Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, 2015, *9*, 243-252

Open Access

Graph Partition Based Decomposition Approach for Large-Scale Railway Locomotive Assignment

Xin Zhang[1,*], Wenting Mo[2], Baohua Wang[2], Fengjuan Wang[2] and Peng Gao[2]

[1]*Automation Department, Tsinghua University, Tsinghua University, Beijing, 100083, P.R. China;* [2]*IBM Research - China, Building 19 Zhongguancun Software Park, Beijing, 100193, P.R. China*

Abstract: Locomotive assignment is a classic planning problem in railway industry. Different models and algorithms (*e.g.*, network flow model, MIP model, adaptive dynamic programming) are proposed to solve this problem and have got remarkable progress. In practice, the locomotives are usually in the size of hundreds to even more than a thousand. So assigning individual locomotives to trains on railway network will result in too huge modeling space for the problem to be solved. Hence most existing research work takes a trade off to concern the types (or type combinations) of locomotives, which are usually in the size of tens at the most. This impedes the application value for being unable to consider the availability and maintenance requirement of individual locomotives. In this paper, a novel path-based MIP model is proposed for modeling the locomotive assignment problem. By adopting graph partition method on the space-time network, the original problem is decomposed into inter-connected multiple sub-problems. Then an iterative MIP solving algorithm is devised to find the near optimized solution for the original problem. The approach proposed in this paper has been validated in a railway bureau in China. The experiment shows that the approach has superior advantage in both scalability and performance with reasonable cost of objective value.

Keywords: Decomposition, graph partition, locomotive assignment, MIP.

1. INTRODUCTION

Locomotive planning and scheduling are very important for railway operations. It's critical to define an efficient locomotive utilization schedule that can provide sufficient pulling power for transportation requirement and at the mean time avoid inefficient utilization of resources. Generally speaking, the locomotive planning and scheduling are a process to assign locomotives to trains according to trains' pulling demand as well as locomotives' availability. Since the locomotive assignment problem is tightly coupled with physical railway network topology concerning certain planning period, it arise as a combinatorial optimization problem which has attracted a lot of research interest in past decades. Beside the network nature, the problem usually accompany with locomotive utilization rules, *e.g.*, at some section of the railway network, type A trains need to be joint pull by two model x locomotives and one model y locomotive. It's very challenging to find optimum solution when there are hundreds of locomotives to be assigned to hundreds of trains.

Previous research work has proved that the locomotive scheduling optimization problem is a NP-hard problem [1]. Different models and algorithms (*e.g.*, network flow model [2], mixed integer programming model (MIP) [3], adaptive dynamic programming [4]) are proposed to solve this problem and have got remarkable progress. Most of existing work considers the assignment on the locomotive type basis. They avoid concerning each individual locomotive either because it's unnecessary for long-term plan or for simplify the problem to avoid variable explosion. While in practice, different locomotives, even if they are of the same type, may have totally different assignment consideration because of their individual status, including available time, maintenance requirement, fueling demand, etc. So it's important to address this problem for generating more effective locomotive plan for daily execution. To the best of our knowledge, there is only limited work [5] that has touched this problem on small problem size. Even a medium size locomotive assignment problem can cause too many variables and constraints for any MIP solver to handle if concerning individual locomotives.

In this paper, we focus on the locomotive assignment problem on a large-scale railway network with hundreds of locomotives and more than a thousand trains. More specifically, we aim to develop method for generating optimized solution which can take individual locomotives' status into model consideration. We model the problem as a MIP model regarding locomotives' candidate working paths as major decision variables. Motivated by the observation from the space-time network of the locomotive assignment problem, a decomposition approach is proposed by leveraging graph partition technology. Through that approach, a global optimization problem is transformed into multiple inter-connected sub problems. Then an iterative MIP solving algorithm is devised to find the near optimized solution for the original problem. The approach proposed in this paper has been validated in a railway bureau in China. The experiment shows that the approach has superior advantage in both scal-

1874-110X/15

ability and performance with reasonable cost of objective value.

The remaining of this paper is as following. We first introduce related work in the literature review section. In the approach section, we come up with a formal definition of path-based MIP model after introducing the locomotive assignment problem. Then a novel graph partition based decomposition method and algorithm is presented and corresponding iterative optimization algorithm is described. Base on the devised approach, we present the experiment results and summarize our work.

2. LITERATURE REVIEW

A network flow model was proposed to solve the simultaneous assignment problem for railway locomotives and cars [2]. They used benders' decomposition based method to separate the model. Instances with a weekly schedule from VIA Rail Canada were tested in the paper. Multiple types of locomotive were assigned to a fixed timetable [3]. Branch and cut method was developed to solve the integer programming model. The model and algorithm was tested by actual data set from Canadian national railway company. Multi-depot locomotive assignment problem with time windows was solved by hybrid genetic algorithm [5]. 15 completely random generated instance problems indicated that this algorithm was efficient and solved the problem in a polynomial time. Light travel was specifically handled when optimizing locomotives [6]. They proposed an integer programming model to solve the problem. However, algorithm was not presented and the methodology may be more suitable for small area. The same problem as [3] was investigated and a backtracking mechanism, which can be added to the branch and price approach, was also presented [7]. Testing result showed the methodology may not be faster than [3] but could observe the solution process tendency. Fuzzy optimization was used to solve the same problem as [5, 8]. The trains should be served with appropriate locomotives in pre-specified hard/soft fuzzy time windows. However, there was no testing result to show the advantage of applying fuzzy theory. In summary modeling the locomotive assignment problem regarding individual locomotives' feasible working paths has not been fully addressed yet.

