non-tabu blocks and non-tabu direct services, and then the outcome block design is validated again. As long as the new block design satisfies the block building constraint and train running constraint, we move to the new eligible block design.

The traffic is re-distributed by solving the associated TRSP. As a linear network flow problem, TRSP on a given block design can be well solved with the simplex method. It is possible that artificial flows exist in the resulting TRSP solution. The reasons for the artificial flow are twofold. First, in the local search procedure, we failed to explicitly take the train load constraint into account, and the defacto relaxation may lead to an overflow on some services. Second, we move a group of demand flows at each time. As a common part of traffic itineraries is replaced, the modification may not be practicable for all the relative demands. A *restoration* phase is then implemented to regain a feasible flow distribution. The restoration phase is similar to the local search procedure. However, we generate cycles deriving from artificial links instead of open blocks, and the residual value set is restricted to the flow volumes on artificial links.

The local search procedure is synthesized below.

1. Given the current solution, for each open block b do,
   - determine the current flow γ on block b ;
   - construct the γ-residual network;
   - based on block b, apply the extended Bellman-Ford algorithm to determine a set of lowest-cost paths with different lengths by non-tabu blocks and direct services;
   - pick the path with the lowest cost, and form a lowest-cost cycle;
   - update the minimum-cost cycle.
2. If the minimum-cost cycle is non-positive do,
   - following the minimum-cost cycle, obtain a candidate block design;
   - update *block tabu list* and *service tabu list*;
   - if the candidate block design is eligible,
     - move to the candidate block design by opening and closing the appropriate blocks and the associated direct services;
     - solve the associated TRSP;
     - if artificial links are used, *restoration*;
3. Stop.

The local search performs the basic search move in the cycle-based neighborhood. Compared with the cycle-based tabu search by Ghamlouche et al. (2003), in which the algorithm evaluates all the cycles coming from all the links, our local search is restricted to the cycles deriving from blocks currently open, and we only adopt the non-positive cycles. Furthermore, in our algorithm, we focus our attention on the design decisions only, and leave the in-yard flow determined by the network flow problem. All these arrangements impel the procedure to converge faster, and enable us to find superior solutions within logical solving time.

5.3 Intensification

One of the privileges of the cycle-based neighborhood is the wide range of moves in the solution space. However, when working in a petite promising space, it becomes a drawback and the search may "step over" the optimum with a faulty cycle, which shows a considerable cost reduction but leads to an infeasible solution. We propose the *intensification*, a supplementary procedure aims to elaborately polish the current solution. Two separate intensification phases are employed in a sequence.

In the first intensification phase, an *elite* service network design problem is solved. The elite problem has the same formulation as the original model, and only includes the opening blocks as well as their "parallel" blocks – the blocks with the same routing departing at the previous time point as well as the next time point. The elite problem integrates many cycles that represent postpone/advance the existing open blocks for one period, and it helps to converge steeply by avoiding the analysis of a sequence of "small" cycles separately. With only a few variables considered, the elite problem is tractable with the branch-and-bound method.

No matter the first intensification progresses or not, we implement a second phase. The second phase can be viewed as a special local search procedure, and we consider the flow of a demand at each time, assuming the flows of other demands remain unchanged. The same cycle-based neighborhood definition is applied. As in the local search, a set of residual values is first determined, which is the flow of a traffic class on every open block. For a traffic class $p \in P$, Γ_p is then defined as $\Gamma_p(\bar{y}) = \{x_b^p > 0, b \in B \text{ and } \bar{y}_b = 1\}$. A γ-residual network is built based on each γ, and the lowest-cost cycles are formed and evaluated. Moves are adapted only if the minimum-cost cycle has negative value. The process repeats until no negative cycle is detected in any γ-residual network, and then the process continues to the next traffic p. Since the pivots in the intensification phase concern the flow of one traffic class only, there is no need to re-solve the associated TRSP every time. Car flows will be shifted by hand, as long as the train load constraints are satisfied.

The intensification phases are implemented when reaching a hopeful solution area, that is, when a new best or near-best solution is found.

5.4 Diversification

In case the local search fails to improve the current best solution, or the improvement is insignificant, it implies that we are encaged in a local optimum. As we are working in a huge solution space, being trapped in a local area is inevitable. To jailbreak from the local optima, we develop a *diversification* process.

