State-of-the-Art Auction Algorithms for Multi-depot Bus Scheduling Problem Considering Depot Workload Balancing Constraints

M. Niksirat
Department of Computer Sciences, Birjand University of Technology, Birjand, Iran

ABSTRACT
This paper deals with multi-depot bus scheduling (MDBS) problem. Depot workload balancing constraints are introduced. In this case, the problem can be stated as a two-objective multi-commodity flow problem with soft constraints. Two state-of-the-art heuristics are developed including schedule-based and cluster-based heuristics, both of them extend auction algorithm. Also three different approaches are proposed to satisfy depot workload balancing constraints. The convergence of each algorithm is investigated and its complexity obtained. To illustrate the main concepts and results, a small example is solved. Also, to demonstrate the performance of the proposed algorithms, some benchmark examples are considered and CPU-time and optimality gap are compared. The results show a great improvement in the CPU time and quality of the solution of the proposed algorithms. Also the extension of the proposed algorithms under ϵ-scaling approach is analysed.

1. Introduction
Transportation scheduling is an important task to improve the quality of service in urban transportation. Transportation planning process covers a very wide operating decisions [1]. This process includes six steps [2]:

(1) Transportation network design  
(2) Frequency development  
(3) Timetable setting  
(4) Bus scheduling  
(5) Crew scheduling  
(6) Fleet maintenance scheduling.

Transit planning is performed in a sequential manner in which the outcome of one step is fed as the input to the next step [3]. This paper focuses on step 4 to assign buses to trips.
with minimum cost. This problem is addressed with MDBS problem and can be stated as a multicommodity network flow problem [4].

To solve this problem, some exact algorithms were studied, see e.g. [5–10]. Also, a lot of heuristic algorithms were proposed [11–14]. Pepin et al. [4] have presented a comparison between five different heuristic approaches for solving MDBS problem. Some researches were conducted to integrate vehicle scheduling problem with other problems in the scope of transportation system management [15–18]. An integrated approach based on $\epsilon$-constraint method is proposed in [19] for the timetabling and vehicle scheduling problem jointly. The approach was applied for transportation network with up to 50 bus lines. A large neighbourhood search meta-heuristic was proposed for vehicle scheduling and passenger service problem to modify timetable to establish a trade-off between cost of operation and quality of service [20]. A multi-objective formulation was proposed in [21] to integrate timetables, vehicle schedules and user routing problems. Researchers have considered different versions of vehicle scheduling problem. He et al. presented vehicle scheduling problem under stochastic trip times [22]. They proposed an approximate dynamic programming approach (ADP) in which a three-layer feed-forward neural network is adapted to approximate the value function. Dynamical bus vehicles scheduling under traffic congestion is considered in [23]. To generate a set of non-dominated solutions, a non-dominated sorting genetic algorithm is adopted in this paper. Also, the effects of autonomous buses to vehicle scheduling system are studied in [24]. Multi-depot vehicle scheduling problem is also raised as a subproblem in many important problems [25–27].

This paper presents some extensions of auction algorithm for MDBS problem. Auction algorithm is a primal–dual algorithm for solving network flow problems. For more description of auction algorithm, see e.g. [28, 29]. Note that, Freling et al. [30] have used auction algorithm for single depot vehicle scheduling problem. However, based on the best of our knowledge there is not any work on auction algorithm for MDBS problem.

In this paper, MDBS problem is considered with some practical constraints. These new constraints are stated to balance the workload at the depots. This modern problem is stated by a two-objective multi-commodity flow problem with soft constraints and is solved applying some improvements of auction-based heuristics. Then, the status of the proposed heuristics is derived and its performance is illustrated on some standard benchmarks.

The rest of this paper is organised as follows. In the next section, MDBS problem with workload balancing constraints is introduced and its mathematical model is stated. In Section 3, state-of-the-art heuristic algorithms are presented. Section 4 shows the results of the algorithms on benchmark examples. Final section ends the paper with a brief conclusion and future directions.