3. APPROACH

In this section, we firstly introduce the locomotive assignment problem in detail. Base on that, a formal MIP model taking locomotive feasible path as major variable is defined. After that, the approach that decomposes the global MIP problem into inter-connected sub MIP problems is illustrated. And the algorithm that iteratively solves the sub MIP problems according to information of the inter-connection matrix is depicted.

3.1. Problem Description

As former mentioned, the locomotive assignment problem is a process to assign locomotives to trains based on the consideration of railway network topology. The assignment needs to consider the matching between pulling demand and supply: At the demand size, each train is defined with its train type, departure yard, departure time, arrival yard, arrival time, and total weight, etc. At the supply size, each locomotive has its type, initially available time and available yard, its pulling capacity, current fuel or energy level, and maintenance demand, etc. In the output of the locomotive assignment, when a locomotive is assigned to one or multiple trains, it means the locomotive will pull (or deadhead on) those trains in sequence according to their time. In another way, we call this as the locomotive's working path. To achieve operation efficiency, a good locomotive plan shall be able to pull more trains with fewer locomotives. This comes to be the major objective of the locomotive assignment problem. Besides, there are usually certain locomotive utilization rules need to be complied, *e.g.*, at some section of the railway network, type A trains need to be joint pull by two model x locomotives and one model y locomotive. In general whether a locomotive can be assigned to pull a train depends on the following three elements: 1) Availability: if the locomotive is available at the yard before the train departs. 2) Fuel amount: if the locomotive has enough fuel for pulling the train. 3) Locomotive type: if the locomotive is of a locomotive type allowed for pulling the train. When all the conditions are satisfied, the locomotive becomes a candidate that can be assigned to the train, or in another way, the train is regarded a candidate in the locomotive's feasible paths.

Previous work mostly models the problem taking the number of locomotives (of certain type) assigning to a train as major decision variable. This approach on the one hand has the benefit for controlling the problem size, on the other hand has its weakness in considering the difference of individual locomotives. While in practice, different locomotives can have dramatically different feasible paths only for the difference in available time and fuel level. Maintenance requirement is another factor that can also impact this issue: if a locomotive plans to be maintained at depot A starting from Time X, then it shall be deliberately assigned to a train which arrives at yard around A before Time X. There are more other factors, such as some locomotive switching rule or turn around policy at a yard which requires the model be more flexible to take these factors into consideration.

So motivated by this requirement, we proposed a path-based MIP model. It regards the full set of all locomotives' feasible paths as the major decision variables. In this way, individual locomotives' characters and requirement, as well as some complex in-yard operation rules can be handled in the path set generation process. In the following we will formally define the path-based MIP model and further articulate the solving method and algorithm. The procedure of generating candidate paths can be an effective depth-first exhausted search, which will not be covered in detail.

3.2. Mathematical Model

Set notations:

L: All locomotives

TA: All trains

F_a: Set of feasible locomotive type on train a

P_l: Feasible path set of locomotive l

Constant notations:

C_{lp}: Cost of path p of locomotive l

RP_{la}: The horse power locomotive l can contribute on train a

$$\xi_{lpa} = \begin{cases} 1, \textit{if path p of locomotive l includes train a} \\ 0, \textit{otherwise} \end{cases}$$

w_1, w_2: Predefined weights of objectives

M: A big number

Decision variables:

$$y_{lp} = \begin{cases} 1, \textit{path p of locomotive l is selected} \\ 0, \textit{otherwise} \end{cases} \quad l \in L$$

$$x_{la} = \begin{cases} 1, \textit{locomotive l contributes power on train a} \\ 0, \textit{otherwise} \end{cases}$$

$a \in TA$

$$z_a = \begin{cases} 0, \textit{train a is satisfied} \\ 1, \textit{otherwise} \end{cases} \quad a \in TA$$

Constraints:

1. Each locomotive chooses exclusively one path from its candidate path set.

$$\sum_{p \in P_l} y_{lp} = 1 \quad \forall l \in L$$

2. A train is regarded as being satisfied if it is assigned enough horse power.

$$U_a - \sum_{l \in F_a} RP_{la} \cdot x_{la} \le M \cdot z_a$$

3. A locomotive either pull a train or deadhead on a train.

$$x_{la} \le \sum_{p \in P_l} \xi_{lpa} \cdot y_{lp} \quad \forall l \in L, \forall a \in TA$$

The objective of the path-based MIP model is to maximize the satisfied pulling demand (number of satisfied trains) with least total locomotive utilization cost:

$$\max imize\ w_1 \sum_{a \in TA} (1 - z_a) - w_2 \sum_{l \in L} C_{lp} \cdot y_{lp}$$

Note that in real application the locomotive utilization cost can be a mixture of many elements, such as locomotive cost, fuel cost, waiting cost, etc. Here without lost of model generality, we simplify the objective to the number of locomotive required, *i.e.*, $C_{lp} = 1$ if p_l contains one or more trains (otherwise $C_{lp} = 0$ since l doesn't work if it chooses p_l).

Light travel is a term of locomotive movement from one yard to another without pulling a train. Light travel shall be avoided unless a train has no locomotive to be assigned without the light travel. This model does not cover the light travel arrangement. One can resort to a post step to supply unsatisfied trains with locomotives through light travel after solving the MIP model. However this is out of the scope of this paper and will not be discussed in this paper.

