A vehicle scheduling problem with fixed trips and time limitations

Deniz Türsel Eliiyia,*, Arslan Orneka, Sadık Serhat Karakütükb

a Izmir University of Economics, Izmir 35330, Turkey
b Dokuz Eylül University, Izmir 35100, Turkey

ARTICLE INFO

Article history:
Received 22 February 2007
Accepted 8 October 2008
Available online 1 November 2008

Keywords:
Vehicle scheduling
Fixed job scheduling
Time constraints
Heuristics

ABSTRACT

We consider minimum-cost scheduling of different vehicle types on a predetermined set of one-way trips. Trips have predetermined ready times, deadlines and associated demands. All trips must be performed. The total time of operations on any vehicle is limited. We develop a mixed integer model to find the optimal number of vehicles at a minimum cost. Based on the hard nature of the problem, we propose six heuristics. Computational results reveal that heuristics return exceptionally good solutions for problem instances with up to 100 jobs in very small computation times, and are likely to perform well for larger instances.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The vehicle scheduling problem (VSP) consists of assigning a set of scheduled trips to a set of vehicles, in such a way that each trip is associated with one vehicle and a cost function is minimized (Baita et al., 2000). VSPs have different objective functions like minimizing the fixed cost or maximizing the utilization of vehicles. VSP is modeled by integer programming.

There are efficient algorithms for some versions of the VSP, i.e., when all vehicles are equal and share the same depot. Nevertheless, real-life applications may turn out to be complex due not only to the dimension of the problem but also, and more importantly, to particular requirements which are inherent in practical situations. For example, there may be several vehicle types to choose from. Vehicle types may differ according to capacity or the cost of operation. Another complicating factor may be time windows, where visits to customer locations have to be made at specific times like postal and bank pickups and deliveries. Also, split deliveries may be of concern when one customer has a particular requirement, and it makes sense to have more than one vehicle assigned to that customer. As a result, most real VSPs are difficult for modeling and complex for solving.

In this study, we consider the problem of determining the optimal number of different types of vehicles at a minimum cost to meet a given schedule of trips with varying demands. Each vehicle can be run for a given time interval. That is, a spread time constraint imposes an upper bound $S$ on the total time between the start and the finish times of the operations on any vehicle. There may be idle times of the machine between these two time points, yet these times are included in the spread time.

Our study is motivated by a real-life VSP, which has many complicating factors. In a games event organization, there are different vehicle types having different capacities and costs of operation. The daily starting and ending times of trips are fixed in advance, and each trip has to be started at the exact starting time, otherwise it cannot be performed. There are certain trips which have to be split (because of capacity restrictions of vehicles). On the other hand, the passengers of certain other trips must be carried together; that is, some trips cannot be split. There are predetermined departure and arrival points for each trip. Hence, a trip can be taken by a specific vehicle depending on the previous trips taken by that vehicle. With these
characteristics, our VSP problem may be modeled as a tactical fixed job scheduling (TFJS) problem, as the ready times and the deadlines of the trips (jobs) are fixed, and the objective is to minimize the total cost of vehicles (machines) to perform all trips. However, unlike a typical TFJS problem, there are sequence-dependent setup times and capacity restrictions in our model.

The TFJS problem is a special case of interval scheduling (IS) problems. IS is an area of scheduling where tasks, each with specified ready times and deadlines, are to be processed on a number of resources. The problem is typical for reservation systems and has many real-life applications such as classroom assignment, transportation systems, and shift scheduling. A reservation system typically involves a parallel resource environment. Using conventional scheduling terminology, the problem environment can be represented as follows: There are \( N \) independent jobs available to be processed in a non-preemptive fashion on \( M \) parallel machines. A mapping \( g : N \rightarrow 2^M \) determines on which machines each job can be scheduled. The time window of job \( j \) is specified by a ready time \( r_j \) and a deadline \( d_j \). The problem of IS is to find a schedule of operations on the machines, such that each job is started on or after its ready time and completed before its deadline. If the time window of a job is larger than its processing time; that is, if it may be started after its ready time, the problem is referred to as the variable job shop scheduling (VJS). On the other hand, if the job cannot be delayed after its ready time, then the IS problem becomes a fixed job scheduling (FJS) problem. The processing time \( p_j \) of job \( j \) is equal to \((d_j - r_j)\) in the FJS problem, where \( p_j \) is smaller than or equal to \((d_j - r_j)\) in the VJS.

The FJS problem has two variants based on the objective functions. The first is the operational fixed job scheduling (OFJS) problem, in which each job has a weight \( w_j \) that represents its value for the decision maker, and maximizing weighted number of processed jobs with a given number of processors is of concern. When all jobs have equal weights, i.e., when \( w_j = w \) for all \( j \), the objective reduces to maximizing the number of jobs processed. Bouzina and Emmons (1996) provide an \( O(n \log n) \) algorithm for solving the OFJS problem with the objective of the maximization of the number of processed jobs. They show that the preemptive and non-preemptive versions of the problem with the objective of maximizing total weight are NP-hard.

The second variant is the TFJS problem, where there is a fixed cost \( c_k \) associated with machine \( k \), and the objective is the minimization of the total cost of the machines needed to process all available jobs. In a study by Kroon (1990), the TFJS problem is used as the core model in tactical capacity planning of aircraft maintenance personnel for an airline company.

The machines can be of different types and they can have different costs. When the machines are restricted, then a given job can only be scheduled on a subset of the machines. In this case, the problem is known to be a FJS problem with eligibility constraints (FJSE). Arkin and Silverberg (1987) show that the feasibility problem for the operational FJSE problem is NP-complete, when the number of machine classes is greater than two. They also present an \( O(nm+1) \) time dynamic programming-based algorithm for the problem.

Two different generalizations of operating time constraints of FJS problem, namely spread time and working time constraints, have been considered in the literature. Spread time constraints impose an upper bound on the time between the start and finish times of the operations on any machine. The working time constraints impose an upper bound on the sum of the processing times of the operations assigned to each processor. TFJS problem with spread time constraint is considered by Fischetti et al. (1987). In this problem each processor is available only for \( S \) time units from the ready time of the earliest task assigned to it, in the sense that any pair of tasks \((T_j, T_k)\) that are assigned to a processor must satisfy the relation \( d_j - r_k \leq S \). The idle time between the start and finish times are included in the spread time. The authors model the problem and show that the problem is NP-hard. They propose a branch and bound algorithm for the exact solution of problem.

Eliyi and Azizoglu (2006) recently consider the OFJS problem on identical machines. The objective is to maximize total weight. They show that the problem is strongly NP-hard, and investigate several special polynomially solvable cases. They propose a branch and bound algorithm that returns optimal solutions for large problem instances in reasonable solution times.