2. Multi-depot Bus Scheduling (MDBS) Problem with Depot Workload Constraints

To state the mathematical model of MDBS problem, consider a set $T$ of timetabled trips and a fleet of buses in $K$ depots. In feasible bus schedules, each trip is accomplished exactly once. The capacity of depot $k$, $k = 1, \ldots, K$ is $v_k$ buses. Each trip $i \in T$ is defined from an origin location $s_i$ to a destination location $e_i$. The start time $a_i$ and duration time $\delta_i$ are also considered to travel from $s_i$ to $e_i$. A bus schedule must be started and ended in a same depot and it is composed of a sequence of trips in which for every pair $(i, j)$ of consecutive trips,
the relation $a_i + \delta_i + t_{ij} \leq a_j$ holds. In this relation, $t_{ij}$ is the necessary time to travel from the
destination location of trip $i$ to the origin location of trip $j$. The scheduling cost of a bus in
depot $k$ is the sum of travelling cost and waiting cost between consecutive trips in the cor-
responding schedule, the cost for pulling out from the depot and the cost of pulling in the
depot. The scheduling cost also includes a fix cost of bus operating. Trips with respect to a
bus schedule are linked by deadheading trips or shortly (dh-trips) regarding to a movement
of buses without passengers. For dh-trips travel time and waiting time can be defined.

The problem has been formulated as an integer multi-commodity flow problem in which
each depot supplies a commodity[6]. In many real-life applications of the problem, the
following requirement can be given:

- Depot workload balancing constraints: To balance the number of buses pulling out from
a depot with respect to its capacity, new constraints are introduced. These restrictions
prevent traffic flow from accumulating at one or more depots and cause workforce
balancing between depots.

Assume that trips belonging to $T$ are renumbered regarding to their start time. Com-
patible trips are defined when a same bus can service trip $j$ directly after trip $i(i < j)$, i.e.
$$a_i + \delta_i + t_{ij} \leq a_j,$$
(1)
For each depot $k = 1, 2, \ldots, K$, consider a directed acyclic connection network $G^k =
(V^k, A^k)$. As depicted in Figure (1), set $V^k$ contains a node for each $i \in T$ and two nodes $o(k)$
and $d(k)$ regarding to the depot $k$, so $V^k = \{o(k), d(k)\} \cup T$. Set $A^k$ contains pull-out arcs
(o(k), i) and pull-in arcs (i, d(k)) for each trip $i \in T$ and connection arcs (i, j) with respect to
compatible trips. The cost of arc (i, j) $\in A^k$, denoted with $c_{i,j}$, equals to the sum of travel and
waiting costs. The cost $c_{o(k), i}$ also consists of a fixed cost corresponding to usage a bus. A
path from $o(k)$ to $d(k)$ in this network represents a feasible schedule for a bus in depot $k$ [4].

For each arc $(i, j) \in A^k$, let $X^k_{i,j}$ be the flow of commodity $k$ through arc $(i, j)$. Therefore
$X^k_{i,j} = 1$ if trip $j$ is accomplished immediately after trip $i$ by a bus in depot $k$. MDBS problem
with depot workload balancing constraints can be stated as follows:

$$\min \sum_{k=1}^{K} \sum_{(i,j) \in A^k} c_{i,j} X^k_{i,j},$$
$$\min \sum_{k=1}^{K} \sum_{k'=1,k' \neq k} \mu_{k,k'} X^k_{i,j},$$
(2)
$$\text{s.t.}$$
$$\sum_{k=1}^{K} \sum_{(i,j) \in A^k} X^k_{i,j} = 1, \quad \forall i \in T,$$
$$\sum_{(i,o(k), j) \in A^k} X^k_{o(k), j} \leq v_k, \quad \forall k = 1, \ldots, K,$$
$$\sum_{(i,j) \in A^k} X^k_{i,j} - \sum_{(i,j) \in A^k} X^k_{i,j} = 0, \quad \forall i \in V^k - \{o(k), d(k)\},$$
$$\forall k = 1, \ldots, K,$$
$$-\mu_{k,k'} \leq \frac{\sum_{(i,j) \in A^k} X^k_{i,j}}{v_k} - \frac{\sum_{(i,j) \in A^k} X^k_{i,j}}{v_{k'}} \leq \mu_{k,k'}, \quad \forall k, k' = 1, \ldots, K, \quad k \neq k',$$
$$X^k_{i,j} \in \{0, 1\}, \quad \forall (i,j) \in A^k, \quad k = 1, \ldots, K.$$  
(3)

