UNIVERSITY OF PIRAEUS

DEPARTMENT OF INDUSTRIAL MANAGEMENT AND TECHNOLOGY

POSTGRADUATE PROGRAM

LOGISTICS MANAGEMENT

DISSERTATION OF

MICHAEL PAPAIOANNOU

TRAIN SCHEDULING AND ROUTING USING SIMULATED ANNEALING ALGORITHM

SUPERVISOR

SOFIANOPOULOU STILIANI, PROFESSOR OF UNIVERSITY OF PIRAEUS

PIRAEUS 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

ΔΙΟΙΚΗΣΗ LOGISTICS

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΟΥ

ΜΙΧΑΛΗ ΠΑΠΑΪΩΑΝΝΟΥ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΔΡΟΜΟΛΟΓΗΣΗ ΤΡΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΠΡΟΣΟΜΟΙΩΜΕΝΗΣ ΑΝΟΠΤΗΣΗΣ

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ

ΣΟΦΙΑΝΟΠΟΥΛΟΥ ΣΤΥΛΙΑΝΗ, ΚΑΘΗΓΗΤΡΙΑ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΠΕΙΡΑΙΩΣ

ΠΕΙΡΑΙΑΣ 2016

Αφιερώνεται στη μνήμη του παππού μου Σταμάτιου Λουκά.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την καθηγητριά μου Κυρία Σοφιανοπούλου Στυλιανή, που με δίδαξε έναν νέο τρόπο σκέψης και μου έδωσε την ευκαιρία να επεκτείνω τους ορίζοντες μου.

ABSTRACT

In this dissertation, a work is presented using simulated annealing method in order to solve a train routing and scheduling problem. If these problems are solved separately the solution might not be optimum. In an attempt to overcome this obstacle both problems are solved simultaneously.

This has as side effect that the problem becomes much more difficult. The objective is to minimize the cost of train formation, the cost of idle time of wagons in stations waiting for trains and the cost of wagon classifications in shunting stations.

ΠΕΡΙΛΗΨΗ

Στη παρούσα εργασία χρησιμοποιείται η μέθοδος της προσομοιωμένης ανόπτησης με σκοπό να λυθεί το πρόβλημα του προγραμματισμού και της δρομολόγησης τρένων.
Εάν επιχειρήσουμε να λύσουμε τα δύο αυτά προβλήματα ξεχωριστά, η λύση που θα λάβουμε είναι πιθανό να μην είναι βέλτιστη. Σε μία προσπάθεια να υπερνικήσουμε αυτό το εμπόδιο προσπαθούμε να τα λύσουμε ταυτόχρονα, με αποτέλεσμα να αυξάνεται η δυσκολία του προβλήματος.
Οι στόχοι που τίθενται είναι οι εξής: Η ελαχιστοποίηση του κόστους σχηματισμού των τρένων, του κόστους που αντιστοιχεί στο χρόνο αδράνειας των βαγονιών στους ενδιάμεσους σταθμούς αναμένοντας την ανταπόκριση με τα επόμενα τρένα και τέλος του κόστους ταξινόμησης στους σταθμούς διαλογής.

Table of Contents

CHAPTER 1

1.1 INTRODUCTION

Despite the efforts of European policies to lead freight traffic to more green ways of transport, nothing seems to have changed. Between 2008 and 2013 as we can observe in figure 1, the share of railways in modal split remains the same at 12,3%. Although railways are one of the most environment friendly means of transportation, the

(*) Air and maritime cover only intra-EU transport (transport to/from countries of the EU) and exclude extra-EU transport.

problems seem to concentrate to economical issues. One of the most important factors for a customer when making a choice is price. Others that may follow are speed of service, frequency and reliability.

Figure 1: Freight transport in the EU-28 modal split based on five transport modes (% of total tonne-kilometres)

In most cases, companies' orders include a small number of wagons. In the direction that the cost will have to be as low as possible for the carriers to be competitive, these wagons must be pulled to a marshalling yard to be rearranged and consolidated with other wagons from different origins before being able to reach their destinations. On the other hand this action contains shunting costs that will have to be added to our objective function.

1. 2. NETWORK ROUTING MODELS: BLOCKING PROBLEM

A blocking policy is usually specified as follows: cars at yard i which are destined for yard j must be added to a block that will next be shipped to yard k. Cars in a block will not be reclassified until the block reaches its final destination. A blocking model thus places the emphasis on the movement of cars as opposed to the movement of trains. The solution of this model indicates the routing of freight through the network and the distribution of classification work among yards, but does not specify the trains to be run or the assignment of blocks to trains. Instead, an additional problem must then be solved to determine the routing of trains and their makeup.

1.3. ROUTING AND MAKEUP MODELS

Whereas blocking models indicate the routing of freight and the distribution of classification work among the yards of the network, routing and makeup models

determine the routing and frequency of trains and the assignment of blocks to trains. In routing and makeup models, the blocking policy may be either determined endogenously or given as an input. These models thus produce a complete train and freight routing plan. However, because they do not provide actual departure times for the trains to be run, an additional scheduling problem must be solved at a later stage.

1.4. COMPOUND ROUTING AND SCHEDULING MODELS

Routing and makeup models produce a transportation plan that completely describes the routing of freight, the set of trains to be operated and their respective frequency. But because these models do not take scheduling into consideration, it may be difficult to later find a timetable accommodating all scheduled trains and satisfying line and yard capacity. Hence compound models, which address both the routing and the scheduling aspects of freight transportation, can significantly help to improve service reliability and reduce costs.

1.5 THE MAIN GOALS OF TRAIN FORMATION PROBLEM

The main goals of the train formation problems are:

1. to minimize classifying operations in shunting stations
2. to minimize train formation costs
3. to minimize the idle time of wagons waiting for trains in shunting stations
4. to maximize the railroad track capacity for train movements to share almost equal wagon classification operations in all shunting stations

5. to yield the optimum scheduling for each wagon

Furthermore, the following strategic planning goals can be achieved:

- to develop the critical shunting stations with the high wagon classification operations
- to build new shunting stations if required
- to procure more locomotives if needed

1.6 PROBLEM DEFINITION

In order to represent a railroad network we can use a graph, where its nodes will represent shunting stations and the existing paths between them, the arcs. A train can move to the next station by selecting only one of the available arcs each time. It has to be stated that the allowed movement on the graph is from left to right and once a train has been used it cannot return to its origin station.

For example let us assume that we have a graph with four stations. If a train starts from station 1, it has the following paths available: (1-2),(1-3),(1-4).

If it starts from station 2 then it has to make a choice between arcs: (2-3),(2-4).

Last but not least our choices are limited when we begin from station 3. In this case the only arc left is (3-4).

When a train traverses an arc, a cost occurs. This can be also viewed as a time slot usage cost. Costs for traversing an arc are given below, but it is worth to mention that the cost related for traversing arcs (1,2) and (2,3) equals the cost passing arc (1,3). The difference lies at the idle time cost of wagons and locomotives at the intermediate station 2.

For the proposed model the following assumptions have been made:

- The unit of each consignment is a wagon.
- An upper limit exists on how many wagons a locomotive can pull for each arc depending on the gradient, the distance between signals, the train's couplers and the internal length of stations.
- The distances between stations are known and so the transit times of trains are pre-estimated.

1.7 REVIEW

Branch and price for a European variant of the railroad blocking problem, Robert Voll and Uwe Clausen

The Railroad Blocking problem (RBP) can be modeled as a multi commodity capacitated network design problem (MCNDP). The underlying problem of determining a sequence of yards for each railcar is a routing problem. While finding an optimal routing for all railcars the goal is to find the blocking network which is a subset of the original network. Operating a train induces high fixed costs, which dominate the transport costs.