Bekk and Azizoglu (2008) address an OFJS problem on uniform parallel machines. Their problem is different from ours in the sense that the processing speed of machines are not identical. That is, a speed factor is defined for each machine, and the processing time of a job on a specific machine is defined as the product of the base processing time for the job and the speed factor of that machine. They show that the problem is strongly NP-hard and develop polynomial time algorithms for some special cases. They also propose a branch and bound algorithm for the problem.

Zäpfel and Bögl (2008) are other authors who have worked on a related problem. They consider the weekly planning of the pickup and delivery tours for a postal company in the presence of fluctuating number of shipments, time windows for demand points, and variable vehicle capacities. They model the problem as a vehicle routing problem with time windows, and propose a hybrid metaheuristic which is shown to be suitable for practice.

Kovalyov et al. (2007) provides a thorough review on the applications and theory of the FJS problem. To the best of our knowledge, there is no study in the FJS literature with sequence-dependent setup times and time limitations as in our model. Another very recent survey by Kolen et al. (2007) on IS problems provides a very useful review on the area of IS and presents proofs of results that have been known within the community for some time. The authors motivate the relevance of IS problems by providing an overview of applications that have appeared in literature. They also focus on algorithmic results for IS problems with machine availability constraints. The interested reader is referred to these studies to investigate more on the theory and applications of the IS problem.
In the next section, we provide the specifics of our problem and present the mathematical model. Most variants of TFJS problems are known to be NP-hard, and mathematical models are able to solve only small instances of the problems. Therefore, in Section 3, we propose three heuristic methods with their variants in order to obtain approximate solutions for the problem. Next, in Section 4, through computational results, we show that our heuristics are able to provide approximate solutions for the problem in very small times and they perform quite well. Finally, we present our conclusions along with some future research directions.

2. Problem definition and the mathematical model

The objective of the problem is to create cost-efficient vehicle schedules, which will service in an international games event organization. The vehicles perform the transportation between the participants’ accommodation facilities, and the facilities where the activities will take place, in compliance with the terms of the organization. Problem should be solved daily; that is, a new schedule should be made for each day as last-minute daily schedule changes are typical and the program for the next day is revealed at the end of the previous day.

2.1. Trips and vehicles

A trip is the act of transportation of passengers from a center to another, and it is required to be performed in a certain fixed time interval. The required number of daily trips is known in advance. A trip has only one pair of departure and destination centers, i.e., trips cannot consist of consecutive transportation between three or more centers.

Each trip \(j (j \in \{1, \ldots, N\})\) has a fixed ready time \((r_j)\) and a deadline \((d_j)\), which correspond to departure and arrival points on time axis, as seen in Fig. 1. Hence, trips in the time axis correspond to time intervals, which sometimes overlap. The ready time and deadline are known in advance and deterministic. Thus, every single trip \(j\) has a definite processing time \((p_j = d_j - r_j)\). The vehicle to be assigned to a specific trip necessarily has to be at the departure point exactly on the ready time of that trip; otherwise the trip is considered as lost.

Demand for a trip \(j\) \((D_j)\) corresponds to the number of passengers on that trip. Demands are deterministic and known in advance. The demand for each trip has to be satisfied. There are two kinds of trips based on their demands. The demand of the first type of trip may be split, i.e., more than one vehicle may be employed in order to meet the demand. The demand is split when it exceeds the capacity of the vehicle with the largest capacity. Smaller demands may also be split if the resulting schedule is less costly.

The second type of trips, i.e., team trips, includes the ones which are required to be carried by a single vehicle, due to some organizational conditions. Obviously, this constraint requires an assumption that the largest demand of such type should be equal to or smaller than the capacity of the largest vehicle. Otherwise, an assignment without a split will not be possible, and the problem would be infeasible. The set of second type trips on a given day is known in advance, and is denoted by \(T\).

When a vehicle takes trip \(b\) after trip \(j\), there is a certain time during which the vehicle traverses the distance from the arrival point of trip \(j\) to the departure point of trip \(b\). We denote this as the setup time \((a_{jb})\) of trips \(j\) and \(b\). In a manufacturing context, this time would correspond to the changeover time of the machine from the latest task to its new task. As it can be noted easily, the setup times are sequence-dependent.

It is assumed that there are enough vehicles from each type to meet the demand of the daily trips, as the vehicles are leased.

The index for vehicles is \(i\), where \(i \in \{1, 2, \ldots, I_1, I_1 + 1, \ldots, I_2, I_2 + 1, \ldots, I_K\}\). \([k - I_{k-1}]\) is an upper bound on the number of available vehicles of type \(k\), \(k = 1, \ldots, K\). There are \(K\) types of vehicles, having different capacities and costs. \(I_1\) denotes the number of available vehicles of type 1, \((I_2 - I_1)\) gives the number of available vehicles of type 2, and \((I_K - I_{K-1})\) gives the number of available vehicles of type \(k\). These numbers are determined to be nonbinding, according to our assumptions. We use this kind of indexing in order to keep the mathematical formulation simpler.

Each vehicle \(i\) has a definite capacity \((c_i)\). Capacity limits are strict, yet in order for a vehicle to complete a trip not all the seats have to be occupied. Even though only one person is to be transported, a vehicle should be assigned to that trip. As mentioned before, each vehicle can be assigned to only one trip and can transport passengers between two centers at a single slot of time. In other words, even though the vehicle’s capacity is

![Fig. 1. The time intervals of trips.](image-url)
adequate for more, it is allowed to transport the passengers of a single trip.

The average speed of the vehicles is assumed to be constant. Based on this assumption, it has been mentioned before that the trip and setup times are deterministic. These times are estimated based on distance data and a constant average speed for vehicles.

Each vehicle has a daily fixed cost dependent on its capacity. When a vehicle $i$ is assigned to a trip, its fixed cost ($f_i$) is incurred, and the right to use the vehicle for a certain period is acquired. This period is called the daily available regular time, or the spread time limit of the vehicles ($S$). In other words, the spread time limit imposes an upper bound on the time between the starting time of the first trip and the finishing time of the last trip on any vehicle. The spread time limit is constant for all vehicles. Throughout this period, the vehicle can be used without extra charges. In addition, each vehicle $i$ has a variable cost ($v_i$). Variable cost is the unit cost of overtime ($o_i$) usage of vehicle $i$. If the vehicle is used beyond its spread time, variable cost is charged for every overtime unit. Each vehicle also has a constant daily available overtime limit ($O$). Hence, any vehicle can be used daily for a time ($S+O$) at most.