The problem is modelled as a bi-objective model with soft constraints. The first objec-
tive function minimises the total cost. The positive variable $\mu_{k,k'}$ controls the deviation from
balanced status for each couple of depots. So, the second objective function minimises the sum of deviations from balance status. The objective functions are considered with a lexicographic order in which the priority of total cost is more than the sum of deviations. The first constraint of (3) ensures that each trip is assigned exactly once. The next constraint limits the number of available buses in each depot. Flow conservation constraints are explained by the third constraint in (3). The workload balancing constraints are formulated by the fourth soft constraints.

In what follows, some algorithms for solving this problem are studied.

### 3. State-of-the-Art Auction Algorithms for MDBS Problem with Depot Workload Balancing Constraints

In this section, auction algorithm is extended for MDBS problem with depot workload balancing constraints. Auction algorithm for single depot bus scheduling (SDBS) problem is reviewed from [28] and then new algorithms based on auction algorithm are proposed for MDBS problem with depot workload balancing constraints.

#### 3.1. Auction Algorithm

Auction algorithm is a famous algorithm that originally has been developed for parallel computation. This algorithm is also very fast in sequential computation. There are three forms of auction algorithms: forward, reverse and combined of forward and reverse algorithm [28].

To present auction algorithm for SDBS problem, consider the connection network \(G^{k_0} = (V^{k_0}, A^{k_0})\), which is an acyclic directed network. Let \(E = \{(i,j) \in A^{k_0} | i, j \in T\}\). A path from \(o(k_0)\) to \(d(k_0)\) in this network represents a feasible schedule for a bus and a complete
feasible bus schedule is a set of disjoint paths from \( o(k_0) \) to \( d(k_0) \) that covers each trip of \( T \).

Consider an assignment of a trip \( i \) to a trip \( j \). Trip \( i \) is forward assigned to \( j \) and trip \( j \) is backward assigned to \( i \). An assignment is feasible if each trip is forward assigned to another trip or node \( d(k_0) \), and backward assigned to another trip or node \( o(k_0) \). To implement auction algorithm, SDBS problem is considered as a maximisation problem by taking \( a_{ij} = -c_{ij} \). Assume that all \( c_{ij} \) are integer.

Let \( \pi_i \) is the profit of a forward assignment of trip \( i \) and \( p_j \) is the price of a backward assignment of trip \( j \). Also \( S_{o(k_0)} \) is the set of backward assignment to node \( o(k_0) \), \( S_{d(k_0)} \) is the set of forward assignment to node \( d(k_0) \) and \( S \) is the set of trip to trip assignment. Freling et al. [30] introduced the concept of \( \epsilon \)-complementary slackness (\( \epsilon \)-CS) as follows:

**Definition 3.1:** A feasible assignment \( S \) and a profit-price pair \((p, \pi)\) satisfy \( \epsilon \)-CS if for \( \epsilon > 0 \):

\[
\begin{align*}
\pi_i + p_j & \geq a_{ij} - \epsilon, \quad \forall (i, j) \in E, \\
\pi_i + p_j & = a_{ij}, \quad \forall (i, j) \in S, \\
\pi_i & \geq a_{i,d(k_0)}, \quad \forall i \in T, \\
\pi_i & = a_{i,d(k_0)} + \epsilon, \quad \forall i \in S_{d(k_0)}, \\
p_j & \geq a_{o(k_0),j}, \quad \forall j \in T, \\
p_j & = a_{o(k_0),j}, \quad \forall j \in S_{o(k_0)}. 
\end{align*}
\] (4)

It can be proved that if a feasible assignment \( S \) associate with profit-price pair \((p, \pi)\) satisfies \( \epsilon \)-CS condition, the assignment is an \(|T|\epsilon\)-optimal assignment. Thus to guarantee optimality of the assignment, it is sufficient to satisfy \( \epsilon \)-CS for \( \epsilon < \frac{1}{n} \) [30].

