This is the author's version of a work that was submitted/accepted for publication in the following source:

Hanafi, Rosmalina & Kozan, Erhan (2014)
A hybrid constructive heuristic and simulated annealing for railway crew scheduling.
Computers & Industrial Engineering, 70, pp. 11-19.

This file was downloaded from: http://eprints.qut.edu.au/66685/

© Copyright 2014 Elsevier

This is the author's version of a work that was accepted for publication in Computers & Industrial Engineering. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Computers & Industrial Engineering, [VOL 70, (2014)] DOI: 10.1016/j.cie.2014.01.002

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:

http://doi.org/10.1016/j.cie.2014.01.002
A hybrid constructive heuristic and simulated annealing
for railway crew scheduling

Rosmalina Hanafi - Erhan Kozan *

Decision Science Discipline, Mathematical Sciences School, Science and Engineering Faculty, Queensland University of Technology, 2 George Street, GPO Box 2434, Brisbane Qld 4001, Australia

*Corresponding author. Tel: +61 7 3138 1029 Fax: +61 7 3138 2310 E-mail address: e.kozan@qut.edu.au

Abstract

Railway crew scheduling problem is the process of allocating train services to the crew duties based on the published train timetable while satisfying operational and contractual requirements. The problem is restricted by many constraints and it belongs to the class of NP-hard. In this paper, we develop a mathematical model for railway crew scheduling with the aim of minimising the number of crew duties by reducing idle transition times. Duties are generated by arranging scheduled trips over a set of duties and sequentially ordering the set of trips within each of duties. The optimisation model includes the time period of relief opportunities within which a train crew can be relieved at any relief point. Existing models and algorithms usually only consider relieving a crew at the beginning of the interval of relief opportunities which may be impractical. This model involves a large number of decision variables and constraints, and therefore a hybrid constructive heuristic with the simulated annealing search algorithm is applied to yield an optimal or near-optimal schedule. The performance of the proposed algorithms is evaluated by applying computational experiments on randomly generated test instances. The results show that the proposed approaches obtain near-optimal solutions in a reasonable computational time for large-sized problems.

Keywords: railway crew scheduling, mathematical programming, constructive heuristics, simulated annealing, combinatorial optimisation.

1. Introduction

The crew scheduling problem (CSP) in the transportation industry represents a computationally difficult combinatorial optimisation problem. The large number of tasks (trips) to include and the complicated operational and contractual requirements are the main reason for the complexity of the problem. Nevertheless, the CSP has been one of the most important focuses of the transportation industry because it affects the company’s profitability and its service quality. An optimal crew schedule is essential to ensure efficient and reliable operations of transportation services. Furthermore, the cyclic nature of the crew scheduling application makes the CSP a good candidate for
optimisation. A small improvement to the crew schedules can lead to accumulated savings to produce large annual cost savings. The difficulty of solving CSP yet its enormous practical significance, have led to a large number of proposed solution techniques. However, unlike the airline CSP which has been intensively studied, the railway crew scheduling is less cited in literature (Goumopoulos and Housos, 2004). Railway crew scheduling is domain specific and there has been no developed solving method has yet to be applied universally. Models and algorithms are designed mainly for a specific case and may not readily be applied in different applications.

Railway CSP aims at finding a minimum cost crew schedule. The schedule should cover scheduled trips in a train timetable while subjected to various constraints. Morgado and Martins (1992) presented early work on the crew scheduling application for the Portuguese railways. The system provides a possibility of generating alternative schedules using different scheduling criteria and enables evaluation of the cost of the solution considering a set of produced statistics. CSP is more frequently formulated mathematically, as either set covering problem or set partitioning problem, and then solved by exact solution approaches and heuristics (see for example in Caprara et al. (1997), Kroon and Fischetti (2001), and Freling et al. (2004)). In both the set covering and the set partitioning formulations, the decision variable is a binary integer variable that represents whether or not a duty (roundtrip, pairing) is selected as work for a crew member. The constraint in the set covering problem consists of a matrix of binary values, which defines that each piece of work is covered by a duty at least once. Each column represents one possible pairing or work to be performed by an individual crew member over a defined period of time. The set partitioning problem is similar to the set covering problem, except that in the set partitioning formulation the constraint becomes equal to one, meaning that each task is covered exactly once. Alfieri et al. (2007) proposed a set covering problem based on an implicit column generation solution approach for scheduling train drivers on a railway sub-network. Feasible duties are constructed from a set of trips to be serviced by a number of train drivers, with the aim of minimising the number of duties and maximising the robustness of the schedule. A heuristic procedure is applied to obtain an initial feasible solution together with a heuristic branch-and-price algorithm based on a dynamic programming algorithm for the pricing-out of columns. The main difficulty in applying the exact methods to CSP is that in determining all possible solutions. For the CSP with a large number of trips, there can be an unmanageably large number of possible roundtrips. As a consequence, the problem becomes a time-consuming process of enumerating all the possible roundtrips. Bangert (2012) noted that the method of enumeration is not realistic when the number of options is too large and cannot be practically listed.

Generally, CSP involves a large number of decision variables and it is restricted by many constraints. Fischetti et al. (1987, 1989) have shown that the bus crew scheduling belongs to the class of NP-hard problems. For this reason, there is a requirement of large-scale solution techniques such as column generation. The concept of column generation is to solve a sequence of reduced problems (master problem) in which each reduced problem contains a small fraction of the set of variables (columns). The sub-
problem or an auxiliary problem is commonly formulated as a restricted shortest path problem. The restricted shortest path problem however, is difficult to solve and it also needs other optimisation schemes such as dynamic programming algorithms or branch-and-bound methods. Bengtsson et al. (2007) formulated a general crew pairing problem with the objective function being to minimise the cost of selected pairing and the cost of violating soft constraints. The research combines resource constraints, k-shortest path enumeration, and label merging techniques and shows that a column generation approach is able to heuristically solve large and highly complex railway pairing problems in a reasonable time. Given the size and complexity of the railway operation, the researchers indicate the necessity of combined optimisation techniques. Nishi et al. (2011) proposed a column generation with dual inequality for railway crew scheduling. Computational results have shown that the proposed technique can accelerate the convergence of conventional column generation for a large data set application. Yan and Tu (2002), however, stated that column generation-based methods could be inefficient because when the crew scheduling is formulated as a traditional set covering problem, the obtained optimal solutions could be non-integer solutions. Other techniques should then be incorporated to refine the non-integer solutions.