2.2. The incompatibility set

The set of the trips that cannot be taken by the same vehicle of trip $j$ forms an incompatibility set ($Q_j$). There are three reasons for any two trips not to be taken by the same vehicle. These three cases are depicted in Fig. 2.

The first case occurs when a trip cannot be taken by the same vehicle. This is observed for trips 1 and 5 in Fig. 2. Based on these observations, the incompatibility set of trip $j$, namely $Q_j$ is defined as follows:

$$Q_j = \{ b \in \{1, \ldots, N\} : r_j \leq r_b \leq d_j + a_{jb} \text{ or } r_b + p_b - r_j > S + O \text{ or } r_j \leq r_b \leq d_j \}$$

2.3. Assumptions and mathematical formulation

Before presenting the mathematical formulation of the problem, we restate our assumptions briefly. These assumptions are shaped based on the organizational structure.

(i) Team trips have to be performed with a single vehicle. The demand of any team trip is smaller than or equal to the capacity of the largest vehicle.

(ii) There are enough vehicles available in each size.

(iii) The trip times are deterministic, constant, and known in advance.

(iv) No delay or cancellation is possible for trips. All trips must be carried out.

The data of the problem include:

(i) The daily program: The information about the exact timing of the activities, the facilities that the transportation will take place in between, and also the number of passengers to be transported on each trip.

(ii) The distances and times between the facilities: Since the distances of all trips are known, time intervals of the trips are known in advance. Moreover, the setup times of the trips are also known beforehand.

(iii) Capacities and costs of vehicles: The information about the vehicles is known in advance.

Our decision variables are

$$x_{ij} = \begin{cases} 1 & \text{if vehicle } i \text{ takes trip } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if vehicle } i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

$$o_i : \text{ overtime usage of vehicle } i$$

![Fig. 2. Illustration of the incompatibility set.](image)
Based on the above assumptions and definitions in the previous sections, the mathematical formulation of our problem is stated as follows:

$$\min z = \sum_{i=1}^{l_k} (f_i y_i + v_i o_i)$$  \hspace{1cm} (1)$$

subject to:

$$\sum_{i=1}^{l_k} c_{ijy} \geq D_j, \hspace{0.5cm} \forall j$$  \hspace{1cm} (2)$$

$$x_j + x_{ib} \leq 1, \hspace{0.5cm} \forall i, j, \hspace{0.5cm} \text{and } b \in Q'_j$$  \hspace{1cm} (3)$$

$$\sum_{j=1}^{N} x_{ij} \leq M y_i, \hspace{0.5cm} \forall i$$  \hspace{1cm} (4)$$

$$(r_b + p_b)x_{ib} - r_j x_{ij} \leq S + o_i + (1 - x_{ij})p, \hspace{0.5cm} \forall i, j, \hspace{0.5cm} \text{and } b \notin Q'_j$$  \hspace{1cm} (5)$$

$$\sum_{i=1}^{l_k} x_{ij} = 1, \hspace{0.5cm} j \in T$$  \hspace{1cm} (6)$$

$$x_{ij}, y_i \text{ binary}, \hspace{0.5cm} \forall i, j$$  \hspace{1cm} (7)$$

$$o_i \geq 0, \hspace{0.5cm} \forall i$$  \hspace{1cm} (8)$$

The objective function in (1) minimizes the total fixed cost (regular time) plus the total variable costs (overtime) of all vehicles that are used. Constraint set (2) ensures that the passenger demands of all trips are met. The constraints (3) handle incompatibility between any two trips. Notice that the processing and setup times for any two trips are considered in the definition of $Q'_j$, therefore the parameter $a_{ij}$ does not appear in the model. Constraint set (4) links the model’s binary variables. The value of the large number $M$ is taken as the total number of the trips. Constraint set (5) is used for determining the total overtime usage of the vehicle, if any. The value of the large number $P$ is set as equal to the total time span of a day. Constraint set (6) ensures the assignment of a single vehicle to any team trip. Constraint sets (7) and (8) enable the variables to be nonnegative and binary. Note that $M$ and $P$ are large integer numbers.

The mathematical model of the problem resembles the mathematical formulation of a TFJS problem with spread time constraints. However, some major differences make the formulation much more complex. These differences include a demand greater than unity for each trip, and the overtime usage of vehicles.

The optimal solution to the mathematical model developed in this section is attempted using GAMS 20.2 with CPLEX solver on a P4 2.4 MHz, 256 MB RAM computer. The optimal solutions could be obtained only for cases up to $J = 20$ in reasonable times. However, the daily number of trips in the real-life problem that we are inspired from can be up to 100 on a typical day. The complex structure of the model does not allow fast optimal solutions for such large-size instances. For this reason, and since the problem should be solved daily as stated in the previous sections, three heuristics and their variants are presented in the next section to come up with quick handy solutions for especially large problem instances.

3. Heuristics

The first heuristic, vehicle-based heuristic (VBH), is based on assigning consecutive trips to a single selected vehicle to consume its daily regular time limit $S$. Then, another vehicle is selected and assigned in the same manner until there are no trips to be assigned. The second heuristic, trip-based heuristic (TBP), assigns trips, which are already ordered by their ready times, to vehicles with matching capacities. In the last heuristic, group-based heuristic (GBH), trips are grouped according to their sizes, and are assigned to vehicles of corresponding capacities. Each heuristic has a variant that considers the overtime usage of vehicles: the variants assign trips to vehicles in their regular time plus overtime. Next, we present each heuristic in detail.

3.1. Vehicle-based heuristic (VBH)

In this heuristic, a daily work schedule of each vehicle is formed by assigning the vehicle to all trips that it can take within its regular time limit, $S$. For this purpose, we sort the trips in chronological order of their ready times. Taking the earliest trip in the list, we assign a new vehicle with matching capacity to the chosen trip, and immediately form the trip set that can be performed by this vehicle (based on the compatibility set). If there are any trips that can be assigned to the vehicle, we assign the best trip to the vehicle with respect to a performance criterion, and repeat the procedure until the vehicle’s spread time limit is consumed. The procedure continues with new vehicles until all trips are assigned.

In the variant heuristic, all vehicles are allowed to work overtime. In fact, the variant heuristic somehow forces the vehicles to work overtime, by letting the performance time as $S+O$. The two heuristics result in different schedules.