The diversification is actualized by two long-term memories, which are used to keep the opening frequency for blocks and direct services. These memories are denoted as *block frequency list* and *service frequency list*, respectively. Each time a feasible flow distribution on an eligible block design is obtained, we accumulate the opening frequencies of all the open blocks and open direct services by 1 in frequency lists.

To diversify the solution space, a small fraction of open blocks with high opening frequencies are forced to close. Also, tabu tenures of these blocks are updated to avoid returning back in the following several local searches. Same strategy is applied on direct services, and by closing the most-opened direct services, all blocks passing on the direct service are also closed. Furthermore, to prevent closing the same backbone block repeatedly, an additional local memory is adopted to keep the diversification history.

With the closure of several essential blocks, artificial links might be used to carry extra flows. Restoration process is applied to find an alternative block design and recover the feasible traffic distribution. The local search will restart there when a feasible solution is achieved.

6 Experiments

To evaluate the performance of the tabu search algorithm proposed, experiments are implemented.

6.1 Random Instances

The input of the model includes the physical characteristics, demand pattern, as well as the potential direct services and blocks. Deriving from the rail system, the physical network features and demand pattern are straightforward, and can be easily accessed. Furthermore, we suppose that a complete list of possible direct services is also known. The potential blocks are provided by a *block generation* process.

By definition, a block is a path constructed by direct services together with transfer links and transfer delay links. In the block layer, starting from an OUT node, each block has to start with a direct service. To ensure that the block also ends with a direct service, instead of searching a path between the origin OUT node i to a destination IN node j, we first build a path from i to another OUT node k, where k indicates a different yard from i. A simple deep-first-search algorithm is applied to enumerate all paths from i to every other OUT node k, and then each path is attached with a direct service departing from k if there exists, so that one complete block is formed.

The potential blocks we generate should be practicable, which is always restricted by the equipments, environment, operation time, transportation laws, human resources, etc. For example, the total travelling time for a block should be less than a preset amount of time, e.g. $\mathbf{T} - 1$, the over-flexuous blocks and the blocks following a loop in the physical network should always be pruned, etc. These restrictions are denoted as *application constraints.* The application constraints are closely related to individual instance, and different rails could have different application constraints according to their company policy, scale, location, etc. By erasing the redundant and impracticable blocks, these application constraints may dramatically simplify the time-space network, and show

Inst	Block	D-Serv	Yard	Track	Time	Demand
s01	360	160	3	6	10	30
s02	270	140	3	6	10	60
s03	310	140	3	6	10	90
s04	1340	400	4	10	10	40
s05	720	280	4	10	10	80
s06	660	280	4	10	10	120
s07	1790	680	5	18	10	50
s08	1050	500	5	18	10	100
s09	1420	620	5	18	10	150

Table 1: Instance Set S

great effects on the solution procedure.

The instances tested are randomly generated. Even with the same physical magnitude and the same time horizon, random instances greatly differ in the number of blocks and services. Moreover, various constraints on different operations with diverse resources make it impossible to unify the compact/loose concept between all capacities and flows in the 2-layer time-space network. The relative ratio between fixed cost and flow cost is also hard to specify because we have fixed costs from both blocks and services. As a result, we leave the variance of the instances to randomicity, and for each combination of physical magnitude and time horizon, three instances are produced with increasing demand numbers to roughly partition the car flow density.

Three sets of instances are generated to test the tabu search proposed above. Set S includes the small instances with only 3 to 5 yards and a time horizon with 10 time periods. Set M is the medium set, and the instances in set M have 7 yards with either 10 or 14 time periods. In set X, the instances vary from 7 to 10 yards with 14 time periods, which are very large and extremely difficult to solve.

Instances are shown in Table 1, 2, and 3. Three tables show that the appended time dimension and the 2-layer structure multiple the network scale. Even the smallest instance with only 3 yards, we generate hundreds of blocks and end up with more than 10 thousand variables. Moreover, with the increase of the number of yards and tracks, as well as the length of the time horizon, the size of the instances expands explosively. For large instances with 10 yards and 14 time periods, we easily produce hundreds of thousands of blocks, and millions of flow variables.