De Leone et al. (2011) proposed a mathematical model to solve a CSP. Since their proposed model can only handle small- to medium-sized problems, a greedy randomised adaptive search procedure has then been offered to solve large instances. Network flow approach has also been used in several researches on CSP. An attempt towards this approach was proposed by Vaidyanathan et al. (2007). They describe a network flow-based approach to solve a CSP arising in North American railroads. The CSP is formulated as an integer program on a space-time network enforcing the first-in-first-out requirement by including side constraints with the objective of minimising several components of crew expenses. Due to the difficulty of applying the network flow approach to highly complex constraints, this method may be suitable only for small- to moderately-sized real-world problems.

Metaheuristics have become a popular approach in tackling the complexity of practical optimisation problems. Although metaheuristics cannot guarantee optimality of their solutions, they have shown a very good performance in solving real-world optimisation problems. Metaheuristics represent a general type of solution method that illustrates the interaction between local improvement procedures and higher level strategies to facilitate the algorithm for both escaping local optima and exhaustively searching a feasible region. Elizondo et al. (2010) proposed a constructive hybrid approach to address operation management problems that emerge in underground passenger transport. The results are compared with two alternative methods based on tabu search and a greedy heuristic. The tabu search technique provides better results with regard to idle time than both the hybrid and the greedy methods. Dias et al. (2002) proposed a genetic algorithm for bus driver scheduling, which is developed by using a new coding scheme and considering a complex objective function.

Simulated annealing (SA)-based algorithms have been noticed to produce good quality solutions to several combinatorial optimisation problems. Emden and Proksch
(1999) solved an airline CSP using a SA approach. The results show that the SA yields good quality solutions but requires longer processing times than simpler heuristics. Lucic and Teodorovic (1999) applied a SA approach to solve a multi-objective crew scheduling for an airline. In spite of the potential application of SA algorithms to solve combinatorial optimisation problems, there has been few crew scheduling related applications in the literature using the SA algorithm.

This paper presents a new mathematical model and a hybrid constructive SA (HCSA) algorithm to solve railway CSP. The mathematical programming model incorporates commonly encountered real-life railway crew scheduling constraints, particularly the integration of the interval of relief opportunities. To the best of our knowledge, the inclusion of the interval of relief opportunities into models and algorithms has not been studied in depth. The rest of this paper is organised as follows. In Section 2, a brief description of the problem is presented. In Section 3, solution techniques that include formulation of the mathematical programming model and the details of the proposed HCSA algorithm are given. Results of the computational experiments on each approach are provided in Section 4. Finally, Section 5 provides the conclusion and recommendations for further study.

2. Problem description

The railway crew scheduling we are dealing with consists of a set of crew home depots (HDs), a set of relief points (RPs), a set of scheduled train trips with fixed starting and ending times at each location. The problem is to construct crew duties based on the train timetable while satisfying operational and contractual requirements. The crew in this context is the train crew which consists of a train driver and a conductor, and they are considered as a team.

The rail network involves interconnected segments of train tracks where trains travel along specified train lines from one station to a subsequent station. Each segment of train journeys consists of a sequence of trips that must be serviced. Fig. 1 illustrates an example of a train timetable. The route of trains can be traced by straightening the travelling path of trains in the train timetable. Each trip in the timetable must be serviced by a train. The railway CSP is to specify the sequence of trips to be performed by the crew. A train journey begins and ends at a crew HD, and can be feasibly serviced by a single crew.
A train service is the overall journey accomplished by a vehicle from the time it begins at its first station until it arrives at its last station. A vehicle block specifies the sequence of trips made by a train during a service workday. It contains pieces of segments in which crew relief may be performed at both ends of each segment. Each crew belongs to one crew base (HD) and the crew has to start (sign-on) and end (sign-off) his/her duty (daily work shift) at the same crew depot (HD). The spread time is the time elapsed between the crew sign on and the crew sign off in a duty. A train crew duty contains a meal break (MB) which begins after the completion of the third hour and finishes before the completion of the sixth hour, relative to the start of the duty. For example, crews sign on at 08:00 and sign off at 16:00, then the earliest MB will be at 11:00–11:30 and the latest MB will be at 13:29–13:59. The time interval between the earliest break and the latest break corresponds to the transition period between two consecutive pieces of duty, and is defined as a relief opportunities period (ROP). The ROP is a period of time within which a train crew is allowed to be relieved. Any RP can be chosen for crew relief within the two limits of the ROP. The set of crew HDs is a subset of the set of RPs. This transition period includes the time spent for taking a meal and other crew relieving related activities such as handling over a train to (from) another train crew. An example of vehicle blocks and a crew duty with ROP is shown in Fig. 2.

![Figure 1. An example of a train timetable](image-url)
The path of a train is indicated by the blue lines and the purple lines, as shown in the diagram of Fig. 2. The blue lines show the movement of a train from the Station 1 (Depot A) to the terminal at Station n, with transition times (short dwell times) at each station. Crew arriving at stop n can be relieved at this point and take a MB at an away depot (Depot B). The relieved crew may then continue with another vehicle block passing through the same terminal station or RP. Alternatively, the crew may return directly along the route in the opposite direction (the purple lines) and take a MB at the home depot (Depot A). When more trips are considered, the network becomes denser and more paths need to be evaluated. A duty covers a set of consecutive trip segments in a block. The 1st part of a duty (duty stretch) is the period from the start of a duty to the start of the MB, whereas the 2nd part of a duty is the period from the end of the MB to the end of the duty. Transition time (idle interval) between two consecutive trips in each partial duty is the time incurred between the departure time of the next trip and the arrival time of the previous trip.

The railway transportation industry imposes a complex set of operational and contractual requirements corresponding to the work regulations for the crew. For safety reasons, for example, there is a restriction on the length of continuous driving time. A crew will be required to take a break when the total continuous driving time on the same vehicle has reached a maximum limit. In the formation of duties, crew schedule should satisfy several constraints corresponding to work load regulations. There are predetermined maximum and minimum durations of a duty. A minimum of 0.5 h for a MB is required in a duty (shift). A crew takes a break only at a RP and the changeover of trains is at the same RP.
3. Solution approaches

Two solution approaches are proposed for the problem. The first is the exact method of a new mathematical programming model. The mathematical model is formulated based on the information provided by Queensland Rail (QR), Australia. The inclusion of the ROP in this model offers flexibility, because it allows a train crew to be relieved at any RP within the interval of ROP. Existing models usually only consider relieving crew at the beginning of the interval of ROP which may be impractical. Allowing the train crew to be relieved at any RP during the ROP will provide a better representation of real-world conditions and improve the robustness of the schedule. The second solution approach is the metaheuristic which consists of two phases. The first phase constructs initial solutions by a constructive heuristic and the second phase improves the obtained solutions by applying a hybrid constructive SA.