With a rough test of the path-based MIP model, one can find its search space can be extraordinary huge: Assume in a medium size network with 2 or 3 hundreds of trains. In average each time there are 4 possible routes for each locomotive. And on each route there are 8 trains can be chosen to pull. And each locomotive pulls 2.5 trains per day. Then, the average number of feasible paths for a locomotive is $(4 \times 8)^{2.5} \approx 5800$. When the number of locomotives reaches several hundred, the decision variables of the model can exceed several millions. A combinatory optimization problem of this size is a disaster for deterministic MIP algorithms like Branch and Cut [9]. Thus, we proposed a graph partition based decomposition approach that will be introduced in the next subsection.

3.3. Graph Partition Based Decomposition Method and Algorithm (GPDMA)

It's a common understanding that when the scale of variables and constraints reaches multiple millions, it's hard for any commercial MIP solver to handle, even though acceleration techniques such as column generation [10] or Bender's decomposition [2] are considered. In order to reducing the problem size of the path-based MIP model, we trace the path explosion issue back to the network structure. It can be observed that the number of locomotives' feasible paths may increases exponentially along with the size of space-time network. Hence if we can decompose the network into multiple sub-networks with minimal connectivity lost, the original problem can be approximated as multiple inter-connected sub-problems. Each sub-problem can be solved in reasonable time using existing MIP solver. And through coordinating the solving process among the sub-problems taking their inter-connectivity into consideration, the near optimum result of the original problem can be generated.

In the following, the graph partition based decomposition method and algorithm will be introduced in detail. The method has the following steps: firstly construct the space-time network based on the network topology and train legs. Then formulate a graph model representing the connectivity among nodes in the Time-Space network by populating the locomotive information along the arcs of the network. Thirdly leverage graph partition method to split the graph into multiple sub-graphs with minimum cutting cost. After that, trace the cutting back to the space-time network and derive multiple sub space-time networks and the inter-connectivity matrix. For each sub space-time network, one MIP model instance can be generated. Lastly, resort to an iterative optimization algorithm to find the optimize solution for the problem.

3.4. Space-Time Network Construction

Space-time network is widely used for modelling network flow problems (*e.g.*, [2]). In this paper, we leverage the space-time network representation as the base for searching locomotive working paths. A sample space-time network is shown in Fig. (**1**). Formally we define it as:

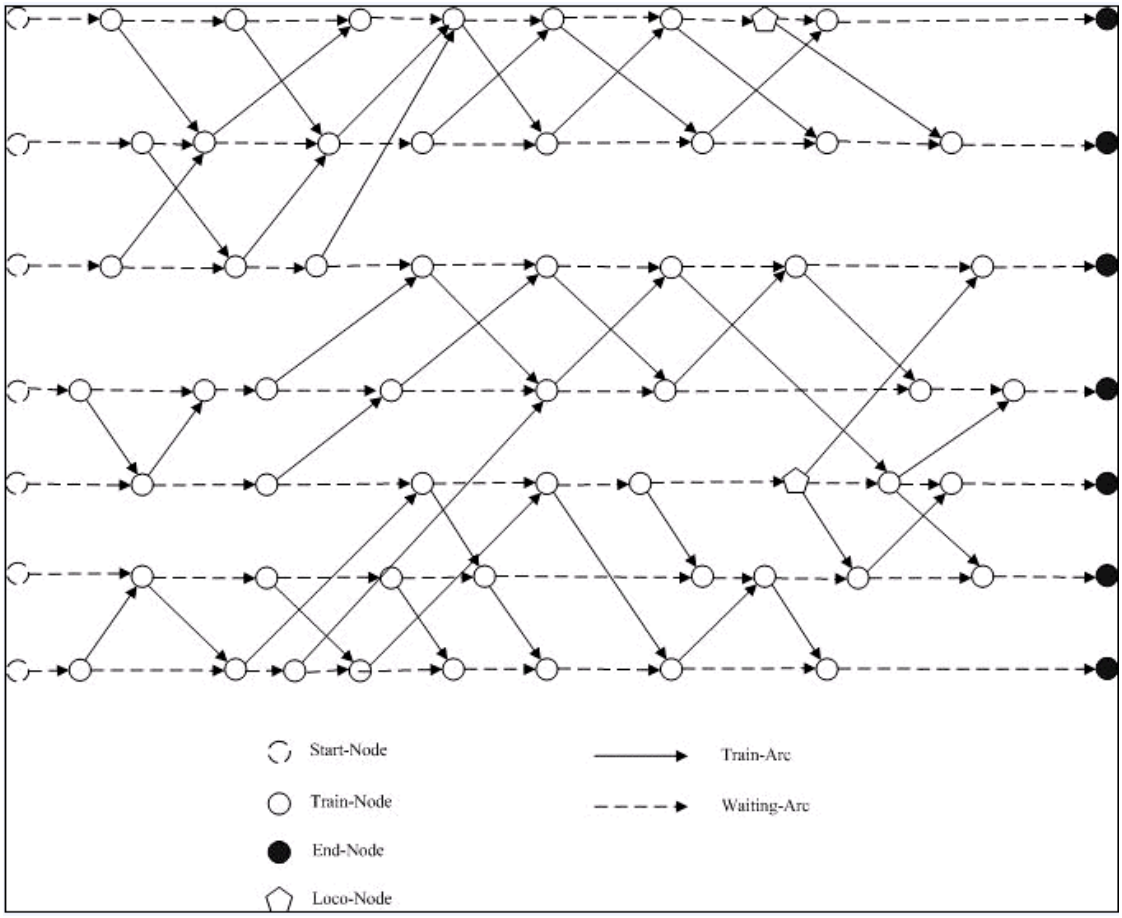

Fig. (1). A Sample space-time network.