6.2 Parameters Calibration

The major parameters guiding the heuristic algorithm are calibrated first.

- Tabu tenure is the most important parameter in a tabu search. A proper setting of tabu tenure will prevent the need to modify the block/direct service status again. Two intervals, $7-15$ and $10-20$ are proposed. Each

Inst	Block	D-Serv	Yard	Track	Time	Demand
m01	9310	2140	7	20	10	150
m02	5760	1440	7	20	10	200
m03	6740	1240	7	20	10	250
m04	6240	2300	7	32	10	200
m05	2970	1230	7	32	10	250
m06	5730	2040	7	32	10	300
m07	29092	3080	7	20	14	250
m08	80374	3136	7	20	14	300
m09	19824	2212	7	20	14	350

Table 2: Instance Set M

Inst	Block	D-Serv	Yard	Track	Time	Demand
x01	35014	6356	7	32	14	450
x02	9212	3556	7	32	14	500
x03	17570	3528	7	32	14	400
x04	99946	15274	10	60	14	400
x05	126490	18872	10	60	14	500
x06	116494	13356	10	60	14	600
x07	39172	16464	10	60	14	700
x08	81298	23072	10	60	14	800
x09	124488	22358	10	60	14	900

Table 3: Instance Set X

TabuTenure	IntensThreshold	DiversRatio	Sol. Score	Time Score	Ttl Score
07-15	5%	8%	47	43	90
07-15	5%	10%	41	51	92
07-15	5%	15%	64	30	94
07-15	7%	8%	46	39	85
07-15	7%	10%	35	48	83
07-15	7%	15%	59	27	86
10-20	5%	8%	48	40	88
10-20	5%	10%	35	32	67
10-20	5%	15%	49	47	96
10-20	7%	8%	35	41	76
10-20	7%	10%	42	45	87
10-20	7%	15%	57	35	92

Table 4: Parameter Calibration

time a tabu status is updated, either in block tabu list or service tabu list, a random tenure is selected in the interval and is associated to the according block or direct service.

- To conduct the intensification phase, a threshold is used to determine the good solutions. When the gap between the current solution and the best solution so far is less than the threshold, the intensification phase is triggered. Two values, 5% and 7%, are compared.
- The diversification percentage figures out the variance range in each diversification phase, and manages the searching scope of the algorithm. Either the 8%, 10% or 15% top-frequency blocks/services are closed in diversification phase.

A total of 12 parameter settings are contrasted. Calibration experiments are implemented on 9 random instances, which are each solved with all parameter settings. The calibration experiments end when we reach the 20 continuous non-improved local searches. We rank the parameter combinations for each instance according to both solution quality and solving time. A score of 10 to 1 is assigned to each of the first 10 places on quality and time respectively. Better the solution, higher the solution score; longer the solving time, lower the time score. The scores on each problem are summed up and shown in Table 4, where the first three columns present the setting, the two columns in the middle hold the solution score and time score, and the last column displays the total score.

Statistics in Table 4 show that setting (7-15, 5%, 15%) gives the best convergence, and setting (7-15, 5%, 10%) leads to the shortest solution time. Parameter setting with $TabuTenure = 10 - 20$, $IntensThreshold = 5\%$ and $DiversRatio = 15\%$ achieves the highest total score, and balances the trade-offs between solution quality and efficiency. In the following, we experiment with this setting.

6.3 Result Analysis

The instances in Table 1, 2, 3 are then put into the proposed tabu search algorithm. The program is coded in C++ under Linux. All instances are tested on the computers with the same configuration: AMD 2.4G CPU and 16GB MEM.

For comparison purposes, the instances are also solved by CPLEX, a state-of-the-art mathematical solver. CPLEX v.10.1.1 is applied, and the implementation is coded and tested under the same programming language and computational environment as the tabu search program. The CPLEX solver stops either when the optimum is found or when we reach the maximum CPU time (t = 10 hours = $36,000$ sec).