3.1 The mathematical programming model

This section presents a mathematical model for railway crew scheduling. The following notations are used through the description of the model.

3.1.1 Notations

Indices
- $i, i'$: trips
- $j, j'$: duties
- $k, k'$: shifts
- $ohd$: originate at crew home depot
- $thd$: terminate at crew home depot
- $orp$: originate at relief point
- $trp$: terminate at relief point
- $ots$: originate and terminate at any station

Sets
- $I$: set of all trips
- $I_{ohd}$: set of trips that originate at crew home depot ($I_{ohd} \subseteq I$)
- $I_{thd}$: set of trips that terminate at crew home depot ($I_{thd} \subseteq I$)
- $I_{orp}$: set of trips that originate at relief point ($I_{orp} \subseteq I$)
- $I_{trp}$: set of trips that terminate at the relief point ($I_{trp} \subseteq I$)
- $I_{ots}$: set of trips that can be sequential in the same duty ($I_{ots} \subseteq I$)
- $J$: set of duties
- $J_i$: set of duties which can contain trip $i$ ($J_i \subseteq J$)
- $K$: set of shifts
- $K_j$: set of shifts for duty $j$ ($K_j \subseteq K$)

Parameters
- $t_{ijk}$: driving time of trip $i$ in duty $j$ of shift $k$
- $\xi_{ii'}$: transition time from trip $i$ to trip $i'$ of the $j^{th}$ duty of shift $k$
transition time from trip $i$ of the 1st duty to the trip $i'$ of the 2nd duty of shift $k$

minimum duration of 1st part of duty in shift $k$

maximum duration of 1st part of duty in shift $k$

minimum duration of 2nd part of duty in shift $k$

maximum duration of 2nd part of duty in shift $k$

departure time of trip $i$

arrival time of trip $i$

departure station of trip $i$

arrival station of trip $i$

normal working time per shift

minimum working time allowed per shift

actual driving time in shift $k$

spread time of shift $k$

maximum spread time allowed per shift

The decision binary variables are defined as follows:

$$v_{jk}^i = \begin{cases} 
1 & \text{if trip } i \text{ is assigned in duty } j \text{ of shift } k; \\
0 & \text{otherwise} 
\end{cases}$$

$$w_{jk}^i = \begin{cases} 
1 & \text{if trip } i \text{ is the first trip of duty } j \text{ of shift } k; \\
0 & \text{otherwise} 
\end{cases}$$

$$x_{jk}^{ii'} = \begin{cases} 
1 & \text{if trip } i \text{ is followed by trip } i' \text{ in the same duty;} \\
0 & \text{otherwise} 
\end{cases}$$

$$y_{jk}^j = \begin{cases} 
1 & \text{if trip } i \text{ is the last trip of duty } j \text{ of shift } k; \\
0 & \text{otherwise} 
\end{cases}$$

$$z_{jk}^{ii'} = \begin{cases} 
1 & \text{if there is a transition from trip } i \text{ to trip } i' \text{ in the end of any partial duty and} \ 
\text{trip } i' \text{ is assigned in the subsequent partial duty;} \\
0 & \text{otherwise} 
\end{cases}$$

$$z_{jk}^{ii'} = \begin{cases} 
1 & \text{if there is a transition between trip } i \text{ at the end of a shift to be followed by} \ 
\text{trip } i' \text{ at the beginning of the subsequent shift;} \\
0 & \text{otherwise} 
\end{cases}$$

Additionally, the non-negative continuous variables $\sigma_{jk}^i$ and $\sigma_{jk}^l$ denote the start time of trip $i$ in duty $j$ of shift $k$ and the completion time of trip $i$ in duty $j$ of shift $k$, respectively. $U$ is a binary variable.

The objective function is designed to minimise the total number of duties by minimising idle transition times. The idle transition times includes the idle intervals.
between trips and an idle transition during a MB. The function consists of driving period and non-driving period.

\[
\text{Min } \left( \sum_{j \in J} \sum_{i \in I} t_{jk}^i v_{jk}^i + \sum_{j \in J} \sum_{i \in I} \sum_{i' \in I} \sum_{k \in K} \zeta_{jk}^{ii'} x_{jk}^{ii'} + \sum_{j \in J} \sum_{i \in I} \sum_{i' \in I} \sum_{k \in K} \zeta_{jj'}^{i'i} z_{jj'k}^{i'i} \right)
\]

\[\forall \ i, i' \in I, i \neq i', j, j' \in J, k \in K \]  

(1)

Equation (2) is the trip assignment. It enforces every trip \( i \) to be allocated in exactly one duty \( j \) of shift \( k \).

\[
\sum_{k \in K_j} \sum_{j \in J} v_{jk}^i = 1 \quad \forall \ i \in I \]

(2)

This equation implies that no deadheading is allowed. A crew has to wait for the next trip at a RP and the changeover of trains is at the same RP. When the assignment of trip \( i \) is followed by trip \( i' \) in the same duty, a sequence of the trips is enforced via constraint (3). Trips \( i \) and trip \( i' \) are consecutive only in the case that the binary variable \( x_{jk}^{ii'} = 1 \). Similarly, constraint (4) denotes that the assignment of trip \( i \) in duty \( j \) is followed by trip \( i' \) at the next duty \( j' \). The transition variable \( z_{jj'k}^{i'i} \) is activated when both \( v_{jk}^i \) and \( v_{jk}^{i'} \) are equal to one. As a result, one transition from trip \( i \) to trip \( i' \) occurs at the end of any duty if and only if trip \( i' \) is assigned in the subsequent duty.

\[
x_{jk}^{ii'} \geq v_{jk}^i + v_{jk}^{i'} - 1 \quad \forall \ i, i' \in I_k, i \neq i', j \in J, k \in K_j \]

\[z_{jj'k}^{i'i} \geq v_{jk}^i + v_{jk}^{i'} - 1 \quad \forall \ i, i' \in I_k, i \neq i', k \in K_j \]

(3)  

(4)

Constraint (5a) ensures that no overlap is allowed. The start time of trip \( i' \) in any duty requires the completion of the previous trip. Constraint (5b) and constraint (5c) are included to ensure a connectivity of the trip sequences.