$$G_N = (N_N, A_N) \tag{1.1}$$

A space-time network G_N is composed by a set of nodes N_N and a set of arcs A_N. There are four types of nodes N_N: train node $n_T \in N_T$, standing for the departure or arrival of a train leg at a yard on a predefined time; start node $n_S \in N_S$, which is on the plan's start time at each yard; end node $n_E \in N_E$, which is on the plan's end time at each yard; loco node $n_L \in N_L$, indicating that some locomotives are firstly available at the time and yard the node represents. Normally, a locomotive can get available at any node, and flows through the network to an end node. Arcs A_N have two types: train arc $a_T \in A_T$ connecting two train nodes and representing a train leg; waiting arc $a_W \in A_W$ connecting two temporally adjacent nodes at the same yard. The locomotive flow on the space-time network needs to cover train arcs A_T as much as possible in a cost effective way (indicating by the total number of assigned locomotives).

It is straightforward to construct a space-time network in the following steps: Firstly put yards in vertical and plan period as horizon. For each yard, create one start node at the plan begin time and one end node at the plan end time. For each train leg, create two train nodes at the train's departure and arrival yards and time, and connect them with a train arc. For each locomotive, put its number in the node representing its first available place. If no proper node exists, add a loco-node according to the locomotive available information. In the end, at each yard, chain up all the nodes on the yard according to their time sequence by waiting arcs.

One shall notice that in the space-time network the number of locomotives is not balanced for each node on its income arcs and outcome arcs. More specifically, those nodes with locomotive initially available will have more outcome locomotives than income locomotives, and some other nodes where locomotives sink in will have more income locomotives. And also, since there is no predefined source node and sink node indicating the two poles of the network, one can not resort to the maximum flow algorithm for partition the network.

3.5. Graph Model Formulation

We formulate a graph to model the connectivity of the space-time network. The graph model can be generated by inferring the locomotive flow on the space-time network. The key equation is as following:

$$l^n_{i,w} + l^n_{i,t} + l^n_{av} = l^n_{o,t} + l^n_{o,w} \tag{1.2}$$

$$\forall n \in N_N - N_E$$

For any node n except end nodes in N_E, its number of income locomotives, including income from waiting arc $l_{i,w}^n$ and from train arc $l_{i,t}^n$, plus the number of locomotives initially available on node n l_{av}^n, shall equal to the number of outcome locomotives on node n, which is the sum of outcome locomotives on train arc $l_{o,t}^n$ and on waiting arc $l_{o,w}^n$.

The graph model is formally defined as:

$$G = (V, E) \tag{1.3}$$

Each node in space-time network G_N is directly mapped a vertex in graph G. And each arc in G_N can mapped to an edge in G with the following algorithm to infer the weight of the edge.

Algorithm 1: Graph model generation(GraphGen) algorithm.

function GraphGen (N_N, A_N, L_T, L_{AV})

/* L_T is the list of required locomotive number on each train arc; L_{AV} is the list of initially available locomotive number on each node. */

$V \leftarrow \varnothing, V_d \leftarrow \varnothing, E \leftarrow \varnothing$

for each $n_i \in N_N$ **do**

new a vertex v_i, initial its attributes (initial available w_{av}^{vi}, in-wait $w_{i,w}^{vi}$, out-wait $w_{o,w}^{vi}$, in-train $w_{i,t}^{vi}$, out-train $w_{o,t}^{vi}$) as: $w_{av}^{vi} \leftarrow l_{av}^{ni}; w_{i,w}^{vi} \leftarrow 0; w_{o,w}^{vi} \leftarrow 0; w_{i,t}^{vi} \leftarrow 0; w_{o,t}^{vi} \leftarrow 0$

$V \leftarrow V \bigcup \{v_i\}$

if ($n_i \notin N_E$) **then** $V_d \leftarrow V_d \bigcup \{v_i\}$

for each $a_j \in A_N$ **do**

new an edge e_j connecting mapped vertexes (vs, ve)

if ($a_j \in A_T$) **then**

$w_{ej} \leftarrow l_{aj}; w_{o,t}^{vs} \leftarrow w_{o,t}^{vs} + w_{ej}; w_{i,t}^{ve} \leftarrow w_{i,t}^{ve} + w_{ej}$

else

$w_{ej} \leftarrow NA; w_{o,w}^{vs} \leftarrow NA; w_{i,w}^{ve} \leftarrow NA$

$E \leftarrow E \bigcup \{e_j\}$

while ($V_d \neq \varnothing$)

find a v_d in V_d that $w_{i,w}^{vd} \neq NA$

get v_d's out waiting edge e_d, and the other vertex v_e of e_d

$$w_{o,w}^{vd} \leftarrow w_{i,w}^{vd} + w_{i,t}^{vd} + w_{av}^{vd} - w_{o,t}^{vd}; w_{ed} \leftarrow w_{o,w}^{vd}; w_{i,w}^{ve} \leftarrow w_{o,w}^{vd}$$

$$V_d \leftarrow V_d - \{v_d\}$$

The space-time network is acyclic since each arc has an earlier time in its start node than in its end node. And also each node has at most one out waiting arc (a waiting arc has the node as its start node). So the GraphGen algorithm is deterministic.

As each loop in the GraphGen algorithm scans each node or arc at most once, so the computation complexity of GraphGen algorithm is O(n).