The parameters of the algorithm are as follows:

**Parameters:**

$F$ The set of unused vehicles.
$T_i$ The set of team trips that cannot be assigned to vehicle $i$.
$S$ The regular time limit (spread time limit).
$O$ The overtime limit.
$O^B$ The loaded overtime usage of vehicle $i$ if it takes trip $b$ after trip $j$. This is the time that the vehicle travels without passengers between the arrival time of trip $j$ and the ready time of trip $b$. Simply $O^B = r_b - d_j$. Note that this time includes setup time, as well.
$O^O$ The empty time of vehicle $i$ corresponding to its overtime usage, if it takes trip $b$ after trip $j$. This is the time that the vehicle travels between the completion of trip $j$ and the arrival of trip $b$.
$T_i$ The set of team trips that cannot be assigned to vehicle $i$.
$F$ The set of unused vehicles.
$T_i$ The set of team trips that cannot be assigned to vehicle $i$.
$S$ The regular time limit (spread time limit).
$O$ The overtime limit.
$O^B$ The loaded overtime usage of vehicle $i$ if it takes trip $b$ after trip $j$. This is the time that the vehicle travels without passengers between the arrival time of trip $j$ and the ready time of trip $b$. Simply $O^B = r_b - d_j$. Note that this time includes setup time, as well.
$O^O$ The empty time of vehicle $i$ corresponding to its overtime usage, if it takes trip $b$ after trip $j$. This is the time that the vehicle travels between the completion of trip $j$ and the arrival of trip $b$.
Based on these parameters, we define a performance criterion for assignment of vehicle \( i \) to trip \( b \) after trip \( j \):

\[
R_{ijb} = (w_{ijb} - w_{ijb}'\frac{f_i}{c_i} + w_{ijb} \ast v_i + \max(c_i - D_{ub}, 0)) \\
\ast \left( (p_b - p_{ijb}'\frac{f_j}{c_j} + p_{ijb} \ast v_j) \right); \quad \forall b \in (Q_i^j - T_j)
\]

The sum in \( R_{ijb} \) is made up of two parts. The first and second components in the summation represent the costs of empty time for vehicle \( i \) in its regular time and overtime, respectively. The third component estimates the cost of empty seats of vehicle \( i \) on regular time and overtime, if it takes trip \( b \) after trip \( j \). An \( R_{ijb} \) value is calculated for every trip \( b \) that vehicle \( i \) can take after trip \( j \). Vehicle \( i \) is assigned to trip \( b \) that gives \( \min_b(R_{ijb}) \). This value enables the vehicle to be assigned to the busiest trip with minimum delay, that is, the vehicle is assigned to the trip with the lowest free seat cost and shortest delay time this way.

The detailed description of the heuristic with overtime consideration (VBHO) is given below. We don't provide a separate description for the heuristic not considering overtime (VBH), since the one is a generalization of the other. The only difference in the algorithm will be the exclusion of Step 0 (vi), and replacement of (S+O) in Step 0 (iv) with \( S \).

**Heuristic VBHO**

**Step 0: Initialization:**

(i) Sort the vehicles by their sizes in descending order.
Put all vehicles in set \( F \).

(ii) Create a list \( L \) of trips by ordering the trips chronologically according to their ready times.

(iii) Form a set \( T_i \) for each vehicle: \( T_i : \{ j \in T : c_i < D_j \} \).

(iv) Form a \( Q_i^j \) set for each trip in list \( L \):

\[
Q_i^j = \{ b > j : d_j + a_{jb} \leq r_b \quad \text{and} \quad d_b - r_j \leq S + O \};
\]

(v) Calculate \( w_{ijb} \) values for each trip: \( w_{ijb} = r_b - d_j \), \( \forall b \in Q_i^j \).

(vi) Calculate \( w_{ijb}' \) and \( p_{ijb}' \) values for each trip (if overtime is allowed):

\[
w_{ijb}' = \max(r_b - \max(d_j, (r_{jb} + S)), 0), \quad \forall b \in Q_i^j
\]

\[
p_{ijb}' = \max(d_b - \max(r_b, (r_{jb} + S)), 0), \quad \forall b \in Q_i^j
\]

**Step 1:** Choose the first unassigned trip from list \( L \); let it be trip \( j \).

**Step 2:** Assign an unused vehicle to trip \( j \):

If \( D_j \geq \max(c_i) \), then assign vehicle \( i \) to trip \( j \), where \( i = \arg\max_{i \in F}(c_i) \).
Else, calculate \( U_i = (c_i - D_j), \forall i \in F \). Assign the vehicle \( i \) with minimum positive \( U_i \) value to trip \( j \) (assign randomly in case of ties).
Eliminate vehicle \( i \) from set \( F \). Set \( k_i \leftarrow j \).

**Step 3:** Set \( D_j = \max((D_j - c_i), 0) \). If \( D_j = 0 \), then eliminate trip \( j \) from list \( L \).

**Step 4:** Assignment of the vehicle \( i \) to its new trips within its allowed time.
If \( (Q_i^j - T_j) \neq \emptyset \) then

(i) Calculate \( R_{ijb} \) value for every trip \( b \in (Q_i^j - T_j) \).
(ii) Assign vehicle \( i \) to trip \( b \) whose \( R_{ijb} \) value is minimum.
(iii) Set \( b \leftarrow j \) (Consider the newly assigned trip \( b \) as trip \( j \)).
(iv) Go to Step 3.

Else, go to Step 5.

**Step 5:** If list \( L \) is nonempty, go to Step 1, else stop.

The objective function is calculated by adding the fixed costs of used vehicles when no overtime is allowed. In case of overtime, the total overtime usage of each vehicle is calculated and the respective overtime cost is included in the objective function. The complexity of this heuristic is \( O(n^2) \), which is mainly defined by the loop in the third and fourth steps.

### 3.2. Trip-based heuristic (TBH)

In this heuristic, the trips are sorted in chronological order of their ready times. We take the earliest trip in the list and assign enough number of vehicles to meet the demand of that trip. While doing this, the vehicles that have been already used in previous trips have a priority. The algorithm stops when all trips are assigned.

The notation and parameters are the same as in heuristic VBH, except the definition of the compatibility set. Namely, we let \( Q_i^j \) be the set of trips a vehicle is able to take during its daily usage period (S or S+O) before it is assigned to trip \( j \). This kind of definition for the compatibility set enables us to assign higher priorities for the used vehicles. The detailed flow of the algorithm is presented below. Again, we don't give a separate description for the heuristic without overtime; the only differences in the algorithm are that Step 0 (vi) is for the generalized version only (presented below), and S+O should be replaced by \( S \) in Step 0 (iv) in the heuristic without overtime consideration.

**Heuristic TBHO**

**Step 0: Initialization:**

(i) Sort the vehicles by their sizes in descending order.
Put all vehicles in set \( F \).

(ii) Create a list \( L \) of trips by ordering the trips chronologically according to their ready times.

(iii) Form a set \( T_i \) for each vehicle: \( T_i : \{ j \in T : c_i < D_j \} \).