So the cost function must focus on train costs rather than on costs per wagon. The decision problem can be **proven to be NP- complete.**

In this paper a branch and price approach is introduced for the problem under consideration, where specialized cuts are used, which are incorporated into the branching scheme.

A mathematical model for train routing and scheduling problem with fuzzy approach, Amin Jamili

In classical linear programming the violation of any constraint renders the solution infeasible. But in real world, applicable, cases the role of constraints can be different. In real life problems the decision maker might accept small violations of constraints but might also attach different degrees of importance to the violations of different constraints. In this paper the parameter of the maximum allowable wagons hauled by a locomotive in an arc, is supposed to be imprecise. As the objective function value assumed to be crisp, the Werner's approach can be applied. The specified problem is a pure binary one and merges the scheduling problem with the routing problem, each of which is **known to be NP – hard**. The proposed heuristic method is as follows: the problem is divided in two individual sub-problems, one is routing, also known as train formation problem and the other is train scheduling problem.

CHAPTER 2

2.1 OPTIMIZATION

Several optimization methods exist and can be categorized into two main groups. Those using analytic type of solutions, such as Lagrange or Newton's methods and those employing more algorithmic type of methods, such as Simplex, Dijkstra's algorithm, dynamic programming, particle swarm optimization etc. Among these methods is simulated annealing.

At Los Alamos in 1953, Nicholas Metropolis (Νικόλαος Μητρόπουλος), a Greek – American physicist, developed a modified Monte Carlo scheme. His main idea was that: **instead of choosing configurations randomly, then weighting them with $exp(\frac{-E}{k_T})$, we choose configurations with probability $exp(\frac{-E}{k_T})$ and then weight them evenly.**

2.2. SIMULATED ANNEALING

In physics, the term annealing denotes a physical process where a solid is heated inside a thermal reservoir, increasing the temperature to a maximum value (melting point) in which all the particles of the solid are randomly distributed during the transition phase to liquid form.

Subsequently it is cooled by slowly lowering the temperature of the thermal reservoir. Thereby at the point where the energy takes its lowest value, all the particles are in order forming lattice.

Under the precondition that the maximum temperature will be high enough and the

cooling will be carried out enough slowly so as the solid wont revert to metastable state.

Starting from the maximum value of the temperature, the cooling phase of the annealing process can be described as follows.

At each temperature value T, the solid is allowed to reach at thermal equilibrium, characterized by a probability of being in a state i with energy Ei given by the Boltzmann distribution:

$$P_T(X = i) = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{k_B T}\right) \quad (1.1)$$

$$Z(T) = \sum_j \exp\left(\frac{-E_j}{k_B T}\right) \quad (1.2)$$

Where Z(T) is the partition function, T is the thermodynamic temperature and k_B is the Boltzmann constant.

As the temperature is lowered Boltzmann distribution concentrates on the states with lowest energy and finally when the temperature approaches zero only the minimum energy states have a non-zero probability of occurrence.

2.3 METROPOLIS ALGORITHM

To simulate the evolution towards thermal equilibrium of a solid for a constant value of temperature T, (Metropolis et al., 1953) used a Monte Carlo method, that can provide us sequences between the previous and the following states of the solid in the following manner.

Given the current state of the solid, given by the positions of its particles, a random change (perturbation) is made by transposing randomly a particle.
If the energy difference $\left(E_j - E_i\right) \leq 0$ is less than or equal to zero, between the current and the perturbed one, it is implied that the change had resulted to a lower energy state, so state j is accepted as current state.

If $\left(E_j - E_i > 0\right)$ then the probability to accept the perturbed state is given by the Boltzmann factor: $exp\left(\frac{-\Delta E}{k_B T}\right)$

This acceptance rule for new states is known as the **Metropolis criterion**.
Following that criterion the system eventually is transitioned to thermal equilibrium after a large number of perturbations the probability distribution of the states, approaches the **Boltzmann distribution**.

In statistical mechanics this Monte Carlo method is known as Metropolis algorithm and can be used for generation of neighboring solutions for optimization problems.
In this occasion solutions correspond to states of the solid, cost function to energy and the control parameter to the temperature.

2.4 PSEUDO CODE OF SIMULATED ANNEALING ALGORITHM

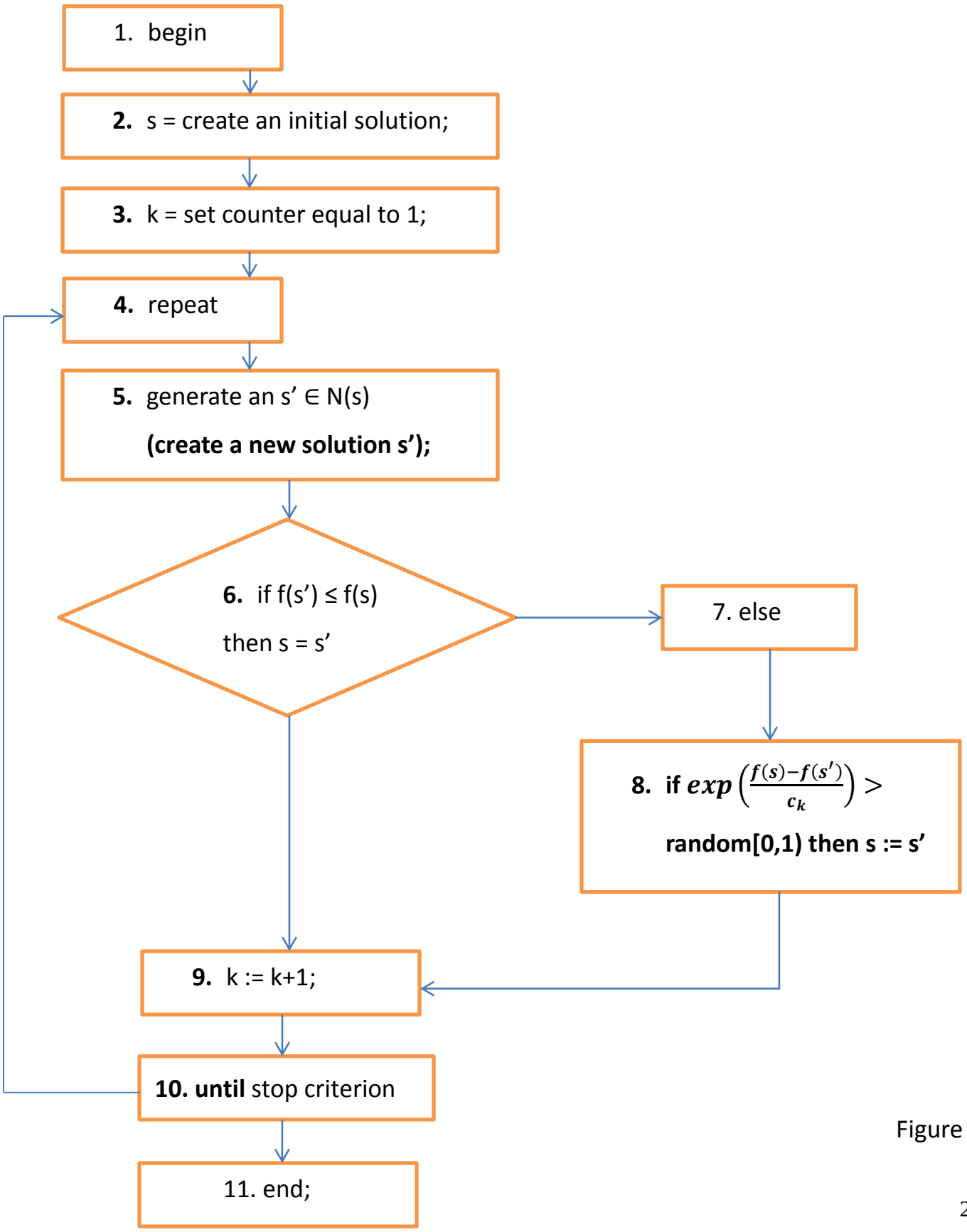

Figure 3

2.5 MARKOV CHAINS AND THEIR CONNECTION WITH SIMULATED ANNEALING

A stochastic process is defined to be an indexed collection of random variables {Xt}, where the index t runs through a given set T. Often T is taken to be the set of non-negative integers, and Xt represents a measurable characteristic of interest at time t. Stochastic processes are of interest for describing the behavior of a system operating over some period of time.