Numerical results of the 3 instance sets are reported in Table 5, 6, 7 respectively. Comparison tables are divided into 3 groups. The first group is the CPLEX solution, including the best solution, solving time, and the optimal gap. The second group is the tabu search solution, which terminates at 20 non-improved local searches. Longer runs are preformed to further illustrate and analyze the heuristic's behavior. When the tabu search program stops at a maximum of 300 local searches, the results are shown in the third group. As the CPLEX solution, the tabu search always ends after reaching the maximum CPU time.

The conclusion that emerges from the first instance set is that for small instances, our algorithm is proven to be efficient to find an optimal or near-optimal solution within a short time. Compared with CPLEX, which solves all the instances, in 5 out of 9 instances the tabu search finds the optimal solution and the maximal optimal gap reported is only 1.77%.

Inst	CplexSol	time(s)	OptGap	TS.Sol	time(s)	CplexGap	TS.(300)	time(s)	CplexGap
s01	30003.32	1.89	0.00%	30003.32	3.19	0.00%	30003.32	172.70	0.00%
s02	47318.77	13.22	0.00%	47318.77	17.75	0.00%	47318.77	956.43	0.00%
s03	54106.88	11.46	0.00%	54106.88	16.95	0.00%	54106.88	1986.98	0.00%
s04	33670.44	100.16	0.00%	33670.44	25.16	0.00%	33670.44	497.15	0.00%
s05	57412.38	108.49	0.00%	57630.59	175.80	0.38%	57630.59	2401.62	0.38%
s06	88099.22	282.62	0.00%	88099.22	124.16	0.00%	88099.22	6225.63	0.00%
s07	46944.10	278.99	0.00%	47536.75	116.86	1.26%	46944.10	1046.03	0.00%
s08	92600.70	2601.76	0.00%	92906.69	324.99	0.33%	92906.69	4884.79	0.33%
s09	136493.00	1142.52	0.00%	138915.75	276.42	1.77%	138023.01	13056.51	1.12%

Table 5: Results on Instance Set S

The results displayed in Table 6 prove the robust performance on medium instances. Where CPLEX fails to prove the optimality, we found good solutions which are close to the best solution yielded by CPLEX, yet within a much shorter solution time. Moreover, with the increase of the instance, our results approach and start to catch up with the best CPLEX solution. When the optimal gap is considerable, we have a better opportunity to outperform CPLEX.

As expected, the solutions are further improved when we extend the solution process up to 300 local searches. In a CPU time of 10 hours, we have identified higher-quality solution for 4 out of 9 instances in set M. Under the same solution

Inst	CplexSol	time(s)	OptGap	N.S.Sol	time(s)	CplexGap	N.S.(300)	time(s)	CplexGap
m01	138356.83	t	3.17%	142824.76	5137.46	3.23%	142824.76	24615.90	3.23%
m02	206363.33	t	1.57%	211229.45	7315.00	2.36%	209084.01	t	1.32%
m03	205191.46	t	2.77%	206431.70	10589.93	0.60%	206431.70	t	0.60%
m04	181202.22	t	3.57%	183982.71	9760.38	1.53%	183982.71	t	1.53%
m05	197500.29	t	1.51%	202723.82	5677.39	2.64%	200088.64	t	1.31%
m06	213495.19	t	1.48%	215002.54	6432.69	0.71%	214119.76	t	0.29%
m07	208530.07	t	5.34%	215738.17	10504.70	3.46%	209485.39	t	0.46%
m08	261751.24	t	18.18%	232060.01	26189.75	-11.34%	232060.01	t	-11.34%
m09	286731.04	t	4.32%	284656.01	19428.81	-0.72%	284656.01	t	-0.72%

Table 6: Results on Instance Set M

Inst	CplexSol	time(s)	OptGap	N.S.Sol	time(s)	CplexGap	N.S.(300)	time(s)	CplexGap
x01	304781.94	t	12.67%	284528.55	t	-6.65%	284528.55	t	-6.65%
x02	337940.44	t	24.40%	291498.93	11540.75	-13.74%	290015.86	t	-14.18%
x03	282843.64	t	26.22%	244149.46	t	-13.68%	244149.46	t	-13.68%
x04	393135.36	t	19.44%	360178.80	t	-8.38%	360178.80	t	-8.38%
x05	456321.13	t	25.70%	397350.19	t	-12.92%	397350.19	t	-12.92%
x06	×	t	-	459215.88	t	-	459215.88	t	-
x07	×	t	-	555332.69	t	-	555332.69	t	-
x08	×	t	-	510350.36	t	-	510350.36	t	-
x09	×	t	-	639816.46	t	-	639816.46	t	-

Table 7: Results on Instance Set X

time, we either surpass the CPLEX result or end up with a solution rather close to the best known one.