Forward auction procedure is as follows [28]. The value of a bid of trip \( i \) for another trip \( j \), which is a candidate for forward assignment is denoted by \( f_{ij} = a_{ij} - p_j \). The value of the bid for node \( d(k_0) \) is \( f_{i,d(k_0)} = a_{i,d(k_0)} + \epsilon \), with \( \epsilon > 0 \). The flowchart of the forward auction procedure is given in Figure (2).

The reverse auction procedure is similar. In reverse algorithm bid for forward assignments is replaced by bid for backward assignments. In the combined forward and reverse auction algorithms, trips are assigned forward in the forward iterations and backward in the reverse iterations. The algorithm starts with an empty assignment and an arbitrary set of prices and iteratively switches between forward and reverse algorithms until all trips assigned forward as well as backward. The combined forward and reverse auction algorithm consists of forward and backward auction iteration for forward and backward assignments respectively. In each forward and backward auction iteration, an unassigned trip bids for another trip or the depot. In the case of SDBS problem, combined algorithm is the only option because bidding needs to occur from trips to node \( d(k_0) \) (forward assignment) and from trips to node \( o(k_0) \) (backward assignment). The algorithm switches between sequences of forward and reverse auction iterations until all trips are assigned backward and forward.

Until now auction algorithm has been designed for only one depot. In this paper, some state-of-the-art algorithms are proposed to apply auction algorithm where multiple depots are available.
Figure 2. Forward auction procedure (The structure of the algorithm is based on the work of [28]).

3.2. Schedule-Based Heuristic (SBH)

The schedule-based heuristic decomposes the problem into two subproblems. In the first subproblem, a trip scheduling problem is used to minimise the transportation cost between trips. The scheduling problem can be stated as follows:

$$\min \sum_{(i,j) \in A} c_{i,j}X_{i,j}^k$$  \hspace{1cm} (5)
where \( X^k_{ij} = 1 \) if trip \( j \) is covered directly after trip \( i \), \( X^k_{ij} = 0 \) otherwise. Constraints (6) ensure that each trip is assigned to exactly one predecessor and one successor. These constraints guarantee that the trips are partitioned into a set of disjoint paths. The combined auction algorithm can be used for solving scheduling problem as follows: in forward iterations a trip bids for another trip or depots \( d(k), k = 1, \ldots, K \). Also in backward iterations a trip bids for another trip or depots \( o(k), k = 1, \ldots, K \). In this way, a set of disjoint paths from \( o(k_1) \) to \( d(k_2) \) are generated in which \( k_1, k_2 \in 1, 2, \ldots, K \). In the generated solution, a path can be assigned to different depots which is impossible. For obtaining feasibility, an assignment problem is considered to assign each disjoint path to a single depot. Let \( \Omega \) be the set of disjoint path. The cost of assigning path \( r : i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_r \) to depot \( k \), denoted with \( c^k_r \) that equals to \( c^k_r = \frac{c_{o(i_1)} + c_{d(k)}}{2} \). If workloads balancing is ignored, an auction multi-assignment algorithm can be used for assigning paths to depots (see [28] for auction multi-assignment algorithm). If workloads balancing is considered, three different approaches can be followed: cost-directed, balance-directed and greedy method. In the next subsections, these methods are described.

### 3.2.1. Schedule-Based Heuristic with Cost-directed (SBHCD) Method

In this method, the depot workload balancing constraints are relaxed. The purpose of the assignment problem is to minimize the cost instead of satisfying depot workload balancing constraints. The following model expresses the assignment problem:

\[
\begin{align*}
\min & \sum_{k=1}^{K} \sum_{r \in \Omega} c^k_r y_{k,r} \\
\min & \sum_{k=1}^{K} \sum_{k'=1}^{K} \mu_{k,k'} \\
\text{s.t.} & \sum_{k=1}^{K} y_{k,r} = 1, & \forall r \in \Omega, \\
& -\mu_{k,k'} \leq \sum_{r \in \Omega} y_{k,r} - \sum_{r \in \Omega} y_{k',r} \leq \mu_{k,k'}, & \forall k = 1, \ldots, K, \forall k' = 1, \ldots, K, k \neq k', \\
y_{k,r} \in \{0, 1\}, & \forall k = 1, \ldots, K, \forall r \in \Omega, \\
& \mu_{k,k'} \geq 0, & \forall k = 1, \ldots, K, \forall k' = 1, \ldots, K, k \neq k',
\end{align*}
\]