Markov chains:
a stochastic process {Xt} is said to have the Markovian property
if P{Xt+1 = j | X0 = k0, X1 = k1, … ,Xt-1 = kt-1, Xt = i } = P{Xt+1 = j | Xt = i },
for t=0,1,... and every sequence i, j, k0, K1,..., kt-1.

The Markovian property applies when the conditional probability of any future event, given any past event and the present state Xt =i, is independent of the past event and depends only upon the present state.

A stochastic process {Xt } (t = 0, 1,...,) is a Markov chain if it has the Markovian property.

The conditional probabilities P{Xt+1 = j | Xt = i } for a Markov chain are called transition probabilities. If, for each i and j,P{Xt+1 = j | Xt = i } = P{X1 = j | X0 = i },
for all t= 1, 2,..., then the transition probabilities are said to be stationary.
Thus having stationary transition probabilities implies that the transition probabilities do not change over time.

If the transition probabilities in a Markov chain are independent of the time point t, if P(t) = P(t') for all t,t' ≥ 0, then the Markov chain is said to be homogeneous. Otherwise it is heterogeneous.

A homogeneous Markov chain with transition matrix P is called irreducible if for each pair (i,j) of states, the state transition graph contains a path from i to j, which is equivalent to saying that an n ≥ 1 exists with (Pn)ij > 0

A homogeneous Markov chain with transition matrix P is called aperiodic if for each state i the greatest common divisor gcd(Wi) = 1, where Wi is the set containing the lengths of all paths from i to itself in the state transition graph, n ∈ Wi if and only if (Pn)ii > 0

2.6 MATHEMATICAL MODEL OF THE ALGORITHM

The generation matrix G defines for each pair of solutions i, j ∈ S the probability of generating solution j from solution i. An entry Gij is called the generation probability and satisfies Gij > 0 if and only if j ∈ N(i).

$$G_{ij}(c_k) = \begin{cases} \frac{1}{|N(i)|}, & if\ j \in S \\ 0, & if\ j \notin S \end{cases} \qquad (2.1)$$

The acceptance matrix A(ck) defines for each pair of solutions i, j ∈ S the probability of accepting solution j from solution i in the kth iteration of the simulated annealing algorithm. The acceptance probability Ai,j(ck) is given by

$$A_{ij}(c_k) = \begin{cases} 1, & if\ f(j) \leq f(i) \\ \exp\left(\frac{f(i)-f(j)}{c}\right), & if\ f(j) > f(i) \end{cases} \quad (2.2)$$

the transition matrix P(ck) defines for each pair of solutions i, j ∈ S the probability of moving from solution i to solution j in the kth iteration of the simulated annealing algorithm.

The transition probability Pij(ck) is given by

$$P_{ij}(c_k) = \begin{cases} G_{ij}A_{ij}(c_k), & if\ i \neq j \\ 1 - \sum_{l \in S,\ l \neq i} P_{il}(c_k), & if\ i = j \end{cases} \quad (2.3)$$

If the generation matrix satisfies Gij = Gji for all i,j ∈ S, then the finite homogeneous Markov chain associated with a run of simulated annealing at a fixed value c of the control parameter is strongly ergodic and the components of the unique stationary distribution Q(c) to which its probability distribution converges are given by

$$P(X = i) = q_i(c) = \frac{1}{N_0(c)} exp\left(\frac{-f_{(i)}}{c}\right) \quad (2.4)$$

$$N_0(c) = \sum_{j \in S} exp\left(\frac{-f(j)}{c}\right) \quad (2.5)$$

CHAPTER 3

3.1 PROBLEM SETUP

This problem is a minimization problem and it consists of p consignments, with different number of wagons each n1, n2, n3 accordingly, with k number of trains in service. The objective is to find the minimum cost.

Below notation is given:

Indices

$\boldsymbol{p}$ is defined as the index for consignments. In our problem we have three consignments.

$\boldsymbol{a}$ is the index for each wagon. For the first consignment p = 1 let's say we have a demand from a shipper to move 30 wagons. When p = 2 we have a quantity of 18 wagons while when p = 3 we have 21 wagons as can be seen in the table below.

$\boldsymbol{k}$ is the index for trains. Specific to our problem, seven trains are available and each one has a well-defined timetable.

$\boldsymbol{i}, \boldsymbol{j}$ are the indices for stations. We have four stations, numbered from 1 to 4. These stations are usually given into pairs of origin and destination depending on each consignment. For example if p equals 1, origin is station 1 and destination is station 4. if p equals 2, origin is station 1 and destination is station 3. Last but not least, if p equals 3, origin is station 2 and destination is station 4. It must be noted that trains can stop at intermediate stations.

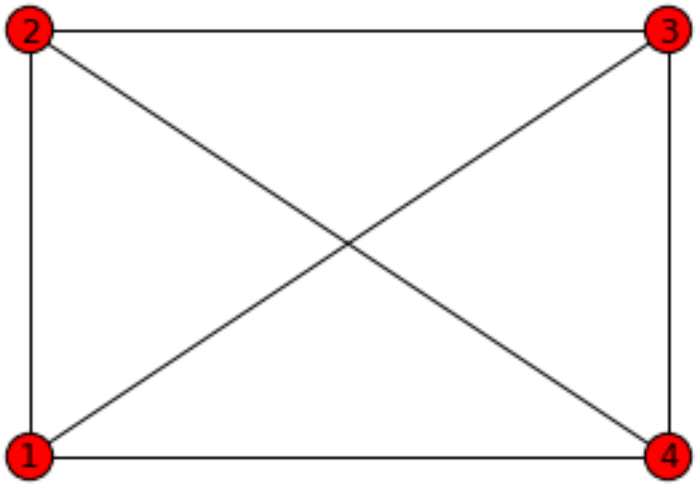

Figure 2: Railroad sample

Variables

$$\delta_{(i,j,a,p,k)} = \begin{cases} 1, & if\ wagon\ a\ of\ consignment\ p\ is\ transported \\ & \quad by\ train\ k\ in\ arc\ (i,j) \\ 0, & otherwise \end{cases}$$

δ is a five dimensional matrix that contains all available data. Origin station, terminal station, specific number of wagon, consignment number and train number.

Parameters

$\theta_{i,j}$ Required time for passing arc (i, j)

$\varphi_{i,j,k}$ Departure time of train k in arc (i, j)

$Ct_{i,j}$ Train formation cost for arc (i, j)

Cw_p Cost of one hour being idle or in service related to each wagon of consignment p

Cc Shunting operation cost for each wagon

$u_{i,j}$ Maximum number of wagons hauled by a locomotive in arc (i, j)

s_p Origin station of consignment p

e_p Destination station of consignment p

Consignment data

	Consignment 1	Consignment 2	Consignment 3
Origin	1	1	2
Destination	4	3	4
Quantity of wagons	30	18	21

The objective is to determine:

The number of required trains to transport consignments from their origin to their destination, the timetable for each train containing the departure and arrival time in each station and the timetable for each wagon containing the departure and arrival time in addition to the idle time of wagons in each station.

The wagon classification cost Cc for each wagon is 3 units. The cost Cw_p for one hour idle or being in service for each wagon is 2 units. For all arcs the parameter $u_{i,j}$ is considered to be equal to 36 wagons. The required time for connecting and disconnecting wagons to/ from trains in stations is assumed to be 1 hour.