\[
at_{jk}^i + \zeta_{jk}^{ii'} \leq dt_{jk}^{i'} \quad \forall \ i, i' \in I_k, i \neq i', j \in J, k \in K_j \]

\[a_{jk}^i = d_{jk}^i \quad \forall \ i, i' \in I_k, i \neq i', j \in J, k \in K_j \]

\[a_{jk}^i = d_{jk}^i \quad \forall \ j, j' \in J, k \in K_j \]

(5a)  

(5b)  

(5c)

Constraint (6a) and constraint (6b) denote the relation between the start and completion times in a duty. The completion time of the last trip in a duty is greater than or equal to the start time of the first trip plus the total driving time and the total transition time in the duty.
\begin{align}
\mathbf{a}t^{i}_jk & \geq \mathbf{d}t^{i}_jk + \sum_{i \in h} t^{i}_j k v^{i}_j k & \forall j, k \in K_j \quad (6a) \\
\sigma^{i}_jk & \geq \varphi^{i}_jk + \sum_{i \in h} t^{i}_j k v^{i}_j k + \sum_{i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \forall j, k \in K_j \quad (6b)
\end{align}

Constraint (7a), along with constraint (7b), constraint (8a), and constraint (8b), indicate that the total continuous driving time in the 1st part of a duty should be greater than or equal to the minimum allowable duration of the 1st part of a duty in shift \(k\) \((\varphi^{i}_jk)\) and the total continuous driving time of the 2nd part of a duty should be less than or equal to the maximum duration of the 2nd part of a duty in shift \(k\) \((\delta^{i}_jk)\). Otherwise, the total continuous driving time of the 1st part of a duty should be less than or equal to the maximum duration of the 1st part of a duty in shift \(k\) \((\varphi^{i}_jk)\) and the total continuous driving time of the 2nd part of a duty should be greater than or equal to the minimum duration of the 2nd part of a duty in shift \(k\) \((\delta^{i}_jk)\). The set of constraints satisfy a condition in which a train crew takes a MB at the earliest or latest times or in any time between the two limits.

\begin{align}
\sum_{i \in h} t^{i}_j k v^{i}_j k + \sum_{i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \geq \varphi^{i}_jk U & \forall j, k \in K_j \quad (7a) \\
\sum_{i \in h} t^{i}_j k v^{i}_j k + \sum_{i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \leq \varphi^{i'}_jk (1 - U) & \forall j, k \in K_j \quad (7b) \\
\sum_{i \in h} t^{i'}_j k v^{i'}_j k + \sum_{i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \leq \delta^{i}_jk U & \forall j, j' \in K, k \in K_1 \quad (8a) \\
\sum_{i \in h} t^{i'}_j k v^{i'}_j k + \sum_{i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \geq \delta^{i'}_jk (1 - U) & \forall j, j' \in K, k \in K_1 \quad (8b)
\end{align}

Equation (9a) calculates the total actual driving time in shift \(k\) \((W_{tk})\), which is equal to the total working time of all partial duties in the shift. Constraint (9b) states that the total actual driving time within the shift must not exceed the upper bound \((W_{tmax})\) and the lower bound \((W_{tmin})\).

\begin{align}
\sum_{j \in J, i \in h} t^{i}_j k v^{i}_j k & + \sum_{j \in J, i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & \leq W_{tk} & \forall j, k \in K_j \quad (9a) \\
W_{tmin} & \leq W_{tk} & \leq W_{tmax} \quad (9b)
\end{align}

Constraint (10) restricts the spread time of a shift from exceeding the maximum allowed total spread time. Spread time of a shift \((St_{tk})\) is equal to the total working time plus the transition time between each partial duty (MB).

\begin{align}
\sum_{j \in J, i \in h} t^{i}_j k v^{i}_j k & + \sum_{j \in J, i, i' \in h} \zeta^{i i'}_jk x^{i i'}_jk & + \sum_{i, i' \in h} \zeta^{i i'}_jk Z^{i i'}_jk & \leq St_{max} & \forall j, j' \in J, k \in K_1 \quad (10)
\end{align}
Considering that each duty consists of at least one trip, only one trip can be the first or the last one in each duty. Equation (11) expresses the requirement that the first trip in the 1st part of a duty which is also the first trip of the corresponding duty (shift) should originate from a HD. Equation (12) states that the last trip in a duty should terminate at a HD or at a RP. Equation (13) ensures that each trip, except the first trip, is assigned after another trip. Similarly, equation (14) ensures that each trip, except the last trip, is assigned before another trip. Equation (15) expresses that for each trip which terminated the 1st part of a duty, there is a transition time (MB) from this trip to the first trip in the subsequent part of a duty. Similarly, equation (16) expresses that for each trip which originated a duty, there is a transition time from the last trip of the previous duty to the current duty. Equation (17) ensures that for each trip which terminated a duty (shift), there is a transition time from this trip to the first trip of the next duty (sign off to sign on). Similarly, equation (18) ensures that for each trip which originated a duty (shift), there is a transition time from the last trip of the previous duty (shift) to the first trip of the current duty (shift).

\[
\sum_{i \in I_{o, o}} w_{jk}^i = 1 \quad \forall j \in J, k \in K_j
\] (11)

\[
\sum_{i \in I_{o, o} \cup I_{o, p}} y_{jk}^i = 1 \quad \forall j \in J, k \in K_j
\] (12)

\[
\sum_{i \in I_{o, o}, i \neq i'} x_{jk}^{ij} - x_{jk}^{i'} = v_{jk}^{i'} - w_{jk}^i \quad \forall i' \in I_{o, s}, j \in J, k \in K_j
\] (13)

\[
\sum_{i \in I_{o, o}, i \neq i} x_{jk}^{ii} = v_{jk}^i - y_{jk}^i \quad \forall i \in I_{o, s}, j \in J, k \in K_j
\] (14)

\[
\sum_{i \in I_{o, o} \cup I_{o, p}} z_{jk}^{ik} = w_{jk}^{i'} \quad \forall i' \in I_{o, o} \cup I_{o, p}, j, k \in K_j
\] (15)

\[
\sum_{i \in I_{o, o} \cup I_{o, p}} z_{jk}^{ij} = y_{jk}^i \quad \forall i \in I_{o, o} \cup I_{o, p}, j, k \in K_j
\] (16)

\[
\sum_{i \in I_{o, o} \cup I_{o, p}} z_{jk}^{ik'} = w_{jk}^{i'} \quad \forall i' \in I_{o, o}, j \in J, k \in K_j
\] (17)

\[
\sum_{i \in I_{o, o} \cup I_{o, p}} z_{jk}^{ik'} = y_{jk}^i \quad \forall i \in I_{o, o} \cup I_{o, p}, j \in J, k \in K_j
\] (18)

Generally, these constraints can be divided into three groups. The first group focuses on the scheduling and sequencing of trips (Eqs. (2) – (6)); the second group addresses the duty restrictions (Eqs. (7) – (10)); and the remaining group determines the assignment and sequencing of trips in a duty (Eqs. (11) – (18)). Because of the large number of decision variables and constraints of the proposed model, it is difficult to solve this model by exact methods, especially for large-sized instances. Therefore, we
propose a hybrid constructive heuristic with the SA metaheuristic for solving the problem. This is presented in the next section.