Fig. (**2**) shows a sample of weighted graph in the space-time network representation. The sample reveals some clue for decomposition: there exist some cliques in the graph which are intensively connected internally and weakly linked with other parts of the graph.

3.6. Graph Partition

The motivation here is to decompose the original space-time network into multiple inter-connected sub-networks. By that means the task to find the optimum solution of a global MIP problem, which is usually infeasible with the increase of problem size, is transformed into the task to collaboratively optimize multiple smaller size MIP problems. Intuitively, decomposing a connected network will certainly cause some information lost. So it's critical to minimize the total weight of arcs that are cut for disconnecting the network. With minimized lost of connectivity, we can expect that the optimization result in the transformed space is able to approximate the original problem quite well.

In this paper, we resort to graph partition method to decompose the network with minimum lost of information. Graph partition methods are widely used in machine learning, pattern recognition, as well as data mining field [11-13]. There are many variations of graph partition methods, mostly on undirected graphs [14-16], and a few on directed graphs [17]. Here we adopt the normalized cut [18] approach on Laplace matrix which is both computational efficient and can balance the size of parts in the mean time to minimize the cutting edges.

To apply normalized cut, a matrix W is derived from graph model G representing the connectivity between each pair of vertex, where

$$W_{i,j} = \begin{cases} w_e, & \exists e \quad e = (v_i, v_j) \wedge e \in E \\ 0, & otherwise \end{cases}$$

Since W is an asymmetric matrix, it can be transformed into a symmetric one $M \leftarrow W + W^T$. Then the Laplace matrix can be constructed $L \leftarrow D - M$, where D is the diagonal matrix with $D_{ii} \leftarrow \sum_j M_{ij}$ [19]. The normalized Laplace matrix is $L_n \leftarrow D^{-1/2} L D^{-1/2}$. By solving the Eigen values and

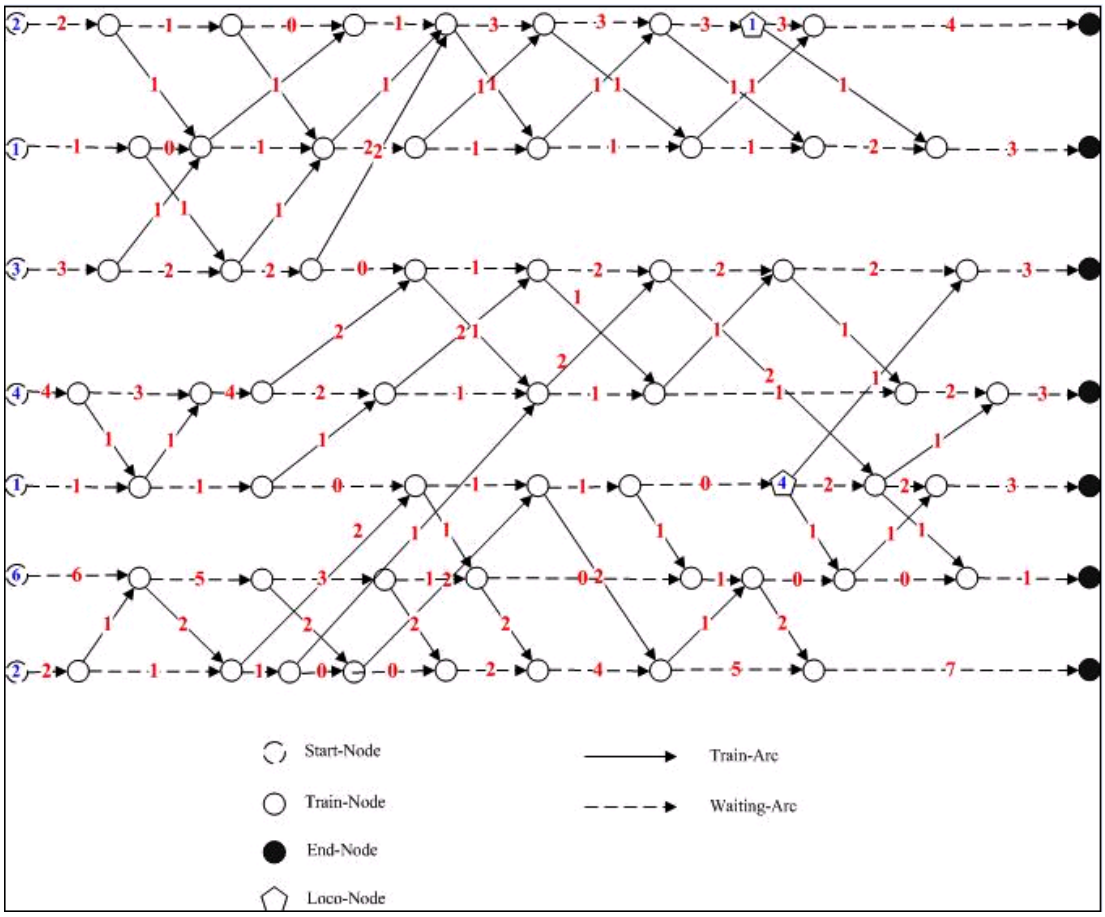

Fig. (2). A Sample of weighted graph.

corresponding Eigen vectors of L_n, the algebraic connectivity of a graph is disclosed. By sorting the Eigen values in ascendant order $0 = \lambda_1 \le \lambda_2 \le \lambda_3 \le \cdots \le \lambda_n$ and picking Eigen values start from λ_2, the sign of element in corresponding Eigen vector indicates which group a vertex shall belongs.