Departure time of trains

Train number	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Starting time of trains from station 1 (hour)	1	2	3	4	5	6	7
Starting time of trains from station 2 (hour)	6	8	9	10	11	12	13
Starting time of trains from station 3 (hour)	14	16	17	18	19	20	21

Train formation costs

$$Ct_{i,j}=\begin{bmatrix}0 & 2000 & 5000 & 7000\\ 0 & 0 & 3000 & 5000\\ 0 & 0 & 0 & 2000\end{bmatrix}$$

Required time for traversing an arc (i,j)

$$\theta_{i,j}=\begin{bmatrix}0 & 5 & 13 & 20\\ 0 & 0 & 8 & 15\\ 0 & 0 & 0 & 7\end{bmatrix}$$

COSTS THAT OCCUR SPECIFIC TO THIS PROBLEM

Train formation costs include: train personnel wages, consumed oil, gasoline and the amortization of the locomotive. The cost of using a wagon creates costs, i.e rental/ lease of a wagon. Lastly the cost derived from the shunting station for the classification work of the wagons, this includes separation and connection from the inbound train to the outbound train.

The cost of train formation

The objective of this model consists of three parts: the model minimizes the cost of each train formation. This cost consists of train personnel wages, consumed oil and gasoline and amortization of the locomotive. This cost is equal to

$$\sum_{i}\sum_{j}\sum_{k}\gamma_{(i,j,k)} \times Ct_{i,j}$$

The cost of wagons usage:

The cost of using a wagon creates costs and is proportional to time. This cost is equal to

$$\sum_{i}\sum_{j}\sum_{a}\sum_{p}\sum_{k}\left(\delta_{(i,j,a,p,k)} \times \left(\varphi_{i,k} + \theta_{i,j}\right) \times Cw_{p}\right)$$

The cost of wagon classification in shunting stations

This cost includes wagon separation works from arrived trains and wagon connection to leaving trains. This cost contains the oil and gasoline of shunting locomotives, plus the train personnel wages and shunting operators

$$\sum_{i}\sum_{p}\sum_{a}\lambda_{(i,a,p)} \times Cc$$

If we sum up all the three above parts, we have a single **objective function**. It must be noted that our problem is a minimization problem and our objective is to find the minimum cost. This function is:

Min Z=

$$\sum_i \sum_j \sum_k \gamma_{(i,j,k)} \times Ct_{i,j} \; + \; \sum_i \sum_j \sum_a \sum_p \sum_k \left(\delta_{(i,j,a,p,k)} \times \left(\varphi_{i,k} + \theta_{i,j}\right) \times Cw_p\right) \; + \; \sum_i \sum_p \sum_a \lambda_{(i,a,p)} \times Cc \qquad (3.1)$$

Constraints

In order to ensure that all consignments leave their origins, Eq.2 is applied to the model

$$\sum_j \sum_k \delta_{(s_p,j,a,p,k)} = 1 \quad \forall \quad p, a \quad \& \quad s_p < j < e_p \qquad (3.2)$$

Inequality 3 prevents from assigning wagons more than the maximum capacity of trains. In this problem the maximum capacity that a train can carry is 36 wagons.

$$\sum_p \sum_a \delta_{(i,j,a,p,k)} \; \leq \; u_{i,j} . \gamma_{(i,j,k)} \quad \forall \quad i, j, k \quad \& \quad s_p \leq i < e_p \, , \;\; s_p < j \leq e_p \qquad (3.3)$$

Inequality 4 ensures that all wagons travel from their origins to their destinations successively

$$\sum_k \varphi_{i,j,k} \cdot \delta_{(s_p,j,a,p,k)} \geq \sum_k \sum_h \delta_{(h,i,a,p,k)} \times \left(\varphi_{h,i,k} + \theta_{h,i}\right)$$

$$\forall \quad a, p, i, j \quad \& \; s_p \leq i < e_p \, , \;\; s_p < j \leq e_p \qquad (3.4)$$

Inequality 5 specifies that if a wagon stops in a station, the necessary classification works should be done, and therefore the relevant costs are added to the objective function

$$\lambda_{(j,a,p)} \geq \; \sum_i \sum_k \left(\delta_{(i,j,a,p,k)} - \sum_l \delta_{(j,l,a,p,k)}\right)$$

$$\forall \, a, p, j \quad \& \quad s_p \leq i < e_p \, , \;\; s_p < j \leq e_p \qquad (3.5)$$

3.2 DESCRIPTION OF THE PROCEDURE

Initial temperature is defined, as well as the available number of trains, the number of starting stations, the number of terminal stations, the number of consignments and the maximum number of wagons that a train can haul. By using all these values we create three matrices that we will use later, in order to store our data.

The number of iterations is set to zero. While the temperature is not zero and while the number of iterations are below a specific threshold, the algorithm begins by using simulated annealing procedure as will be explained shortly. Objective (old) is made up of three parts that are summed altogether.

The first part contains the evaluation of matrix $\boldsymbol{\gamma}_{(i,j,k)}$ in terms of costs, the second part contains the evaluation of matrix $\boldsymbol{\lambda}_{(a,p,i)}$ and the third and last part contains the costs derived from matrix $\boldsymbol{\delta}_{(i,j,a,p,k)}$.

Having evaluated the costs of each matrix we can now begin to alter the situation of each state – matrix. This is done again in three separate functions each one for each matrix. Beginning with the change of matrix delta we use a 'flag' as a point of reference in order to cut off solutions that do not satisfy our constraints. 'flag' is updated when each of our constraint is being examined.

While flag>0 meaning that at least one constraint is not satisfied, so we keep producing solutions. Then we initialize matrix $\boldsymbol{\delta}_{(i,j,a,p,k)}$ each time the loop is executed by placing zeros to all variables contained in matrix $\boldsymbol{\delta}_{(i,j,a,p,k)}$.

For each consignment (p), the number of wagons that has to be carried ,the starting station and the terminal station are given. Since the main focus is the cost to be based on each wagon, we continue using a random number generator in order to create the routing for each one of the wagons by giving two options for the routing, either to be

direct (straight to the terminal station) or by stopping at an intermediate station, where consequently changes to each train may occur.

Also a train is chosen randomly in order to carry out the specific routing. Since we have completed the routing we must now examine if constraints are met. Because we had a very large number of trains used, we had to figure out how we could suppress the costs formed, which means we had to reduce their number.

The solution proposed is to build a constraint that first will find all the train movements on each arc (it has to be noted that it is assumed that trains travel from left to right) and secondly if there are trains in common that use the same arc we will try to consolidate not all but at least some of them.

In our second constraint we check the total capacity that is being pulled from each train not only in one arc but also in a sequence of them. Previously where we referred to sequence we implied that if a train starts from station 1 it has all the possible routes available as stated here [(1,2),(1,3),(1,4)] so in each one of these stations, although a train might stop, it is not necessary that all wagons will be unloaded. Let's assume that a train can stop in a station not to deliver but to make a pickup and then continue so as to unload all of the wagons at another station. So we will have to assess each previous route until we arrive at a terminal station or where the route of our train ends. For the intermediate stations the assessment has to be done in arcs [(1,3),(1,4),(2,3),(2,4)] and for the terminal stations [(1,4),(2,4),(3,4)].

If a sum of arcs exceeds the maximum value of 36 that a train can pull then the rest of the wagons have to be added to a new train that will continue the routing which is given from the wagon's consignment data until they reach their final destination.

Last but not least it has to be stated that when we randomly choose a new train to carry on the existed routing, an additional check of the capacity of the new train has to

be executed.

In the third constraint the validity of a routing is checked. For example if a wagon is scheduled to stop at a station with train (k1) and then it must proceed to a next station with another train (k2), then the time of departure of successor train k2 must be **>** (bigger than) the time of arrival of the predecessor train (k1).