3.2 The metaheuristic

Most practical optimisation problems involve high complexity and require extensive computational times because of the number of potential solutions. The approximate methods are generally applied to resolve this type of problems. These methods are based on an iterative exploration of the search space to find a good quality solution in reasonable computational times. These approximate methods, among others are the neighbourhood methods, such as local search and SA.

3.2.1 Initial solution generation by constructive heuristic (CH)

Local search explores the neighbourhood $\mathcal{N}(s)$ of a current solution iteratively and finds a better solution $s' \in \mathcal{N}(s)$ according to some criteria. The initial solution is constructed by means of CH algorithm from an ordered list of trips with their attributes to form crew duties. We break down this phase into two sub-phases. The first is the initialising phase that includes listing all vehicle blocks in ascending order of start time, $v_b = \{v_{b1}, v_{b2}, \ldots, v_{bn}\}$; and grouping them based on the length of run, $l_r = \{l_{r1}, l_{r2}, \ldots, l_{rn}\}$. Cutting vehicle blocks into trip segments is also performed in this phase $t_s = \{t_{s1}, t_{s2}, \ldots, t_{sn}\}$. Some vehicle blocks may have sufficient length to be divided into two straight runs that are approximately equal to the length of regular working hours (8 h) each. Other vehicle blocks may be divided into one straight run of 8 h with a piece left over. The remaining of the vehicle blocks do not need to be divided as they have sufficient length to form one straight run with no pieces left over. The second is combining phase which is joining trip segments by progressively selecting uncovered trip segments from a block to create feasible duties. The CH method is described in a pseudo code form in Algorithm 1 in the Appendix. It is desirable to construct feasible schedules that will minimise idle transition times and maximise the length of the route per cycle time. The cycle time is the time spent to drive a roundtrip plus idle intervals while on a route.

3.2.2 Solution improvement by hybrid constructive SA (HCSA)

SA is motivated by an analogy to the physical process of annealing, where the temperature of a material is reduced to achieve its thermal equilibrium (Kirkpatrick et al., 1983). This principle is applied in combinatorial optimisation problems to optimise the objective function value. The advantage of this technique is that it can avoid local optima by occasionally allowing the acceptance of non-improving solutions in the hope that a better solution may be found later on.

We utilise the SA metaheuristic to improve solution and to derive a near-optimal solution. The design of SA algorithm to solve the railway crew scheduling generally consists of four components, an objective function (analogue of energy) to be optimised; the neighbourhood structure that defines how to efficiently generate random solutions from neighbourhood; an acceptance criterion that is a criterion for accepting or
rejecting a new generated solution; and a cooling schedule. Implementation details of
the proposed HCSSA algorithm are given as follows:

a) **Initial sequence.** An initial schedule is obtained from the best schedule returned by
the CH algorithm. This schedule is assumed as the current solution. A set of
scheduled trips, \( J = \{ i_1, i_2, i_3, \ldots, i_n \} \), that need to be serviced during a
defined period of time is identified by its departure station, departure time, arrival station,
arrival time, represented by vector \( ds_i, dt_i, as_i, at_i \), respectively. The algorithm sorts
an array \( J = \{ i(0), \ldots, i(n-1) \} \) of \( n \) trips in increasing order of departure time. Every
iteration removes an element from the input data, inserting it into the correct
position and simultaneously moves the data in the already-sorted list, until no input
elements remain. Initialising can be considered as the process of queueing all the
trips in the right order to their assigned duties. The constraints considered in this
case are connectivity restrictions, travelling times and transition times.

b) **Neighbourhood structure.** The neighbourhood structure defines a method of
generating alternative solution from a current solution. We generate the
neighbourhood using swap and insert mechanisms. The neighbourhood structure
proposed in Elizondo et al. (2010) is adapted for our problem. Two different duties
\( s_x \) and \( s_y \), with the number of trips \( n \) and \( w \), respectively, are selected and denoted as,
\( s_x = \{ i_{x1}, i_{x2}, \ldots, i_{xm}, i_{xm+1}, \ldots, i_{xw} \} \) and \( s_y = \{ i_{y1}, i_{y2}, \ldots, i_{yn}, i_{yn+1}, \ldots, i_{yw} \} \). The
swap operation is performed on the selected duties by exchanging the position of
trip segments \( (t_{i}) \) between two blocks. Thus

\[
\text{swap } (t_x, t_y) = \left[ \begin{array}{c}
\{ i_{x1}, i_{x2}, \ldots, i_{xm}, i_{xm+1}, \ldots, i_{xw} \} \\
\{ i_{y1}, i_{y2}, \ldots, i_{yn}, i_{yn+1}, \ldots, i_{yw} \}
\end{array} \right]
\]

The swap operation is only performed on duties with trip segments originate and
terminate at the same crew depot. The insert operation is performed by moving one
trip segment to another duty. This operation is only applied to the trip segments that
arrive and depart from stations with a local connection.

c) **Acceptance criterion.** Given the initial configuration, a small perturbation is
performed by exchanging a piece of the trip between two duties and moving a piece of
the trip within one duty to another. The change in the objective function value is
then calculated. If it gives a better solution, the new solution is accepted. Otherwise
it still has a chance to be accepted with a particular condition that is the value of the
function \( f(\Delta E) = e^{-\Delta E/T} \) is greater than a randomly generated value between 0 and
1. When the solution is accepted, the current neighbourhood configuration is
updated as the algorithm proceeds.

d) **Cooling schedule.** An annealing or a cooling schedule consists of (i) the initial value
of temperature parameter \( T_0 \), (ii) the cooling factor (a method of gradually
decreasing the value of \( T^c \)), (iii) the number of iterations to be performed at each \( T^c \)
before it is decreased, and (iv) the stopping criterion to terminate the algorithm.

The overall method of the proposed algorithm is captured in the pseudo-code form in
Algorithm 2 in Appendix A.
4. Computational experiments

To evaluate the scheduling methods presented in this paper, we generated benchmark instances for the problem with 24-h scheduling horizons. A sample train schedule with 12 trips is given in Table 1. The railway crew scheduling in this study is to create a feasible set of crew duties to cover a given set of trips. A feasible crew duty (shift) includes one or two partial duties, a period of MB, idle transition times, and the sign-on and sign-off activities. The accumulated time represents the total crew working time.