(iv) Form a \( Q_i^j \) set for each trip in list \( L \):

\[
Q_i^j = \{ b < j : d_j + a_{jb} \leq r_j \quad \text{and} \quad d_j - r_h \leq S + O \}
\]

(v) Calculate \( w_{ijb} \) values for each trip: \( w_{ijb} = r_j - d_b \), \( \forall b \in Q_i^j \).

(vi) Calculate \( w_{ijb}' \) and \( p_{ijb}' \) values for each trip (if overtime is allowed):

\[
w_{ijb}' = \max(r_j - \max(d_b, (r_{jb} + S)), 0), \quad \forall b \in Q_i^j
\]

\[
p_{ijb}' = \max(d_j - \max(r_j, (r_{jb} + S)), 0), \quad \forall b \in Q_i^j
\]

**Step 1:** Choose the first unassigned trip from list \( L \); let it be trip \( j \).

**Step 2:** Assign an unused vehicle to trip \( j \):

If \( D_j \geq \max(c_i) \), then assign vehicle \( i \) to trip \( j \), where \( i = \arg\max_{i \in F}(c_i) \).
Else, calculate \( U_i = (c_i - D_j), \forall i \in F \). Assign the vehicle \( i \) with minimum positive \( U_i \) value to trip \( j \) (assign randomly in case of ties).
Eliminate vehicle \( i \) from set \( F \). Set \( k_i \leftarrow j \).

**Step 3:** Set \( D_j = \max((D_j - c_i), 0) \). If \( D_j = 0 \), then eliminate trip \( j \) from list \( L \).
Step 1: Choose the first unassigned trip from list $L$; let it be trip $j$.

Step 2: If there is not any other trip that can be taken before trip $j\ (Q_j^f = \emptyset)$ then go to Step 4, otherwise go to Step 3.

Step 3: If vehicle $i$ that is previously assigned to trip $b$ is available to take trip $j\ (i \neq T_i)$,

(i) Calculate $R_{bji}$ ratio as follows for every trip $b \in Q_j^f$ that is previously assigned to a vehicle, $i$:

$$R_{bji} = \frac{(w_{bji} - w'_b f_i + W_{bji} v_i + \max(c_i - D_j, 0))}{c_i} = \left[ (p_j - p'_{bji} f_i + p_{bji} v_i) \right]$$

(ii) Form an assignment list by sorting the calculated $R_{bji}$ values in ascending order.

(a) If the assignment list is empty, go to Step 4.

Else, assign trip $j$ to the first vehicle $i$ that yielded the minimum $R_{bji}$ value (the first item in the list). Delete this vehicle from the list.

(b) $D_j = \max((D_j - c_i), 0)$.

(c) If $D_j \neq 0$, then go to Step a.

Else, go to Step 6.

Step 4: Assign an unused vehicle to trip $j$:

If $D_j \geq \max(c_i)$, then assign vehicle $i$ to trip $j$, where $i = \arg\max_{i \in F} c_i$.

Else, calculate $U_i = (c_i - D_j), \forall i \in F$. Assign the vehicle $i$ with minimum positive $U_i$ value to trip $j$.

Eliminate vehicle $i$ from set $F$. Set $k_i \leftarrow j$.

Step 5: Set $D_j = \max((D_j - c_i), 0)$. If $D_j = 0$, then eliminate trip $j$ from list $L$. Else, go back to Step 4.

Step 6: Stop if list $L$ is empty, go to Step 1 otherwise.

The objective function value is calculated as in the VBH. The complexity of this heuristic is $O(n^3)$, as in VBH. The complexity is mainly defined by the loop in the third step.

3.3. Group-based heuristic (GBH)

In this heuristic the trips are divided into two sets; one containing team trips, and the other containing the remaining ones. Further, we form subsets based on demands. The idea behind this grouping is to assign vehicles primarily to the trips with demands matching to vehicle capacities. The algorithm works as follows:

**Heuristic GBH:**

Step 0. Initialization:

(i) Index the trips in chronological order of their ready times.

(ii) Index vehicles in decreasing order of their capacities.

(iii) Form two sets, $A$ and $B$. Gather all trips for which a single vehicle must be assigned (trips that cannot be split, i.e., team trips), in set $A$. Put all remaining trips in set $B$.

(iv) Group the trips in sets $A$ and $B$ separately as follows in $(2K+1)$ groups:

$$\text{Group } 1_A = \{ j \in A : c_1 \geq D_j, c_2 \}, \quad \text{Group } 1_B = \{ j \in B : c_1 \geq D_j, c_2 \}$$

$$\text{Group } 2_A = \{ j \in A : c_3 \geq D_j, c_4 \}, \quad \text{Group } 2_B = \{ j \in B : c_3 \geq D_j, c_4 \}$$

$$\ldots \quad \ldots \quad \ldots \quad \ldots$$

$$\text{Group } K_A = \{ j \in A : c_{K1} \geq D_j \}, \quad \text{Group } K_B = \{ j \in B : c_{K1} \geq D_j \}$$

$$\text{Group } K + 1_A = \{ j \in A : c_{K2} \}, \quad \text{Group } K + 1_B = \{ j \in B : c_{K2} \}$$

Note that, in the problem we have $K$ types of vehicles, having different capacities and costs. In the above definitions, $C_k$ represents the capacity for vehicle type $k$. Hence, for each vehicle type, we form two sets which contain team trips and other trips that this type of vehicle can carry without splitting. The elements in Group $K + 1_A$ are the trips that must be split.

Step 1: Scheduling of Trips in set $A$ (team trips):

Consider the nonempty groups sequentially. For each group $k_A, k = 1, \ldots, K$:

(i) Let $f$ and $l$ be the first and last trips of the group, respectively.

(ii) If $(d_l - r_f) \leq S$, this becomes a TFJS problem. To find the minimum number of the equal-capacity vehicles required to carry out the trips in this group, use Algorithm 1.

Else, assign a new vehicle of type $k$ (call this vehicle $i$) to trip $f$. The ending time of the regular shift for vehicle $i$ then becomes $(r_f + S)$. Determine the trips in Group $k_A$ that can be assigned to this vehicle during its regular shift, i.e., determine set $Q_i = \{ j \in k_A : r_j \geq d_l, d_j \leq r_f + S \}$. Do the following for the trips in $Q_i$:

- $W_i = \{ f \}$. $W_i$ represents the set of trips assigned to vehicle $i$.

- Consider the trips in $Q_i$ sequentially. At each ready time, add the arriving trip to set $W_i$. If the vehicle is not available at the ready time of a trip, remove the trip with the latest deadline from set $W_i$ (but not trip $f$).