By normalized cut, the vertex set of a graph is partitioned into several groups of vertexes. Normalized cut takes $O(n^3)$ operations to compute all Eigen values and Eigen vectors, where n is the number of vertexes in the graph. The algorithm is quite efficient in handle input with thousands of vertexes.

Fig. (**3**) shows the partition result of the sample formerly introduced.

It's very important to determine the proper number of groups to be cut into. Though it's more efficient in solving the optimization problems in smaller size, it's more risky to bias from the original problem for the sake of lost too much connectivity information. The GraphPartition algorithm containing the cut strategy is as following:

Algorithm 2: Graph partition (GraphPartition) algorithm.

function GraphPartition ($G, Threshold$)

Compute the normalized Laplace matrix L_n from G

Compute Eigen values and vectors $\lambda_i, \upsilon_i \quad i \in (1...n)$ for L_n.

$\Omega \leftarrow \{V\}$

$x \leftarrow 2$

while ($\Omega \neq \varnothing$)

Partition each vertex group in Ω into two sub-groups according to υ_x, get $V_1 ... V_K$.

$\Omega \leftarrow \varnothing$

for each V_k **do**

Test the sum of variables and constraints m by searching locomotive feasible paths on the sub-network mapped by V_k

if $m \ge Threshold$ **then**

$\Omega \leftarrow \Omega \cup \{V_k\}$

else

if $\lambda_{x+1} \le 0.01 \wedge \lambda_{x+1} \le \lambda_x * 1.1$ **then**

$\Omega \leftarrow \Omega \cup \{V_k\}$

$x \leftarrow x + 1$

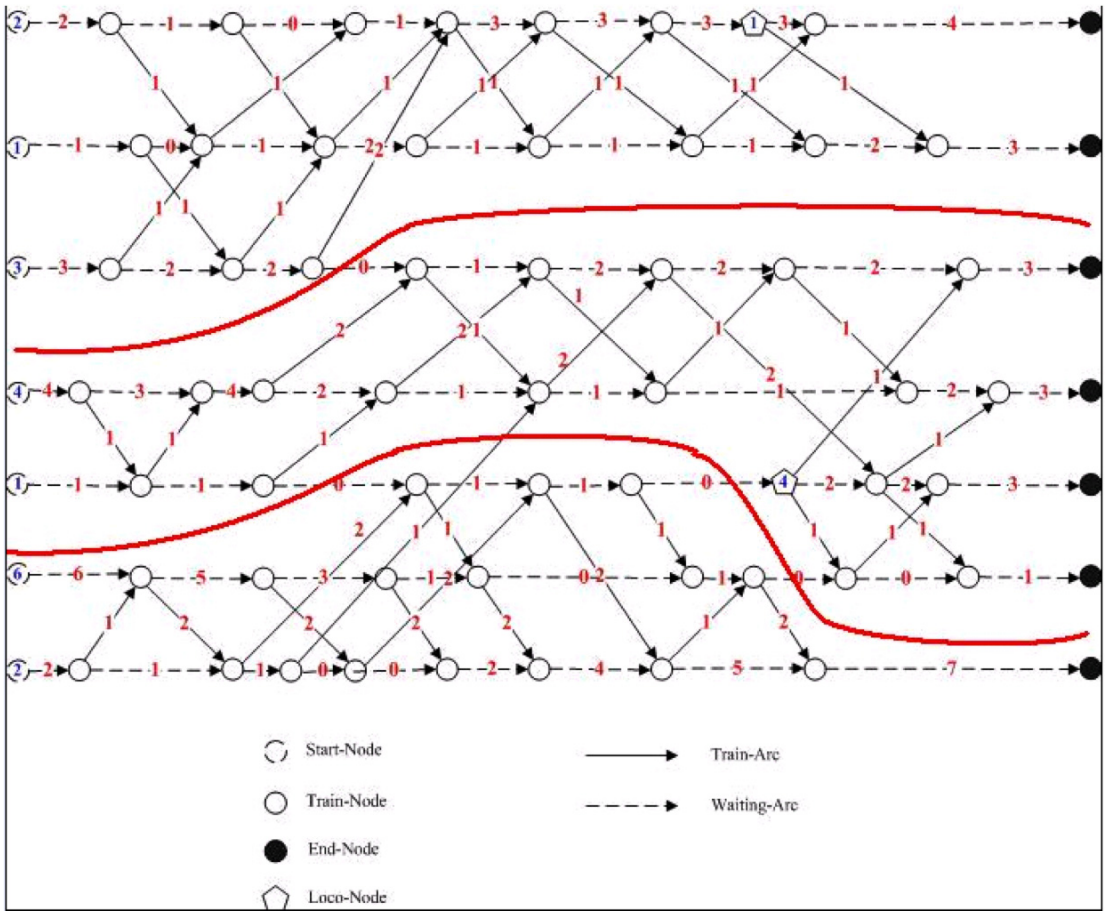

Fig. (3). A partition of space-time network.

The input Threshold is the estimated capacity of the MIP solver to be used, denoting by the sum of variables and constraints that the MIP solver can safely handle. So the Graph-Partition algorithm continue to cut a sub-network when its size is too big to be handled by the MIP solver or there is little lost of connectivity (indicating by the value of λ). The computation complexity of GraphPartition algorithm is also $O(n^3)$.