<table>
<thead>
<tr>
<th>Train ID</th>
<th>Dep. Station (ds)</th>
<th>Dep. Time (dt) (hh:mm)</th>
<th>Arr. Station (as)</th>
<th>Arr. Time (at) (hh:mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L001</td>
<td>A</td>
<td>05:00</td>
<td>B</td>
<td>05:24</td>
</tr>
<tr>
<td>L002</td>
<td>A</td>
<td>05:30</td>
<td>B</td>
<td>05:54</td>
</tr>
<tr>
<td>L003</td>
<td>A</td>
<td>06:00</td>
<td>B</td>
<td>06:24</td>
</tr>
<tr>
<td>L001</td>
<td>B</td>
<td>05:25</td>
<td>C</td>
<td>06:35</td>
</tr>
<tr>
<td>L002</td>
<td>B</td>
<td>05:55</td>
<td>C</td>
<td>07:03</td>
</tr>
<tr>
<td>L003</td>
<td>B</td>
<td>06:25</td>
<td>C</td>
<td>07:34</td>
</tr>
<tr>
<td>L201</td>
<td>C</td>
<td>04:28</td>
<td>B</td>
<td>05:41</td>
</tr>
<tr>
<td>L202</td>
<td>C</td>
<td>04:59</td>
<td>B</td>
<td>06:08</td>
</tr>
<tr>
<td>L203</td>
<td>C</td>
<td>05:27</td>
<td>B</td>
<td>06:37</td>
</tr>
<tr>
<td>L201</td>
<td>B</td>
<td>05:42</td>
<td>A</td>
<td>06:08</td>
</tr>
<tr>
<td>L202</td>
<td>B</td>
<td>06:08</td>
<td>A</td>
<td>06:38</td>
</tr>
<tr>
<td>L203</td>
<td>B</td>
<td>06:38</td>
<td>A</td>
<td>07:02</td>
</tr>
</tbody>
</table>

The exact solution of the mathematical model was obtained using Xpress-Optimizer (FICO) algorithms for mixed integer problems. We considered 3 HDs and 5 RPs. The number of trips varied between 25 and 120 for instances solved by Xpress-Optimizer and the HCSA algorithm. Whereas the number of trips varied between 258 and 732 for instances solved by the CH and HCSA algorithms. We divided all trips in a day into four different intervals, 05.00–08.59; 09.00–12.59; 13.00–16.59; and 17.00–22.59. Normal daily working time was fixed to 8 h and maximum spread time allowed was 12 h. The minimum and maximum lengths of working periods of the 1st part of a duty were 3 h and 5.5 h, respectively. Whereas the minimum length and maximum length of working periods of the 2nd part of a duty were set to 2 h and 4.5 h, respectively. The length of the ROP was 2.5 h within which a MB of minimum 0.5 h is required between the third and the sixth hours of an 8 h duty. There was a time allowance of about 10 min for sign-on or sign-off when a crew starts or ends his duty at a HD. Table 2 gives
computational results, i.e. number of feasible duties, total objective values, driving times, and run times in seconds.

All small-sized instances were solved to optimality by Xpress-Optimizer. As can be seen from Table 2, the computational time increases significantly as the size of the instance becomes larger. The largest instance was solved by Xpress-Optimizer with a reasonable computational time. The HC-SA algorithm was able to solve all small-sized problems with computational time less than one second. We used relative percentage deviation as a performance measure to further evaluate the obtained solutions for the small-sized problems. This was calculated by the following equation:

\[
\text{Gap} (\%) = \frac{[S_{(\text{Alg})} - S_{(\text{Ext})}]}{S_{(\text{Ext})}} \times 100\%
\]

where \(S_{(\text{Alg})}\) is the objective value of the solution obtained by the HC-SA algorithm and \(S_{(\text{Ext})}\) is the objective value of the optimal solution given by the Xpress Optimizer.

Table 2. Computational results by the mathematical model and the HC-SA algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>No. of Trips</th>
<th>Feasible Duties (FDs)</th>
<th>Objective value (Ext)</th>
<th>Driving Time (sec)</th>
<th>Objective value (Ext)</th>
<th>CPU time (sec) (Ext)</th>
<th>Objective value (HCSA)</th>
<th>CPU time (sec) (HCSA)</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM-01</td>
<td>25</td>
<td>6</td>
<td>2944</td>
<td>0.95</td>
<td>0.42</td>
<td>2944</td>
<td>0.0001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>11</td>
<td>5407</td>
<td>0.93</td>
<td>0.88</td>
<td>5407</td>
<td>0.0001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>65</td>
<td>16</td>
<td>7851</td>
<td>0.93</td>
<td>1.09</td>
<td>7851</td>
<td>0.0002</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DM-02</td>
<td>70</td>
<td>23</td>
<td>11288</td>
<td>0.94</td>
<td>1.73</td>
<td>11288</td>
<td>0.0004</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>28</td>
<td>13710</td>
<td>0.91</td>
<td>3.50</td>
<td>13710</td>
<td>0.0006</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>33</td>
<td>16205</td>
<td>0.92</td>
<td>9.24</td>
<td>16205</td>
<td>0.0019</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DM-03</td>
<td>110</td>
<td>35</td>
<td>17166</td>
<td>0.87</td>
<td>121.51</td>
<td>17166</td>
<td>0.0033</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>115</td>
<td>36</td>
<td>17685</td>
<td>0.82</td>
<td>2262.04</td>
<td>17685</td>
<td>0.0085</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>38</td>
<td>18601</td>
<td>0.80</td>
<td>15927.33</td>
<td>18601</td>
<td>0.0127</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The proposed algorithms were implemented in Microsoft Visual C# and run on an Intel Core 2 Duo 1.96 GHz Processor with 3.46 GB of RAM under Microsoft Windows XP operating system. The computational results obtained from both the CH algorithm and the HC-SA algorithm are summarized in Table 3. The number of trips per duty varies because of the length of a trip also varies. On average, the number of trips in each duty varies from 3 to 6 trips. The problem of smaller size corresponds to a higher percentage of driving time with less computational times. This is because smaller sized problems can be better optimised due to a more exhaustive search. For larger problems, long idle transition times remain high as indicated by a lower percentage of driving time. This is due to the fact that services at different times of the day have different frequency. Early morning and late afternoon hours have higher service frequency than that of the middle day. It seems that the more the trips we include the higher probability of the delay.
The driving time is used to measure the performance of the obtained schedules (productivity rate). Driving time of the crew is the ratio between the total working time ($Wt$) and the total spread time or elapsed time ($Et$) in a duty. Excess cost ($Ec$) was calculated as follows. $Ec(\%) = \frac{(Et - Wt)}{Wt} \times 100\%$. Both the CH and the HCSA algorithms produced acceptable solutions, although the produced solutions are not guaranteed to be an optimal solution. The CH algorithm was sometimes unable to include some trip segments and left them out unscheduled. In all cases, the HCSA algorithm was able to produce better solutions than the CH in terms of the solution quality and the runtime. As can be seen from Table 3, the HCSA algorithm significantly improves the solution produced by the CH. The HCSA algorithm increases the average driving time by 3.06% and decreases the average excess cost by 3.35%. Furthermore, the number of leftovers in the HCSA is smaller than that in the CH. Overall, the HCSA algorithm increases the total crew working time and reduces the number of crew duties for all data sets. As the number of crew duties corresponds to the number of crew needed, significant savings can be gained on the annual cost of crew related expenses.