In the fourth constraint we have to check if exists the available margin of one hour between trains that exchange wagons in order to ensure that the necessary classification work has been done and the classification yard has the time to set the wagons in the track of departure.

Since all the above constraints hold we have a solution formed inside the neighborhood that we created from the limitations of our constraints. That means that a new matrix **δ**_new is set up at least for now, until the next evaluation begins.

From **δ**_new we can derive **γ**_new and **λ**_new so as to help us with the evaluation of our solution.

The next step now is to evaluate objective_new, which, as mentioned before, it consists of three parts, the evaluation of each new matrix. If we sum up these three partial objectives we have the total of the objective_new.

Next, the most important part of the simulated annealing algorithm is presented. If objective_new <= (smaller than or equal to) objective_old accept the changes, else give a random number (p0) in the interval (0,1) and if that number (p0) < (smaller than) exp(-ΔE/T) where ΔE = (equals the difference) (objective_new – objective_old) then accept the changes.

number of iterations = number of iterations + 1. Reduce the temperature based on a cooling schedule (T=T/log (number of iterations)).

Microstate

Change from s to s’

Microstate change is divided in two phases.

Before all it has to be noted that during the entrance in every repetition to the part where we change microstate and by that we mean the attempt to tweak the routing of a wagon, matrix δ is zeroed. In first phase for every consignment p given that initial station is known and for every wagon a, the algorithm gives a random intermediate or final station j and a random train k in order to formulate an initial solution. In the second phase in an attempt to optimize the running time of the procedure we are locating which trains have their routes in common and we try to consolidate their wagons. When a consolidation occurs we choose randomly the next train to carry on the routing from the already pre-selected trains that existed at the previous step. Secondly as a consequence of the consolidations we ought to check the capacity in every routing from its beginning to its end so as to avoid violation on constraint given by our problem. Moreover we will have to check if a wagon has wrongly been to a next station with a previous train. That would lead to a wrong solution according to programming and would have given wrong timetable. Also we check if there is enough time margin for the wagons classification operations between the time of arrival and the time of departure for each connection. Finally a total check runs through all these changes that occurred to the specific solution to decide if there exists a constraint violation and gives us a flag meaning that only then we can approve that solution.

CHAPTER 4

4.1 COMPUTATIONAL RESULTS

In the above presented numerical problem we have 7 trains, 3 consignments consisting of 69 wagons and a complete graph of four stations. As a next step we will describe a solution as given from our algorithm.

Train k=3 starts from station 1 for station 2 at 3:00 and arrives after 5 hours at 8:00, loaded with cargo, 7 wagons of consignment 1 and 10 wagons of consignment 2. At station 2 unloads 7 wagons of consignment 1, (3,8,11,16,21,25,27) and loads 10 wagons of consignment 3. It then departs from station 2 for station 3 at 9:00. At station 3 unloads 10 wagons of consignment 2 (terminal station for consignment 2) and 10 wagons of consignment 3. (1,3,4,6,7,10,13,17,20,21). The specific train is put out of service.

train k=5 starts from station 1 at 5:00 with destination station 3 where it arrives after 13 hours at 18:00 loaded with 23 wagons of consignment 1 and 8 wagons of consignment 2.

at station 3 unloads 8 wagons of consignment 2 and loads 10 wagons of consignment 3 (1,3,4,6,7,10,13,17,20,21) which train k=3 had delivered. It departs from station 3 for station 4 at 19:00.

train k=7 starts from station 2 at 13:00 with destination station 4 where arrives after 15 hours at 4:00. at station 2 loads 7 wagons of consignment 1, (3,8,11,16,21,25,27) and 11 wagons of consignment 3 (which has as start station 2).

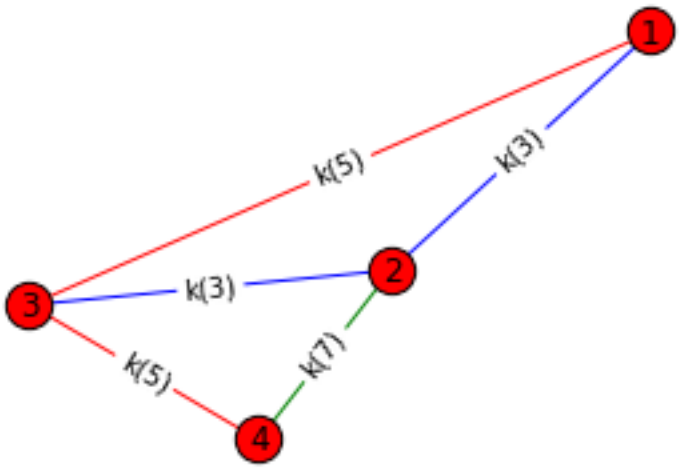

Figure 4

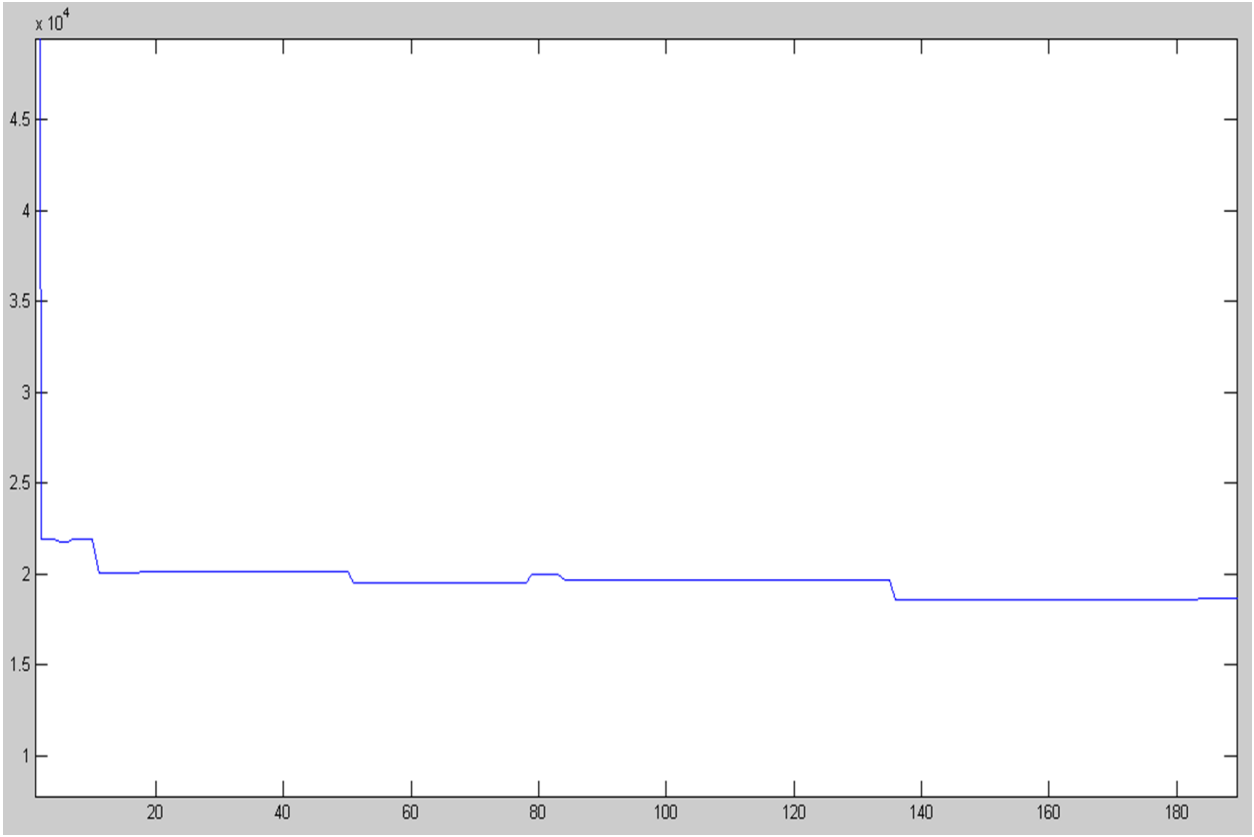

Figure 5

Solution with total cost: 19883

the trains formed:

γ(1,2,3) = γ(2,3,3) = γ(1,3,5) = γ(3,4,5) = γ(2,4,7) = 1

the wagons should stop in stations:

λ(3,1,2) = λ(8,1,2) = λ(11,1,2) = λ(16,1,2) = λ(21,1,2) = 1

λ(25,1,2) = λ(27,1,2) = λ(1,3,3) = λ(3,3,3) = λ(4,3,3) = 1

$\lambda(6,3,3) = \lambda(7,3,3) = \lambda(10,3,3) = \lambda(13,3,3) = \lambda(17,3,3) = 1$

$\lambda(20,3,3) = \lambda(21,3,3) = 1$

the assigned trains and arcs to wagons:

$\delta(1,2,3,1,3) = \delta(1,2,8,1,3) = \delta(1,2,11,1,3) = \delta(1,2,16,1,3) = \delta(1,2,21,1,3) = 1$

$\delta(1,2,25,1,3) = \delta(1,2,27,1,3) = \delta(1,2,2,2,3) = \delta(2,3,2,2,3) = \delta(1,2,3,2,3) = 1$

$\delta(2,3,3,2,3) = \delta(1,2,4,2,3) = \delta(2,3,4,2,3) = \delta(1,2,8,2,3) = \delta(2,3,8,2,3) = 1$

$\delta(1,2,9,2,3) = \delta(2,3,9,2,3) = \delta(1,2,14,2,3) = \delta(2,3,14,2,3) = \delta(1,2,15,2,3) = 1$

$\delta(2,3,15,2,3) = \delta(1,2,16,2,3) = \delta(2,3,16,2,3) = \delta(1,2,17,2,3) = \delta(2,3,17,2,3) = 1$

$\delta(1,2,18,2,3) = \delta(2,3,18,2,3) = \delta(2,3,1,3,3) = \delta(2,3,3,3,3) = \delta(2,3,4,3,3) = 1$

$\delta(2,3,6,3,3) = \delta(2,3,7,3,3) = \delta(2,3,10,3,3) = \delta(2,3,13,3,3) = \delta(2,3,17,3,3) = 1$

$\delta(2,3,20,3,3) = \delta(2,3,21,3,3) = \delta(1,4,1,1,5) = \delta(1,4,2,1,5) = \delta(1,3,4,1,5) = 1$

$\delta(3,4,4,1,5) = \delta(1,4,5,1,5) = \delta(1,4,6,1,5) = \delta(1,4,7,1,5) = \delta(1,4,9,1,5) = 1$

$\delta(1,3,10,1,5) = \delta(3,4,10,1,5) = \delta(1,4,12,1,5) = \delta(1,4,13,1,5) = \delta(1,4,14,1,5) = 1$

$\delta(1,4,15,1,5) = \delta(1,3,17,1,5) = \delta(3,4,17,1,5) = \delta(1,4,18,1,5) = \delta(1,3,19,1,5) = 1$

$\delta(3,4,19,1,5) = \delta(1,4,20,1,5) = \delta(1,4,22,1,5) = \delta(1,3,23,1,5) = \delta(3,4,23,1,5) = 1$

$\delta(1,3,24,1,5) = \delta(3,4,24,1,5) = \delta(1,4,26,1,5) = \delta(1,3,28,1,5) = \delta(3,4,28,1,5) = 1$

$\delta(1,3,29,1,5) = \delta(3,4,29,1,5) = \delta(1,3,30,1,5) = \delta(3,4,30,1,5) = \delta(1,3,1,2,5) = 1$

$\delta(1,3,5,2,5) = \delta(1,3,6,2,5) = \delta(1,3,7,2,5) = \delta(1,3,10,2,5) = \delta(1,3,11,2,5) = 1$

$\delta(1,3,12,2,5) = \delta(1,3,13,2,5) = \delta(3,4,1,3,5) = \delta(3,4,3,3,5) = \delta(3,4,4,3,5) = 1$

$\delta(3,4,6,3,5) = \delta(3,4,7,3,5) = \delta(3,4,10,3,5) = \delta(3,4,13,3,5) = \delta(3,4,17,3,5) = 1$

$\delta(3,4,20,3,5) = \delta(3,4,21,3,5) = \delta(2,4,3,1,7) = \delta(2,4,8,1,7) = \delta(2,4,11,1,7) = 1$

$\delta(2,4,16,1,7) = \delta(2,4,21,1,7) = \delta(2,4,25,1,7) = \delta(2,4,27,1,7) = \delta(2,4,2,3,7) = 1$

$\delta(2,4,5,3,7) = \delta(2,4,8,3,7) = \delta(2,4,9,3,7) = \delta(2,4,11,3,7) = \delta(2,4,12,3,7) = 1$

$\delta(2,4,14,3,7) = \delta(2,4,15,3,7) = \delta(2,4,16,3,7) = \delta(2,4,18,3,7) = \delta(2,4,19,3,7) = 1$

A MATHEMATICAL MODEL FOR TRAIN ROUTING AND SCHEDULING PROBLEM WITH FUZZY APPROACH.

SOLUTION:

In the paper presented by Jamili (2012) the batch of wagons are presented specifically in triplets which do not split during their whole routing from the start until the terminal station.

The final optimum value for objective function: 18615

The trains formed are :

$\gamma(1,2,4) = \gamma(1,3,5) = \gamma(2,3,6) = \gamma(3,4,5) = \gamma(3,4,7) = 1$

The wagons should stop in stations:

$\lambda(1,2,2) = \lambda(2,2,2) = \lambda(3,2,2) = \lambda(4,2,2) = \lambda(5,2,2) = 1$

$\lambda(6,2,2) = \lambda(1,3,3) = \lambda(2,3,3) = \lambda(3,3,3) = \lambda(4,3,3) = 1$

$\lambda(5,3,3) = \lambda(6,3,3) = \lambda(7,3,3) = 1$

The assigned trains and arcs to wagons:

$\delta(1,3,1,1,5) = \delta(1,3,2,1,5) = \delta(1,3,3,1,5) = \delta(1,3,4,1,5) = \delta(1,3,5,1,5) = 1$

$\delta(1,3,6,1,5) = \delta(1,3,7,1,5) = \delta(1,3,8,1,5) = \delta(1,3,9,1,5) = \delta(1,3,10,1,5) = 1$

$\delta(1,2,3,2,4) = \delta(1,2,4,2,4) = \delta(1,2,5,2,4) = \delta(1,2,6,2,4) = \delta(1,3,1,2,5) = 1$

$\delta(1,3,2,2,5) = \delta(2,3,3,2,6) = \delta(2,3,4,2,6) = \delta(2,3,5,2,6) = \delta(2,3,6,2,6) = 1$

$\delta(2,3,1,3,6) = \delta(2,3,2,3,6) = \delta(2,3,3,3,6) = \delta(2,3,4,3,6) = \delta(2,3,5,3,6) = 1$

$\delta(2,3,6,3,6) = \delta(2,3,7,3,6) = \delta(3,4,1,1,5) = \delta(3,4,2,1,5) = \delta(3,4,3,1,5) = 1$

$\delta(3,4,4,1,5) = \delta(3,4,5,1,5) = \delta(3,4,6,1,5) = \delta(3,4,7,1,5) = \delta(3,4,8,1,5) = 1$

$\delta(3,4,9,1,5) = \delta(3,4,10,1,5) = \delta(3,4,1,3,7) = \delta(3,4,2,3,7) = \delta(3,4,3,3,7) = 1$