3.7. Network Model Re-generation

Given V_k $k \in (1 \ldots K)$ are the sub-groups of vertexes generated through normalized cut, the network will be re-constructed by a set of sub-networks $G_N^k = \left(N_N^k - A_N^k\right)$ and a graph model G_c representing the inter-connectivity among sub-networks. For each V_k, its corresponding sub-network G_N^k can be built in the following steps: 1. map back from vertexes V_k to nodes $N_k^{'}$; 2. create arc set $A_N^k \leftarrow \left\{a \middle| a's\ startnode \in N_k^{'}\right\}$; 3. create node set $N_N^k \leftarrow N_k^{'} \cup \left\{n \middle| n = e.endnode, e \in A_N^k\right\}$. Note that the sub-networks are a partition of arcs and they can have overlapped nodes. So from the path-based MIP model point of view, the network decomposition is at the cost of localizing the locomotive path searching if it covers the overlapped nodes. That's exactly the reason for minimizing the total weight of edges that are cut during the graph partition.

The inter-connection graph G_C can be built in the following steps: 1. define the vertex set by $V_C \leftarrow \{v_{C1}, \cdots, v_{CK}\}$ $V_C \leftarrow \{v_{C1}, \ldots v_{CK}\}$ where each vertex corresponding to a sub-network; 2. define the weight between vertexes $W_{Cij} \leftarrow \sum W_a$ where $a.startnode \in N_N^i$ and $a's\ startnode \in N_N^i$; 3. fix the edge set $E_C \leftarrow \left\{e_{ij}, W_{eij} \leftarrow W_{cij} \middle| W_{cij} > 0\right\}$.

3.8. Iterative Optimization Algorithm

In this sub-section, an iterative optimization algorithm will be developed based on G_N^k, G_C and W_C. W_C is the weight matrix of G_C. At the lower layer, each G_N^k is modeled as a MIP problem and is solved separately. At the higher layer, G_C conveys the dependency information among those sub-problems. Therefore we leverage G_C and W_C to control the sequence and collaboration among those sub-problems. We regard W_{Cij} as the number of locomotives that

Table 1. Experiment Dataset.

DataSet	Experiment Data	Number of Trains	Number of Candidate Locomotives
Data Set 1	DS1_Small	69	60
	DS1_Medium	250	217
	DS1_Large	808	601
Data Set 2	DS2_Small	105	100
	DS2_Medium	406	386
	DS2_Large	876	601

G_N^i inputs to G_N^j. So before G_N^i has been solved, W_{Cij} is uncertain to G_N^j. Definitely one shall start from sub-problem which has less number of uncertain locomotives. So the order of optimization process can be determined by estimating the dynamic matrix of W_C. In each step when a sub-problem has been solved and the locomotives on its outcome arcs has been determined, the W_C will be updated accordingly. One may notice that G_C could be a cyclic graph. So there shall enable multiple rounds of iterations till the locomotive uncertainty among sub-problems would not be improved any more. The detail algorithm is shown in Algorithm 3.

Algorithm 3: Iterative optimization (IterOptim) algorithm.

function IterOptim ($G_C, W_C, G_N^k \quad k \in (1...K)$)

do

$C \leftarrow \{1...K\}$

do

find a k that $\min_k \sum_i W_{Cik} \qquad k \in C$

search locomotive feasible paths on G_N^k

build MIP model instance for G_N^k

solve the MIP model of G_N^k

$C \leftarrow C - \{k\}$

update the values in the k-th row of matrix W_C, for each outward arc from G_N^k to $G_N^{k'}$, if its locomotives are fixed in solving the MIP, then $W_{Ckk'} \leftarrow W_{Ckk'} - l$ where l is the number of fixed locomotives.

while ($C \neq \varnothing$)

while ($\sum_i \sum_j W_{Cij} = 0$ or $\sum_i \sum_j W_{Cij}$ has no more improvement between two iterations)

4. NUMERICAL EXPERIMENT

The graph partition based optimization model and algorithm described in this paper has been implemented in Java code using IBM iLog CPLEX as the MIP solver. The software has been validated with a railway bureau in China using the production data. In the following, we will introduce the experiments in detail, including the experiment dataset, experiment environment and settings, as well as experiment result.

This railway bureau operates a railway network which has its lines more than 6700 kilometers long and has more than 400 yards. The number of daily trains is around 1000. And there are more than 800 locomotives available and more than 600 of them need to be planned for daily operation.

The experiment dataset information is listed in Table **1**.

The experiments are performed on a PC server with 2 Intel CPUs (2.93GHz) and 4G RAM with windows system installed. The IBM iLog CPLEX version is 12.3. According to our test, the safe problem size is 2,000,000 of variables and constraints. So we set the Threshold of GraphPartition algorithm to 2,000,000. There are two predefined weights of objectives in the MIP model (w_1, w_2). In real world, satisfying a train is always more important than occupying one locomotive. So we set w_1 as 2 and w_2 as 1.

The baseline that our algorithm compares with is the global path-based MIP model without decomposition (denote as gpMIP algorithm). We implement it in the same language and run it in the same environment. Considering gpMIP algorithm may not be able to output any result when the problem size get to middle or large size, we adjust the gpMIP algorithm to trim the size of locomotive path when it gets too big. We denote the adjusted algorithm as gpMIP-trim algorithm.

The experiment result is shown in Tables **2** and **3**.

From Table **2**, we can see that the GPDMA takes less running time than both gpMIP and gpMIP-trim on most experiment data sets. On the small size data sets (DS1_Small and DS2_Small), GPDMA requires similar running time as gpMIP and gpMIP-trim. While with the increasing of problem size, GPDMA shows its superior running efficiency than gpMIP. The result demonstrates GPDMA's advantage in

Table 2. Experiment result: Running time.