To measure the quality of solutions obtained by the algorithms, the upper and lower bound values were calculated as follows. $Q = \frac{\text{objective value} - \text{LB}}{\text{UB} - \text{LB}}$ where $0 \leq Q \leq 1$ (Burdett and Kozan, 2010). The equation describes approximately the quality of the solution in the search space. If $Q$ value is close to zero then the obtained solution is near to the optimal solution. The $Q$ values of solutions obtained by both the CH and HCSA algorithms can be seen in the chart of Fig. 3. The $Q$ value shown in Fig. 3 is enough to validate the quality of the proposed algorithms.

![Figure 3. $Q$ values for the CH and HCSA solutions](image-url)
With regard to the SA algorithm, initial temperature $T_0$ was set to be large to allow the search exploring some areas of the solution space with low level quality solutions hence, accepting a worse solution at the beginning of the search. A large neighbourhood is more attractive because it tends to find much better solution in one local search step than in the small neighbourhoods. However, a large neighbourhood is associated with long search process to come up with a better neighbouring solution. Large neighbourhood may also lead to a potential bottleneck for local search when searching a better neighbouring solution. The temperature was updated in each iteration by applying geometric cooling schedule where $T^c = \alpha T^c$. We applied a cooling factor of 0.93, 0.95, and 0.97 as the temperature reduction should be controlled by a constant cooling factor with the value approximately close to one (Kirkpatrick, 1983). Altering this parameter however, did not have a significant effect on the final objective value after 500 iterations.
<table>
<thead>
<tr>
<th>Instance</th>
<th>No. of Trips</th>
<th>No. of Duties</th>
<th>Objective value</th>
<th>Total Elapsed Time (min)</th>
<th>Driving Time (%)</th>
<th>Excess Cost (%)</th>
<th>CPU Time (m:s)</th>
<th>No. of Duties</th>
<th>Objective value</th>
<th>Total Elapsed Time (min)</th>
<th>Driving Time (%)</th>
<th>Excess Cost (%)</th>
<th>CPU Time (m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH-01</td>
<td>258</td>
<td>112</td>
<td>54882</td>
<td>60376</td>
<td>90.90</td>
<td>10.01</td>
<td>00:12</td>
<td>105</td>
<td>51345</td>
<td>54045</td>
<td>95.00</td>
<td>5.26</td>
<td>00:07</td>
</tr>
<tr>
<td>DH-02</td>
<td>350</td>
<td>121</td>
<td>60257</td>
<td>66935</td>
<td>90.02</td>
<td>11.08</td>
<td>00:20</td>
<td>117</td>
<td>58146</td>
<td>62526</td>
<td>92.99</td>
<td>7.53</td>
<td>00:14</td>
</tr>
<tr>
<td>DH-03</td>
<td>425</td>
<td>139</td>
<td>68943</td>
<td>78522</td>
<td>87.80</td>
<td>13.89</td>
<td>00:28</td>
<td>133</td>
<td>65825</td>
<td>72338</td>
<td>91.00</td>
<td>9.89</td>
<td>00:18</td>
</tr>
<tr>
<td>DH-04</td>
<td>455</td>
<td>156</td>
<td>79095</td>
<td>91125</td>
<td>86.80</td>
<td>15.21</td>
<td>00:43</td>
<td>142</td>
<td>71722</td>
<td>79693</td>
<td>90.00</td>
<td>11.11</td>
<td>00:32</td>
</tr>
<tr>
<td>DH-05</td>
<td>470</td>
<td>153</td>
<td>85526</td>
<td>96218</td>
<td>88.89</td>
<td>12.50</td>
<td>00:57</td>
<td>145</td>
<td>80762</td>
<td>87805</td>
<td>91.98</td>
<td>8.72</td>
<td>00:48</td>
</tr>
<tr>
<td>DH-06</td>
<td>485</td>
<td>160</td>
<td>80480</td>
<td>98425</td>
<td>90.00</td>
<td>11.11</td>
<td>01:07</td>
<td>157</td>
<td>78657</td>
<td>85476</td>
<td>92.02</td>
<td>8.67</td>
<td>00:55</td>
</tr>
<tr>
<td>DH-07</td>
<td>550</td>
<td>157</td>
<td>84309</td>
<td>95807</td>
<td>88.00</td>
<td>13.64</td>
<td>01:29</td>
<td>155</td>
<td>83120</td>
<td>92352</td>
<td>90.00</td>
<td>11.11</td>
<td>01:04</td>
</tr>
<tr>
<td>DH-08</td>
<td>582</td>
<td>162</td>
<td>82135</td>
<td>94413</td>
<td>87.00</td>
<td>14.95</td>
<td>01:52</td>
<td>162</td>
<td>81840</td>
<td>92007</td>
<td>88.95</td>
<td>12.42</td>
<td>01:33</td>
</tr>
<tr>
<td>DH-19</td>
<td>607</td>
<td>169</td>
<td>82810</td>
<td>97538</td>
<td>84.90</td>
<td>17.79</td>
<td>02:04</td>
<td>167</td>
<td>81679</td>
<td>93884</td>
<td>87.00</td>
<td>14.94</td>
<td>01:56</td>
</tr>
<tr>
<td>DH-10</td>
<td>625</td>
<td>172</td>
<td>84795</td>
<td>97712</td>
<td>86.78</td>
<td>15.23</td>
<td>02:26</td>
<td>172</td>
<td>84965</td>
<td>96404</td>
<td>88.13</td>
<td>13.46</td>
<td>02:10</td>
</tr>
<tr>
<td>DH-11</td>
<td>660</td>
<td>176</td>
<td>87120</td>
<td>102294</td>
<td>85.17</td>
<td>17.42</td>
<td>05:41</td>
<td>174</td>
<td>86371</td>
<td>97999</td>
<td>88.13</td>
<td>13.46</td>
<td>05:36</td>
</tr>
<tr>
<td>DH-12</td>
<td>732</td>
<td>182</td>
<td>89874</td>
<td>105704</td>
<td>85.02</td>
<td>17.61</td>
<td>09:55</td>
<td>178</td>
<td>88255</td>
<td>100285</td>
<td>88.00</td>
<td>13.63</td>
<td>09:42</td>
</tr>
</tbody>
</table>
5. Conclusion