$\delta(3,4,4,3,7) = \delta(3,4,5,3,7) = \delta(3,4,6,3,7) = \delta(3,4,7,3,7) = 1$

REFERENCES

Cordeau J., Toth P., and Vigo D., 1998 " a survey of optimization models for train routing and scheduling"

Branch and price for a European variant of the railroad blocking problem, Robert Voll and Uwe Clausen

A mathematical model for train routing and scheduling problem with fuzzy approach, Amin Jamili

Simulated annealing: theory and applications, Laarhoven and Aarts, 1987

Theoretical aspects of local search, Wil Michiels, Emile Aarts, Jan Korst, 2007

Introduction to operations research, Hillier / Lieberman,2001

http://ec.europa.eu/eurostat/

APPENDIX MATLAB CODE

```
clear
clc
tic
T=500;
%N1 number of trains
%N2 number of stations
%N3 number of destinations
%N4 number of consignment
%N5 maximum allowed number of wagons
N1=7;
N2=3;
N3=4;
N4=3;
N5=36;
kiter=1;
kiterinner=0;
[gamma delta lamda]=arxikes_times(N1,N2,N3,N4,N5) ;
kiter2=0;
while T>10
while kiter<100
obj10=objGam1(gamma);
obj30=partobj3(lamda);
obj20=objdelta2(delta);
obj0=obj10+obj20+obj30;
%%%%%%%%%%%%%%%%%%
delta1=allaghdelta2(delta);
gamma1=elgxosgamma(delta1);
lamda1=elgxoslambda(delta1);
%%%%%%%%%%%%%%%%%%
obj11=objGam1(gamma1);
obj31=partobj3(lamda1);
obj21=objdelta2(delta1);
obj1=obj11+obj21+obj31;
if obj1<=obj0
   gamma=gamma1;
```

```
   delta=delta1;
    lamda=lamda1;
else
   p0=rand(1);
   if p0<exp(-(obj1-obj0)/T);
      gamma=gamma1;
      delta=delta1;
       lamda=lamda1;
   end
end
   kiter=kiter+1;
    kiter2=kiter2+1;
   y(kiter2)=obj1;
   z(kiter2)=obj0;
 [obj0,obj1,kiter2]
end
kiter=0;
  T=T/log(kiter2);
%    T=0.7*T;
end
toc
figure(1)
plot(z)
gamma
```

function delta=allaghdelta2(~)

```
shm=1;
while shm>0
% initialize turn delta matrix into zeros
delta=zeros(3,4,36,3,7);
for p=1:3
   if p==1
      NN=30;
      init=1;
       term=4;
   elseif p==2
```

```
       NN=18;
       init=1;
       term=3;
   elseif p==3
       NN=21;
        init=2;
        term=4;
   end
     for a=1:NN
 % creation of routing
j=ceil(rand*term);
k=ceil(rand*7);
while j<=init
     j=ceil(rand*term);
end
if j==term
   delta(init,j,a,p,k)=1;
else
delta(init,j,a,p,k)=1;
   % give a next train
   k2=ceil(rand*7);
   while k2<k
     k2=ceil(rand*7);
   end
   delta(j,term,a,p,k2)=1;
end
     end
end
% % % % % % % % % % % % % % %%
% reduce and converge number of trains used
delta=siglisi(delta);
% % % % % % % % % % % % % % %
% check train's total capacity
delta=capacity(delta);
% % % % % % % % % % % % % % %
% check if a wagon has been at the next station with a previous train
delta=pred(delta);
% check for late arrival
```

```
delta=arrival(delta);
% % % % % % % % % % % % % % % % % % % % %
 [delta,shm]=check2(delta);
% shm
end
```

function delta=siglisi(delta)

```
 s5=diadromes(delta);
for i=1:3
 for j=i+1:4
x1= (s5(:,1)==i & s5(:,2)==j);
y1=s5(x1,:);
length(y1(:,1));
if  ~isempty(y1(:,1))
% choose from the existing trains
y2=unique(y1(:,5));
y3=length(y2);
y4=randi(y3);
knew=y2(y4);
x2= (s5(:,1)==i & s5(:,2)==j & s5(:,5)==knew);
x3=s5(x2,:);
x4=setdiff(y1,x3,'rows');
    for v1=1:length(x4(:,1))
delta(x4(v1,1),x4(v1,2),x4(v1,3),x4(v1,4),x4(v1,5))=0;
    end
x4(:,5)=knew;
for v2=1:length(x4(:,1)
delta(x4(v2,1),x4(v2,2),x4(v2,3),x4(v2,4),x4(v2,5))=1;
end
end
end
end
 end
```

function delta=capacity(delta)

```
s5=diadromes(delta);
for k=1:7
%    (1,2)+(1,3)+(1,4)
a=s5(s5(:,1)==1 & s5(:,5)==k,:);
%  (1,3)+(1,4)+(2,3)+(2,4)
b=s5(s5(:,1)==1 & s5(:,2)==3 & s5(:,5)==k,:);
c=s5(s5(:,1)==1 & s5(:,2)==4 & s5(:,5)==k,:);
d=s5(s5(:,1)==2 & s5(:,2)==3 & s5(:,5)==k,:);
e=s5(s5(:,1)==2 & s5(:,2)==4 & s5(:,5)==k,:);
b2=[b;c;d;e];
% (1,4)+(2,4)+(3,4)
f=s5(s5(:,2)==4 & s5(:,5)==k,:);
if length(a)>36
   a2=a(37:end,:);
   for v3=1:length(a2(:,1))
delta(a2(v3,1),a2(v3,2),a2(v3,3),a2(v3,4),a2(v3,5))=0;
   end
kover=randi(7);
a2(:,5)=kover;
for v4=1:length(a2(:,1))
delta(a2(v4,1),a2(v4,2),a2(v4,3),a2(v4,4),a2(v4,5))=1;
end
end
if length(b2)>36
   b6=[];
   while length(b2)>36
   const=length(b2)-36;
   b4=randi(length(b2),const,1);
%    pause
   b3=unique(b4);
   length(b3);
   b5=b2(b3,:);
   length(b5);
   b6=[b6;b5]; % change  specific delta into another train
   length(b6);
   b2=setdiff(b2,b5,'rows'); % continue the routing
   length(b2);
%    pause
```

```
   end
  for v3=1:length(b6(:,1))
delta(b6(v3,1),b6(v3,2),b6(v3,3),b6(v3,4),b6(v3,5))=0;
  end
%  gives rand k
%      check the capacity of the available train
kover=randi(7);
b6(:,5)=kover;
for v4=1:length(b6(:,1))
delta(b6(v4,1),b6(v4,2),b6(v4,3),b6(v4,4),b6(v4,5))=1;
end
end
if length(f)>36
   f1=f(37:end,:);
   for v3=1:length(f1(:,1))
   delta(f1(v3,1),f1(v3,2),f1(v3,3),f1(v3,4),f1(v3,5))=0;
   end
kover=randi(7);
f1(:,5)=kover;
for v4=1:length(f1(:,1))
delta(f1(v4,1),f1(v4,2),f1(v4,3),f1(v4,4),f1(v4,5))=1;
end
end
end
```

function delta=pred(delta)

```
s5=diadromes(delta);
for p=1:3
   for a=1:36
a3=s5(s5(:,3)==a & s5(:,4)==p,:);
% check if a wagon has been at the next station with a previous train
if length(a3(:,1))>1 && a3(1,2)>a3(2,2) && a3(1,5)<a3(2,5)
   k2=a3(2,5);
   k1=randi(7);
   while k1<k2
     k1=randi(7);
```

```
   end
delta(a3(1,1),a3(1,2),a3(1,3),a3(1,4),a3(1,5))=0;
a3(1,5)=k1;
delta(a3(1,1),a3(1,2),a3(1,3),a3(1,4),a3(1,5))=1;
end
   end
end
```