Experiment Data	gpMIP (s)	gpMIP-trim (s)	GPDMA (s)
DS1_Small	16	17	12
DS1_Medium	256	245	39
DS1_Large	N/A*	674	780
DS2_Small	52	53	18
DS2_Medium	300	154	21
DS2_Large	N/A*	560	720

* N/A means the instance is too huge to be solved (either out-of-memory or no solution generated).

Table 3. Experiment result: Performance.

Experiment Data	gpMIP			gpMIP-trim			GPDMA		
	T	L	O	T	L	O	T	L	O
DS1_Small	67	37	97	67	37	97	64	41	87
DS1_Medium	243	133	353	233	125	341	240	130	350
DS1_Large	N/A*			648	424	872	751	384	1118
DS2_Small	105	52	158	105	52	158	93	61	125
DS2_ Medium	401	163	639	341	176	506	390	159	621
DS2_Large	N/A*			690	442	938	771	403	1139

Notes: T: number of satisfied Trains; L: number of used Locomotives; O: objective value

scalability as its runtime time increases much smoother along with the problem size comparing to goMIP. Though gpMIP-trim also enables some acceleration by trimming the problem size, the result in the following table shows the performance is impacted.

Table **3** displays the performance result of each algorithm on every datasets. For each dataset that gpMIP achieve optimum, GPDMA approaches the result of gpMIP within 10% difference. On the medium data sets and large data sets where the gpMIP fail to find the optimum, GPDMA has superior performance than gpMIP-trim.

In summary, comparing with other method which does not leverage the structure information of the problem, GPDMA achieves higher efficiency and scalability with reasonable expense on performance.

5. CONCLUSION

In this paper, the locomotive assignment problem is investigated. We come up with a path-based MIP model to meet the fine-granularity planning requirement. By leveraging the network feature of this problem, we propose a method to decompose the global MIP model to accelerate the problem solving process. Experiment result shows that the partition process could reduce the total running time dramatically with encouraging optimization performance. The proposed approach has superior advantage in its scalability. It is an interesting topic to extend the model and framework to support light travel arrangement. We will explore further in our future research.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] B. Vaidyanathan, R. Ahuja, J. Liu and, L. Shughart, "Real-life locomotive planning: new formulations and computational results," *Transportation Research Part B*, vol. 42, no. 2, pp. 147-168, 2008.

[2] J. Cordeau, F. Soumis, and J. Desrosiers, "A benders decomposition approach for the locomotive and car assignment problem," *Transportation Science*, vol. 34, no. 2, pp. 133-148, 2000.

[3] K. Ziarati, F. Soumis, J. Desrosiers, and M. Solomon, "A branch-first cut-second approach for locomotive assignment," *Management Science*, vol. 45, no. 8, pp. 1156-1168, 1999.

[4] W. Powell, A. George, B. Bouzaiene-Ayari, and H.P. Simao, "Approximate dynamic programming for high dimensional resource allocation problems," In: *International Joint Conference on Neural Networks*, 2005, pp. 2885-2994.

[5] K. Ghoseiri, and S.F. Ghannadpour, "A hybrid genetic algorithm for multi-depot homogenous locomotive", *Applied Soft Computing*, vol. 10, pp. 53-65, 2010.

[6] C. Kuo, and G.M. Nicholls, "A mathematical modeling approach to improving locomotive utilization at a freight railroad," *Omega*, vol. 35, pp. 472-485, 2007.

[7] S. Rouillon, G. Desaulniers, and F. Soumis, "An extended branch-and-bound method for locomotive assignment," *Transportation Research part B*, vol. 40, no. 5, pp. 404-423, 2006.

[8] K. Ghoseiri, S.F. Ghannadpour, and A. Seifi, "Locomotive Routing and Scheduling Problem with Fuzzy Time Windows", In: *Transportation Research Board 89th Annual Meeting* 2010.

[9] M. Fischetti, J. González, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378-394, 1997.

[10] M. Desrochers, and F. Soumis, "A column generation approach to the urban transit crew scheduling problem," *Transportation Science*, vol. 23, no. 1, pp. 1-13, 1989.

[11] M. Szummer, and T. Jaakkola, "Partially labeled classification with markov random works," *Advances in Neural Information Processing Systems*, vol. 14, pp. 945-952, 2002.

[12] A. Kapoor, Y. Qi, and H. Ahn, "Hyperparameter and kernel learning for graph based semi-supervised classification," *Advances in Neural Information Processing Systems*, vol. 18, pp. 627-634. 2006.

[13] A. Blum, and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," In: P*roceedings of The 18th International Conference on Machine Learning*, 2001, pp. 19-26.

[14] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser, "Min-Cut Clustering," *Mathematical Programming*, vol. 62, no. 10, pp. 133-151, 1993.

[15] P. Chan, M. Schlag, and J. Zien, "Spectral k-way ratio cut partitioning," *IEEE Transactions on CAD-Integrated Circuits and Systems*, vol. 13, pp. 1088-1096, 1994.

[16] C. Ding, X. He, H. Zha, ang M. Gu, "A Min-Max Cut for Graph Partitioning and Data Clustering," In: *Proceedings of the 1st IEEE International Conference on Data Mining*, 2001, pp. 107-114.

[17] M. Meila, and W. Pentney, "Clustering by weighted cuts in directed graphs," In: *Proceedings of the 2007 SIAM International Conference on Data Mining,* 2007.

[18] J. Shi, and J. Malik, "Normalized cuts and image segmentation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905, Aug. 2000.

[19] F.R.K. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014