(iii) Check if any other trip can be assigned to the vehicle(s) scheduled in Step (ii):

- Start the checking procedure with Group $K + 1_B$ (if not empty). If there exists a trip $j$ in that group that can be assigned to the vehicle in its regular shift, then split the trip in two. $D_j$ becomes $C_1$, while $D_{j+1}$ becomes $(D_j - C_1)$. Assign trip $j$ to vehicle $i$. Place trip $(j+1)$ in the appropriate group, and update the total number of trips $(j+1)$. After considering all trips in Group $K + 1_B$, proceed with groups $1_B, 2_B, 3_B, \ldots$ in this order.

- Remove all assigned trips from consideration.

Step 2: Scheduling of trips in set $B$:

Consider the nonempty groups in decreasing order of their number of elements (cardinalities). For each group $k_B, k = 1, \ldots, K$:

(i) Let $f$ and $l$ be the first and last trips of the group, respectively.

(ii) If $(d_l - r_f) \leq S$, use Algorithm 1 to assign minimum number of vehicles to the trips in this group.
Else, assign a new vehicle of type \(k\) (call this vehicle \(i\)) to trip \(f\). The ending time of the regular shift for vehicle \(i\) then become \((r_f+S)\). Determine the trips in Group \(k_B\) that can be assigned to vehicle during its regular shift, i.e., determine set \(Q_i = \{ j \in k_B : r_j < r_f, \text{ and } j \text{ is available} \}\). Do the following for the trips in \(Q_i\):

- \(W_i = \{ f \}\). \(W_i\) represents the set of trips assigned to vehicle \(i\).
- Consider the trips in \(Q_i\) sequentially. At each ready time, add the arriving trip to set \(W_i\). If the vehicle is not available at the ready time of a trip, remove the trip with the latest deadline from set \(W_i\) (but not trip \(f\)).

(iii) Check if any other trip can be assigned to the vehicle(s) scheduled in Step (ii):

- Start the checking procedure with Group \(K+1_B\) (if not empty). If there exists a trip \((j)\) in that group that can be assigned to the vehicle in its regular shift, then split the trip in two. \(D_j\) becomes \(C_j\), while \(D_{j+1}\) becomes \((D_j-C_j)\). Assign trip \(j\) to vehicle \(k\). Place trip \((j+1)\) in the appropriate group, and update the total number of trips \((J-J+1)\). After considering all trips in Group \(K+1_B\), proceed with groups \(1_B, 2_B, 3_B, \ldots\) in this order.
- Remove all assigned trips from consideration.

If overtime is allowed, i.e., in GBHO, the above algorithm is executed by replacing \(S\) with \(S+O\).

To solve the problem in Step 1 (ii) of the algorithm, we need to find the minimum number of equal-capacity vehicles to take all the trips in a specific group. This becomes a TFJS problem. Hashimoto and Stevens (1971) prove that the minimum number of machines required to carry out all jobs of an instance of the TFJS problem equals the maximum job overlap of the jobs. Their result is a direct consequence of Dilworth’s theorem on partially ordered sets, stating that in any partially ordered set, the minimum number of chains required for covering all elements is equal to the size of a maximum anti-chain (Dilworth, 1950). An \(O(n \log n)\) algorithm for determining the maximum job overlap of the jobs is described by Hashimoto and Stevens (1971) and Gupta et al. (1979).

This result can not be directly applied in our case, because in addition to overlapping trips in this problem, some trips also cannot be taken by the same vehicle due to the setup time between two trips, as explained in Section 2.2. Therefore, we propose the following \(O(n \log n)\) algorithm for solving this problem to optimality.

**Algorithm 1.**

1. Form incompatibility sets for all trips in the group:
   \[Q_j = \{ b > j : r_j < r_b \leq r_j + p_b + a_{rb} \} \quad \forall j \in k_A\]

2. Find the maximum cardinality among these sets: \(\max(Q_j)\).
3. The minimum number of equal-capacity vehicles to take all trips in group \(k_A\) is equal to \(\max(Q_j) + 1\).
4. Assign a different vehicle to each trip in the set which determined the maximum cardinality.

Algorithm 1 is used in Steps 1(ii) and 2(ii) of GBH to assign minimum number of vehicles to the trips in a group. The overall complexity of the GBH heuristic is mainly defined by the complexity of Algorithm 1. Since Algorithm 1 can be called \(O(k)\) times in the execution of the heuristic, the overall complexity becomes \(O(kn \log n)\).

### 3.4. Lower bounds

In this section, we present our lower bounds (LB), which are used to evaluate the upper bound performances. Two procedures are developed that yield seven different LB values for the problem.

**LB 1:** This lower bounding procedure groups the trips based on their demands, and finds the necessary number of vehicles according to the sum of the processing times in each group. The grouping procedure is similar to the grouping in heuristic GBH. A trip whose demand is larger than the highest capacity vehicle is divided into groups in such a way that the least number of vehicles will be necessary to carry out the trip. The LB value is computed as the sum of costs for all groups, where the cost of a group is the total processing time in that group multiplied by an average unit time cost.

The detailed procedure is given below:

(i) Index the trips in chronological order of their ready times.
(ii) Index vehicles in decreasing order of their capacities.
(iii) Group the trips into \(K\) groups as follows:

- Group 1 = \(\{ j : C_1 \geq D_j > C_2 \}\),
- Group 2 = \(\{ j : C_2 \geq D_j > C_3 \}\),
- \(\ldots\)
- Group \(K\) = \(\{ j : C_k \geq D_j \}\).

\(C_k\) represents the capacity of vehicle type \(k\).

For any trip that has \(D_j > C_1\), add \(n\) duplicates of job \(j\) to Group 1, where \(n = \lceil D_j/C_1 \rceil\) and set \(D_j - nC_1\). Then, put a duplicate of job \(j\) in the group with appropriate capacity. Hence, Group 1 may contain duplicates of the same job.

(iv) Calculate total cost: \(LB1 = \sum_{k=1}^{K} \sum_{j \in \text{Group } k} p_j w_k\)

where \(w_k = (f_k + v_k O)/(S + O)\).

The overlapping trips, the idle times between the trips and setup times are disregarded in this LB.

**LB 2:** This method yields six different LB, using the heuristic solution algorithms that were explained in the previous section. The algorithms remain the same; however vehicle capacities and costs are modified as follows: The capacity of each vehicle is set equal to the capacity of the largest vehicle in the problem instance, while the usage costs are set as the lowest costs. That is, we let:

\[c_i = \max(c_k), \quad \forall i,\]
\[f_i = \min(f_k), \quad \forall i,\]
\[v_i = \min(v_k), \quad \forall i,\]
Six heuristic solution methods presented in the previous section are executed with the above parameter modifications, yielding six different LB:

\[ \begin{align*}
&LB_{21} : \text{solution of TBH}, \\
&LB_{22} : \text{solution of TBHO}, \\
&LB_{23} : \text{solution of VBH}, \\
&LB_{24} : \text{solution of VBHO}, \\
&LB_{25} : \text{solution of GBH}, \\
&LB_{26} : \text{solution of GBHO}.
\end{align*} \]

### 4. Computational experiment

This section describes our computational experiment designed to evaluate the performance of our heuristic algorithms. Our experimental setting is similar to the one described in Fischetti et al. (1987) for the bus driver scheduling problem.