Table 7 summarizes the behavior of the cycle-based tabu search on large instances, where our algorithm performs superiorly. The performance of CPLEX drops fast with the instance increase. With the same running time, we always find better solution than CPLEX with prominent advancement, with improvements that reach up to 14.18%. When CPLEX fails to provide any solution, as the last part of the table, feasible solutions are always presented by the heuristic algorithm. One notices, in most cases, we do not reach the 20 non-improved criteria and the procedure stops due to the maximal solving time, which is also an indication of the complexity of the problem.

After the tests on three instance sets, we conclude that the tabu search algorithm on the cycle-based neighborhood is able to yield very good solutions for the service network design model studied. Compared with the state-of-the-art mathematical commercial solver, the heuristic helps to locate the optimal solution for small-sized problems and near-optimal solutions for medium problems within a reasonable amount of computation time. More importantly, the algorithm provides good quality solutions for the large instances which are intractable for other methods.

7 Conclusions

In this research, we study the service network design problem to help the tactical planning in freight rail transportation. We analyze the inter-relations among scheduled service selection, blocking policy, make-up policy as well as traffic distribution, by introducing all these aspects into one compound model. The model is constructed on a 2-layer time-space network structure, in which we are able to describe the flow of blocks and cars at the same time. The mixed-integer formulation shows a linear objective with particular rail constraints. To the best of our knowledge, it is the first endeavor to integrate all these most important rail operations in a time-dependent context.

Due to the extremely large number of variables and constraints on the realistically dimensioned instance, no exact method is capable for industry applications. A tabu search heuristic is developed. The algorithm is based on a cycle-based neighborhood, which is proven to be efficient for the general network design problem. Numerical experiments demonstrate that the proposed algorithm is robust for providing near-optimal solutions within reasonable solving effort and outperforms CPLEX on large instances. The good solution on the largest experiment (with 10 yards, 14 time periods, 900 demands) makes it possible to be applied to rails (e.g. Canadian National Railway) to identify detailed regional solutions or aggregated national solution with rough schedules.

Other contributions behind the research derive from its generalization. The 2-layer network structure can be further applied to address the operations in related transportation modes, such as less-than-truckload, aviation and shipping lines where consolidation takes an important part, or intermodal transports where the long-haul legs can be viewed as blocks in rail. Also, we notice that the model is very complicated with combinatorial optimization in both layers. Both linear flow cost and fixed design cost are present in each layer. The proposed algorithm gives a first attempt for heuristically solving such problems on cycle-based neighborhood. And, the powerful performance suggests a direction for solving combinatorial problems with fixed-cost multi-commodity network design in multiple layers.

Further research opportunities lie in many directions. One avenue is to import multi-stop services into the model. With the intermediate stops in service, the blocks do not need to be transferred everywhere and, theoretically, a better operating plan can be achieved. A similar, but more complex network structure with more layers is necessary to describe the flows of even more objects, and the complete model will have more variables and be much harder to solve. Another research avenue consists of enhancing the proposed algorithm. To make it more compatible with larger instances, distributed system or parallel computation make intriguing and challenging research approaches.

Acknowledgments While working on this project, 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 and by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec.

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.

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

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. 1999. Long-haul freight transportation. R. W. Hall, ed., *Handbook of Transportation Science*. Kluwer, Norwell, MA.

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., M. Gendreau, J. M. Farvolden. 2000. A simplex-based tabu search method for capacitated network design. *INFORMS J. Comput.* **12**(3) 223–236.

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., 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.

Farvolden, J. M., W. B. Powell. 1994. Subgradient methods for the service network design problem. *Transportation Sci.* **28** 256–272.

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.

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.

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.

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.

Powell, W. B. 1986. A local improvement heuristic for the design of less-than-truckload motor carrier networks. *Transportation Sci.* **20**(4) 246–257.