(8)

where \( y_{k,r} = 1 \), if path \( r \) is assigned to depot \( k \) and \( y_{k,r} = 0 \), otherwise.

The objective function of (7) is stated in a lexicographic manner in which the minimisation of assignment cost is more important than the minimisation of depot workload balancing constraint deviations. The first constraint of (8) ensures that each path must be assigned to exactly one depot. The second constraints refer to depot workload balancing. To solve this problem, a modified version of auction algorithm for multi-assignment problem is used [28]. The multi-assignment problem can be explained as follows:

\[
\min \sum_{k=1}^{K} \sum_{r \in \Omega} c^k_r y_{k,r}, \tag{9}
\]
Figure 3. Converting multiassignment problem to minimum cost flow problem (similar to [28]).

\[
\begin{align*}
\text{s.t.} & \\
\sum_{r \in \Omega} y_{k,r} & \geq 1, \quad \forall k = 1, \ldots, K, \\
\sum_{k=1}^{K} y_{k,r} & = 1, \quad \forall r \in \Omega, \\
y_{k,r} & \geq 0, \quad \forall k = 1, \ldots, K, \quad \forall r \in \Omega, \\
y_{k,r} & \geq 0, \quad \forall k = 1, \ldots, K, \quad \forall r \in \Omega,
\end{align*}
\]

(10)

The problem is characterised by the possibility of assignment of more than one path to a single depot. The problem can be converted to a minimum cost flow problem by introducing a node \( \kappa \), which is connected to each depot node \( k \) by an arc \((\kappa, k)\) of zero cost and feasible flow range \([0, \infty)\) (see Figure 3).

\[
\begin{align*}
\text{max} & \sum_{k=1}^{K} \sum_{r \in \Omega} (-c^k) y_{k,r}, \\
\text{s.t.} & \\
\sum_{r \in \Omega} y_{k,r} - y_{k,\kappa} & = 1, \quad \forall k = 1, \ldots, K, \\
\sum_{k=1}^{K} y_{k,r} & = 1, \quad \forall r \in \Omega, \\
\sum_{k=1}^{K} y_{k,\kappa} & = |\Omega| - K, \\
y_{k,r} & \geq 0, \quad \forall k = 1, \ldots, K, \quad \forall r \in \Omega, \\
y_{k,\kappa} & \geq 0, \quad \forall k = 1, \ldots, K,
\end{align*}
\]

(11)
The dual of the problem (11)–(12) is stated as follows:

\[
\min \sum_{k=1}^{K} \pi_k + \sum_{r \in \Omega} p_r + (|\Omega| - K) \lambda, \tag{13}
\]
\[
\text{s.t.}
\pi_k + p_r \geq (-c_k^r), \quad \forall \ k = 1, \ldots, K, \quad \forall \ r \in \Omega,
\lambda \geq \pi_k, \quad \forall \ k = 1, \ldots, K. \tag{14}
\]

Define a multi-assignment \( \mathcal{S} \) including a set of pair \((k, r)\) such that for each path \( r \), there is at most one pair \((k, r)\) \( \in \mathcal{S} \). A depot \( k \) with more than one pair \((k, r)\) \( \in \mathcal{S} \) is said a multi-assigned under \( \mathcal{S} \). Let \( a_k^r = -c_k^r \). Bertsekas [28] introduced \( \epsilon - CS \) for multi-assignment problems. In what follows some definitions and results are reviewed.