We generate random test problems starting from 20 jobs and increasing \( n \) by increments of 10 up to 100. We assume that 40\% of the trips are team trips. We consider two different types of daily demand: Uniform in range [5,30] for team trips, and [8,100] for others. There are four types of vehicles. The related data for vehicles is given in Table 1.

We assume a day consisting of two hundred time units. Two sets of ready times are generated. In the first set \((r = 1)\), ready times follow a discrete uniform distribution in the range \([0,200]\). In the second set \((r = 2)\), a peak time distribution is considered, where 30\% of the ready times are uniformly random in the range \([30,40]\), 30\% in the range \([130,140]\), and the remaining 40\% in ranges \([0,29]\), \([41,129]\), and \([141,200]\). For each set, we use two sets of uniform distributions for processing times. The duration of team trips in both sets are uniform in the range [5,40]. For other trips, we use two uniform distributions for processing times: [5,10] \((p = 1)\) and [2,20] \((p = 2)\). Setup times are determined using \( \text{U}[0, 10] \). Time limits are set as \( S = 100 \) and \( O = 50 \).

For each problem combination, we generate 10 instances. For generating the random numbers, the prime modulus multiplicative linear congruential generator (PMMLCG) code in Law and Kelton (1991, pp. 449–457) is used with default seeds; hence the instances can be regenerated easily. The optimal solution to the mathematical model developed in Section 2.3 was solved using GAMS 20.2 CPLEX solver on a P4 2.4 MHz 256 MB RAM computer. We terminate the algorithm if an optimal solution is not found in 1 h of central processing unit (CPU) time. The heuristics were coded in C programming language with DEV-C++ compiler, and run on the same computer configuration.

The optimal solutions were only possible for problem instances with \( n = 20 \) trips within the 1-h time limit. For instances having higher number of trips, it was not possible even to find a starting feasible solution within the given time limit. Therefore, the optimal solutions for the 30-trip instances were obtained without time limitation, and the average CPU time for these instances is approximately 5 h. For instances with more number of trips, it was impossible to obtain optimal solutions within reasonable times. During our pilot runs, it is also observed that the optimal solution times are insensitive to the number of types of vehicles or vehicle types.

We first investigate the performance of our upper bounds. Table 2 reports the average and maximum upper bound deviations, expressed as a percentage of optimal objective values for \( r = 1 \) and \( r = 2 \) settings and for 20 and 30 job instances. The table also presents the average deviation of the best upper bound from the optimal solution. The average best upper bound deviation for a specific parameter combination is found by determining the best upper bound value for each instance, calculating the respective deviations, and averaging the deviations over the 10 instances of the setting.

Our heuristics seem to generate quite powerful upper bounds, and no significant performance difference is observed between the heuristics. Hence, we can say that no particular heuristic outperforms another. For both ready time settings, the upper bounding procedures yield average deviations of no more than 24\% from the optimal solutions, if we consider the individual performances. All the heuristics perform robustly under the two different ready time combinations.

Another point worth noting is that there is no deterioration in the performance of our heuristics as the problem size increases. In fact, the overall average deviation of the six upper bounds drops from 23\% to 20\% as the problem size increases from 20 to 30. This is an indicator of the heuristics’ insensitivity to problem size. The average CPU time is under one second for any parameter combination; therefore the CPU times are not reported in Table 2.

As the CPU times are very small for all heuristics, it is a wise method to solve an instance with all six and take the minimum UB value as the solution. Note that this way, the deviation from the optimal objective value is around 11\% for the solved instances, which is rather promising. When the processing times are concerned with this method,

### Table 1

Vehicle data.

<table>
<thead>
<tr>
<th>Vehicle type</th>
<th>( c_i ) (person)</th>
<th>( S ) (time unit)</th>
<th>( O ) (time unit)</th>
<th>( f_i ) (money unit)</th>
<th>( v_i ) (money unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>100</td>
<td>50</td>
<td>1200</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>100</td>
<td>50</td>
<td>900</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>100</td>
<td>50</td>
<td>720</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>100</td>
<td>50</td>
<td>600</td>
<td>10</td>
</tr>
</tbody>
</table>
A low variability case reduces the efficiency. This result is due to reduced differentiation power of the upper bounding procedures when the processing time variation is low.

We use LB values for assessing the performances of our heuristics for larger problem sizes. For this purpose, we next investigate the performance of our LB using the optimally solved instances. Table 3 reports the average and maximum LB deviations, expressed as a percentage of optimal objective values as in Table 2. The average deviation of the best LB from the optimal solution is also presented.

LB1 is clearly dominated by LB2 in terms of performance. In fact, for only one of the 80 instances LB1 was better than LB2. Since the six variants of LB2 are computed using our heuristics, a similar comment apply for them as well. For finding the best LB, all instances are solved with all seven, and the maximums are taken as best LB values. The deviation from the optimal objective value is around 23%, which is stable for different ready times, processing times or number of trips.

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
</table>

Upper bound deviations.

<table>
<thead>
<tr>
<th>r</th>
<th>N</th>
<th>p</th>
<th>TBH</th>
<th>TBHO</th>
<th>VBH</th>
<th>VBHO</th>
<th>GBH</th>
<th>GBHO</th>
<th>Best UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>1</td>
<td>0.231</td>
<td>0.387</td>
<td>0.227</td>
<td>0.331</td>
<td>0.221</td>
<td>0.387</td>
<td>0.253</td>
</tr>
<tr>
<td>2</td>
<td>0.310</td>
<td>0.489</td>
<td>0.137</td>
<td>0.242</td>
<td>0.299</td>
<td>0.458</td>
<td>0.137</td>
<td>0.261</td>
<td>0.285</td>
</tr>
<tr>
<td>Avg</td>
<td>0.270</td>
<td>0.438</td>
<td>0.182</td>
<td>0.283</td>
<td>0.281</td>
<td>0.491</td>
<td>0.335</td>
<td>0.630</td>
<td>0.321</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.216</td>
<td>0.359</td>
<td>0.137</td>
<td>0.177</td>
<td>0.221</td>
<td>0.359</td>
<td>0.171</td>
<td>0.263</td>
</tr>
<tr>
<td>2</td>
<td>0.217</td>
<td>0.342</td>
<td>0.169</td>
<td>0.253</td>
<td>0.205</td>
<td>0.321</td>
<td>0.226</td>
<td>0.377</td>
<td>0.234</td>
</tr>
<tr>
<td>Avg</td>
<td>0.270</td>
<td>0.438</td>
<td>0.182</td>
<td>0.283</td>
<td>0.281</td>
<td>0.491</td>
<td>0.335</td>
<td>0.630</td>
<td>0.321</td>
</tr>
</tbody>
</table>