In this paper, a mathematical model and algorithms for railway CSP are presented. The objective of the model and algorithms is to minimise the number of crew duties by minimising total idle transition times. The idle transition times include idle intervals between trips and an idle interval between partial duties. These unproductive parts of a crew duty contribute the most to the optimisation potential of the crew scheduling. The mathematical model includes the interval of relief opportunities, allowing a train crew to be relieved at any relief point during the ROP. The problem is mathematically intractable due to the number of possible trip combinations and the complexity of the involved constraints. To handle the difficulty due to the combinatorial explosion of the problem, a constructive heuristic coupled with the SA algorithm is proposed to solve the problem. The overall results indicate that the proposed algorithms can produce near-optimal railway crew schedules of large-sized problem instances within an acceptable computational time. This study also shows the effectiveness of the hybridization of a SA-based algorithm in solving a highly constrained combinatorial optimisation problem. A further analysis on the performance of the model and algorithms by varying parameter settings is going to be done in the near future.

Although we have developed a model for railway crew scheduling, this model can be applied to other modes of transportation. The model and solution techniques presented in this paper can be improved and extended in several ways such as:

- The proposed model deals with the construction of duties (shifts) with one period of relief opportunities (straight runs). This model can be extended to model a situation in which a duty may contain more than two pieces of work (split runs).
- The model can be applied to the integration of vehicle and CSPs with ROP.
- Accessing and comparing the performance of the SA algorithm by implementing other metaheuristics such as Tabu Search and Genetic Algorithm.

Appendix A

The following notations are used through the description of the HCSA algorithm.

\[ S \]: set of feasible solutions
\[ \mathcal{N}(s) \]: set of neighbourhood solutions
\[ s' \]: generated solution (sample solution from neighbourhood) \( s' \in S \)
\[ s^* \]: current solution
\[ s^\circ \]: best solution found
\[ f(S_{max}) \]: function value of neighbourhood solution
\[ f(S_{max}^c) \]: function value of current solution
\[ f(S_{max}^b) \]: function value of best solution
\[ T_0 \]: initial temperature
\[ T^c \]: current temperature
\[ \mathcal{R} \]: uniformly distributed random number between 0 and 1
\[ \alpha \]: cooling rate
\[ t_{max} \]: maximum iterations
The pseudo code of the HCSA algorithm is as follows:

Begin

Algorithm 1: Generating initial solutions

Input all relevant data: trip list, vehicle blocks, parameters, and constraints
Output: initial solutions $S^0$

begin Initialisation ()
first phase
\( i \leftarrow 1 \) to \( I \) (I total number of trips) \( \forall \ i \in v_b; \)
\( v_b \leftarrow \{v_{b1}, v_{b2}, v_{b3}, \ldots, v_{bn}\} \) in ascending starting time order, \( \forall \ i \in v_b; \)
\( l_i \leftarrow \{l_{i1}, l_{i2}, \ldots, l_{in}\} \) \( \forall \ i \in v_b; \)
\( t_i \leftarrow \{t_{i1}, t_{i2}, \ldots, t_{in}\} \) \( \forall \ i \in v_b; \)
second phase
\( \zeta_n \leftarrow 1 \) to \( N \) (N number of trip segments);
list trip segments sequentially;
\( S^0 = \emptyset \)
\( \zeta_n \neq \emptyset \)
while (\( \zeta_n \leq t_n - 1 \)) do
allocate trip segments into time slots based on the starting time of the trip;
\( S^0 \leftarrow S^0 \cup \{ \zeta_i \}; \)
\( \zeta_n \leftarrow \zeta_n \setminus \{ \zeta_i \}; \)
determine possible trip segment combinations;
end while
\( S^0 \leftarrow S^0 + 1; \)
end
return (\( S^0 \))

Algorithm 2: Simulated Annealing

Input : initial solutions $S^0$
Output: duties

begin Simulated Annealing()
step
select an initial solution and set it as the current solution
\( s^c \leftarrow s \in S; \)
calculate \( S_{\text{max}}^c \)
\( S^b \leftarrow s^c; \)
\( f(S_{\text{max}}^b) \leftarrow f(S_{\text{max}}^c); \)
select an initial temperature, \( T_0 \)
\( T^c \leftarrow T_0; \)
select maximum iterations \( i_{\text{max}} \)
select temperature reduction function, \( \alpha \) (cooling rate)
initialise step counter \( i \leftarrow 0; \)
define neighbourhood structure();
iterative step
while (\( i < i_{\text{max}} \) and \( T^c > T_0 \)) do
search neighbourhood;
generate solution from neighbourhood, \( s \in N(s); \)
\( s^i \leftarrow s \in N(s); \)
apply the swap and insert mechanisms;
evaluate sample solution from neighbourhood;
\( \Delta E = f(S_{\text{max}}^c) - f(S_{\text{max}}^b) \)
if \( \Delta E < 0 \) then
\( s^c \leftarrow s^i; \)
if \( f(S_{\text{max}}^c) < f(S_{\text{max}}^b) \) then
\[
f(S_{\text{max}}^b) \leftarrow f(S_{\text{max}}^c)
\]
end if
else
generate random number \( R \sim (0,1) \);
\[ P_{\text{accept}} = e^{-\Delta E/T} \]
if \( R < P_{\text{accept}} \) then
\[ s^c \leftarrow s' \]
\[ f(S_{\text{max}}^c) \leftarrow f(S_{\text{max}}^c) \]
end if
\[ T^c \leftarrow \alpha T^c \]
\[ i \leftarrow i + 1 \]
end if
update temperature \( T \);
\[ T^c \leftarrow T^c(i) \]
return \((s^b, f(S_{\text{max}}^b))\)
end while
end

References