function delta=arrival(delta)

```
s5=diadromes(delta);
gamma=elgxosgamma(delta);
phi=[1:7;6,8:13;14,16:21];
theta=[0,5,13,20;0,0,8,15;0,0,0,7];
s=[];
for k=1:7
   for i=1:3
      for j=i+1:4
         if gamma(i,j,k)>0
            s=[s;i,j,k];
         end
      end
   end
   if ~isempty(s)
      a=s(s(:,3)==k,:);
      for c=1:length(a(:,1))-1
         arr=phi(a(c,1),k)+theta(a(c,1),a(c,2));
         ip=a(c,1);
         jp=a(c,2);
         c=c+1;
         dep=phi(a(c,1),k);
         % the time of arrival must not occur with the time of departure
         % in order to be sufficient time for wagons classification
         if arr>=dep
         [ip,jp,k];
           pin=s5(s5(:,1)==ip & s5(:,2)==jp & s5(:,5)==k,:);
           for c1=1:length(pin(:,1))
```

```
          delta(pin(c1,1),pin(c1,2),pin(c1,3),pin(c1,4),pin(c1,5))=0;
           end
           kf=randi(7);
           pin(:,5)=kf;
           for c1=1:length(pin(:,1))
          delta(pin(c1,1),pin(c1,2),pin(c1,3),pin(c1,4),pin(c1,5))=1;
           end
         end
      end
   end
end
```

function s5=diadromes(delta)

```
it5=0;
for k=1:7
   for p=1:3
      for a=1:36
         for j=1:4
            for i=1:3
               if delta(i,j,a,p,k)>0
                  it5=it5+1;
                   s5(it5,:)=[i,j,a,p,k];
               end
            end
         end
      end
   end
end
s5;
```

function [delta,shm]=check2(delta)

```
gamma=elgxosgamma(delta);
s5=diadromes(delta);
shm=0;
```

```
for k=1:7
%     (1,2)+(1,3)+(1,4)
if sum(sum(sum(delta(1,:,:,:,k)==1)))>36
%     a=s5(s5(:,1)==1 & s5(:,5)==k,:);
%     k;
% sum(sum(sum(delta(1,:,:,:,k)==1)));
shm=shm+1;
end
%  (1,3)+(1,4)+(2,3)+(2,4)
if sum(sum(sum(sum(delta(1:2,3:4,:,:,k)))))>36
% b=s5(s5(:,1)==1 & s5(:,2)==3 & s5(:,5)==k,:);
% c=s5(s5(:,1)==1 & s5(:,2)==4 & s5(:,5)==k,:);
% d=s5(s5(:,1)==2 & s5(:,2)==3 & s5(:,5)==k,:);
% e=s5(s5(:,1)==2 & s5(:,2)==4 & s5(:,5)==k,:);
% b2=[b;c;d;e];
%     k;
% sum(sum(sum(sum(delta(1:2,3:4,:,:,k)))));
shm=shm+1;
end
% (1,4)+(2,4)+(3,4)
if sum(sum(sum(delta(:,4,:,:,k)==1)))>36
%     f=s5(s5(:,2)==4 & s5(:,5)==k,:);
%     k;
%
% sum(sum(sum(delta(:,4,:,:,k)==1)));
shm=shm+1;
end
end
% % % % % % % % % % % % % % % % % % % %
phi=[1:7;6,8:13;14,16:21];
 theta=[0,5,13,20;0,0,8,15;0,0,0,7];

s=[];
for k=1:7
   for i=1:3
      for j=i+1:4
         if gamma(i,j,k)>0
            s=[s;i,j,k];
```

```
            end
        end
    end
    if ~isempty(s)
        a=s(s(:,3)==k,:);
        for c=1:length(a(:,1))-1
            arr=phi(a(c,1),k)+theta(a(c,1),a(c,2));
%           ip=a(c,1);
%           jp=a(c,2);
            c=c+1;
            dep=phi(a(c,1),k);
            % the time of arrival must not occur with the time of departure
            % in order to be sufficient time for wagons classification
            if arr>=dep
%           [ip,jp,k];
%               pin=s5(s5(:,1)==ip & s5(:,2)==jp & s5(:,5)==k,:);
              shm=shm+1;
            end
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%
for p=1:3
    for a=1:36
a3=s5(s5(:,3)==a & s5(:,4)==p,:);
% check if a wagon has been at the next station with a previous train
if length(a3(:,1))>1 && a3(1,2)>a3(2,2) && a3(1,5)<a3(2,5)
%   a3;
shm=shm+1;
end
    end
end
% shm;
```

function [gamma delta lamda]=arxikes_times(N1,N2,N3,N4,N5)

```
% gamma=round(rand(N2,N3,N1));
```

```
gamma=ones(N2,N3,N1);
for k=1:N1
   for i=1:N2
      for j=1:N3
         if i>=j
            gamma(i,j,k)=0;
         end
      end
   end
end
lamda=zeros(N5,N4,N2);% (a,p,i)
delta=zeros(N2,N3,N5,N4,N1);
```

function gamma=elgxosgamma(delta)

```
s=zeros(3,4,7);
gamma=zeros(3,4,7);
for i=1:3 % initial station
   for j=i+1:4 % intermediate / terminal
      for k=1:7 % train
            s1=sum(sum(delta(i,j,:,:,k)));
               s(i,j,k)=s1;
      end
   end
end
%       s;
   for k=1:7
         [row,col]=find(s(:,:,k)>0);
         table=[row,col];
         tab2=unique(table);
  if ~isempty(tab2)
x=1;
toksa=[];
   for n=1:length(tab2)-1
   k;
   toksa(x,:)=[tab2(n);tab2(n+1)];
   x=x+1;
```

```
%    pause
   end
for n2=1:length(toksa(:,1))
   gamma(toksa(n2,1),toksa(n2,2),k)=1;
   end
  end
   end
```

function lamda=elgxoslambda(delta)

```
s=zeros(36,3,3);
lamda=zeros(36,3,3);
for i=1:3
   for p=1:3
      if p==1 || p==2
         init=1;
      else
      init=2;
      end
      for a=1:36
            s1=sum(sum(delta(i,:,a,p,:)));
               s(a,p,i)=s1;
               if s(a,p,i)>0 && i~=init
            lamda(a,p,i)=1;
               end
      end
   end
end
```

function y=objdelta2(delta)

```
phi=[1:7;6,8:13;14,16:21];
theta=[0,5,13,20;0,0,8,15;0,0,0,7];
cw=2;
s=0;
for i=1:3
```

```
  for j=i+1:4;
    for p=1:3
if p==1
NN=30;
elseif p==2
NN=18;
else
NN=21;
end
       for a=1:NN
          for k=1:7
s=s+delta(i,j,a,p,k)*(theta(i,j)+phi(i,k))*cw;
          end
        end
    end
  end
end
y=s;
```

function y=objGam1(gamma)

```
N1=7;%number of train
Ct=[0 2000 5000 7000;0 0 3000 5000;0 0 0 2000;0 0 0 0];
N2=3;
N3=4;
s=0;
for k=1:N1
   for i=1:N2
      for j=i+1:N3
         s=s+gamma(i,j,k)*Ct(i,j);
      end
   end
end
y=s;
```

```
function obj3=partobj3(lamda)
cc=3;
s=0;
 for i=1:3
   for p=1:3
      for a=1:36
   s=s+lamda(a,p,i)*cc;
      end
   end
 end
obj3=s;



%%%%%%%%%%%%%%%%%%%
clc
fort=[];
%  find which wagons transferred to another train
for c=1:length(s5(:,1))
    a=s5(s5(:,3)==s5(c,3) & s5(:,4)==s5(c,4) & s5(:,5)~=s5(c,5) ,:);
    if ~isempty(a)
       fort=[fort;a];
    end
end
fort
```