| Table 3 |

Lower bound deviations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
<td>Max</td>
<td>Avg</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>1</td>
<td>0.450</td>
<td>0.537</td>
<td>0.223</td>
<td>0.391</td>
<td>0.277</td>
<td>0.329</td>
<td>0.323</td>
<td>0.391</td>
</tr>
<tr>
<td>2</td>
<td>0.566</td>
<td>0.698</td>
<td>0.256</td>
<td>0.331</td>
<td>0.356</td>
<td>0.435</td>
<td>0.256</td>
<td>0.331</td>
<td>0.290</td>
<td>0.361</td>
</tr>
<tr>
<td>Avg</td>
<td>0.508</td>
<td>0.617</td>
<td>0.290</td>
<td>0.361</td>
<td>0.317</td>
<td>0.391</td>
<td>0.290</td>
<td>0.361</td>
<td>0.344</td>
<td>0.415</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.418</td>
<td>0.545</td>
<td>0.345</td>
<td>0.438</td>
<td>0.296</td>
<td>0.377</td>
<td>0.335</td>
<td>0.387</td>
<td>0.364</td>
</tr>
<tr>
<td>2</td>
<td>0.529</td>
<td>0.604</td>
<td>0.254</td>
<td>0.344</td>
<td>0.365</td>
<td>0.395</td>
<td>0.246</td>
<td>0.344</td>
<td>0.383</td>
<td>0.417</td>
</tr>
<tr>
<td>Avg</td>
<td>0.473</td>
<td>0.575</td>
<td>0.299</td>
<td>0.391</td>
<td>0.330</td>
<td>0.386</td>
<td>0.290</td>
<td>0.366</td>
<td>0.374</td>
<td>0.439</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>1</td>
<td>0.417</td>
<td>0.452</td>
<td>0.346</td>
<td>0.380</td>
<td>0.350</td>
<td>0.411</td>
<td>0.355</td>
<td>0.380</td>
</tr>
<tr>
<td>2</td>
<td>0.538</td>
<td>0.664</td>
<td>0.247</td>
<td>0.331</td>
<td>0.318</td>
<td>0.392</td>
<td>0.247</td>
<td>0.331</td>
<td>0.375</td>
<td>0.407</td>
</tr>
<tr>
<td>Avg</td>
<td>0.478</td>
<td>0.558</td>
<td>0.296</td>
<td>0.355</td>
<td>0.334</td>
<td>0.402</td>
<td>0.301</td>
<td>0.355</td>
<td>0.379</td>
<td>0.409</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.285</td>
<td>0.365</td>
<td>0.316</td>
<td>0.344</td>
<td>0.244</td>
<td>0.368</td>
<td>0.305</td>
<td>0.344</td>
<td>0.321</td>
</tr>
<tr>
<td>2</td>
<td>0.504</td>
<td>0.559</td>
<td>0.282</td>
<td>0.353</td>
<td>0.307</td>
<td>0.371</td>
<td>0.283</td>
<td>0.353</td>
<td>0.340</td>
<td>0.392</td>
</tr>
<tr>
<td>Avg</td>
<td>0.395</td>
<td>0.462</td>
<td>0.299</td>
<td>0.349</td>
<td>0.275</td>
<td>0.369</td>
<td>0.293</td>
<td>0.349</td>
<td>0.330</td>
<td>0.399</td>
</tr>
</tbody>
</table>
We may observe from the table that different heuristics perform better under different parameter combinations, hence solving the problem by all and taking the best value is an intelligent decision. For example when \( p = 1 \), VBH and TBH yield the best 79 and 65 upper bounds in 360 instances, respectively, outperforming all other heuristics. However, when \( p = 1 \), GBHO and TBHO perform better than all, and give the best results in 63 and 62 instances, respectively. The CPU times of the heuristics are ignorable for even the largest problems. The deviations are around 35% in all problem instances, and the performance of the best upper bound does not decline with growing problem size for instances with \( N = 50 \) and more. Note that the deviations are computed using LB; the deviations from the optimal solutions are expected to be much lower, as in the cases for \( N = 20 \) and 30.

### 5. Conclusions

We consider the problem of determining the optimal number of different types of vehicles to meet a given schedule of trips with varying demands. Our objective is to minimize the total (fixed and overtime) cost of the vehicles. All trips must be performed. A spread time constraint limits the total time between the start and the finish times of the operations on any vehicle. As the problem should be solved daily and the nature of the problem is too complex to obtain optimal solutions in reasonable CPU times, we propose six heuristics to obtain near optimal solutions in very small times.

We evaluate the performance of our heuristics by utilizing optimal solutions for small, and lower bounds for large problem instances. The computational results reveal that the heuristics return quality solutions for problem instances with up to 100 jobs within only seconds, and are likely to perform well for larger instances, as the performances are insensitive to problem size. It is advisable to use all heuristics for any problem instance and select the best solution, as the computation times are very small and no heuristic seems to outperform the others.

The future research may point out development of solution procedures for the more general vehicle environments, like an environment where trip times vary with vehicles. The stochastic nature of the trip times may as well be considered.

Another interesting future research topic may be considering the problem as a variable job scheduling (VJS) Problem, which will, in fact, be more realistic considering the nature of the problem. In VJS, a trip does not have to start its processing at the instant it arrives (at its ready time) but it has to be completed before a deadline. This leaves a slack time and adds flexibility to the problem.

Finally, taking passengers to different destination points on a given trip with alternative routes is another research area well worth considering. Evolutionary algorithms may be employed in solving such a complicated vehicle scheduling/routing problem.

### Acknowledgments

The authors thank the anonymous referees whose invaluable comments improved the clarity of the presentation.

---

**Table 4**

<table>
<thead>
<tr>
<th>( N )</th>
<th>( r )</th>
<th>( p )</th>
<th># of Best UB</th>
<th>Best UB</th>
<th>Best LB (%)</th>
<th>Avg.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBH</td>
<td>TBHO</td>
<td>VBH</td>
<td>VBHO</td>
<td>GBH</td>
<td>GBHO</td>
<td>Avg.</td>
<td>Max.</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References