**Definition 3.2:** A multi-assignment \( \mathcal{S} \) and a pair \((\pi, p)\) satisfies \( \epsilon - CS \) if

\[
\pi_k + p_r \geq a_k^r - \epsilon, \quad \forall \ r \in \Omega, \quad k = 1, \ldots, K,
\pi_k + p_r = a_k^r, \quad \forall \ (k, r) \in \mathcal{S},
\]

and \( \pi_k = \max_{l=1, \ldots, K} \pi_l \) if \( k \) is multi-assigned under \( \mathcal{S} \).

The auction algorithm is modified to solve the multi-assignment problem (7)–(8). Start with a multi-assignment \( \mathcal{S} \) and pair \((\pi, p)\) satisfying \( \epsilon - CS \) (15), perform forward auction until each depot is assigned to a path. The multi-assignment \( \mathcal{S} \) satisfies \( \epsilon - CS \) (15), However some paths may not be assigned. Let \( \lambda = \max_{k=1}^{K} \pi_k \) and \( \lambda \) remains fix until the end of the algorithm. Then start with the results of the forward auction, apply a modified reverse auction algorithm as depicted in Figure 4. In Figure 4, \( v_k^l \) is the number of assigned paths to depot \( k \) in iteration \( l \).

**Proposition 3.1:** If a feasible multi-assignment \( \mathcal{S} \) with a pair \((\pi, p)\) satisfies \( \epsilon - CS \) for multi-assignment problem, then \( \mathcal{S} \) is within \( |\Omega| \epsilon \) of being optimal for multi-assignment problem. Also the triplet \((\hat{\pi}, \hat{p}, \hat{\lambda})\) is within \( |\Omega| \epsilon \) of being of an optimal solution of the corresponding dual problem, where

\[
\hat{\pi}_k = \pi_k + \epsilon, \quad \forall \ k = 1, \ldots, K,
\hat{\lambda} = \max_{k=1}^{K} \hat{\pi}_k.
\]

**Proof:** Similar to the same proof in [28], assume that \( A^* \) and \( D^* \) are the optimal value of multi-assignment problem and its corresponding dual. Assume feasible multi-assignment \( \mathcal{S} = \{(k_r, r) | r \in \Omega\} \) with pair \((\pi, p)\) satisfies \( \epsilon - CS \). For all \( r \in \Omega; \hat{\pi}_k + p_r = \pi_k + p_r + \epsilon =

Figure 4. Cost-directed algorithm (this is a modification of auction multiassignment algorithm presented in [28]).
So the proof is complete. ■

**Proposition 3.2:** The modified auction algorithm terminates with a feasible multi-assignment \( \mathcal{S} \) that satisfies \( \epsilon - CS \) for the multi-assignment problem.

**Proof:** The proof is pursued from [28]. The idea of proof is that, if \( \epsilon - CS \) conditions are satisfied at the start of an iteration, they are also satisfied at the end of the iteration. So at the start of iteration \( q \) the conditions

\[
\pi_k + p_r \geq a^k_r - \epsilon, \quad \forall r \in \Omega, \quad k = 1, \ldots, K,
\]

\[
\pi_k + p_r = a^k_r, \quad \forall (k, r) \in \mathcal{S},
\]

are true where \( \pi_k = \max_{l=1, \ldots, K} \pi_l \) and depot \( k \) is multi-assigned under \( \mathcal{S} \). Assume \((k_r, r)\) is added to \( \mathcal{S} \). So,

\[
\pi_{k_r} + p_r = (\pi_{k_r} + \delta) + (\beta_r - \delta) = \pi_{k_r} + a^{k_r}_r - \pi_{k_r} = a^{k_r}_r.
\]

Now, we consider two cases

**Case 1:** \( \delta > 0 \); in this case, assume the algorithm deletes the assignment \((k_r, r')\) form \( \mathcal{S} \). Then,

\[
\pi_{k_r} + p_{r'} = (\pi_{k_r} + \delta) + p_{r'} = (\pi_{k_r} + p_{r'}) + \delta = a^{k_r}_{r'} + \delta \geq a^{k_r}_{r'} - \epsilon.
\]

For any other \((k, r), r \in \Omega, k = 1, \ldots, K\), the values of \( \pi_k, p_r \) and \( a^k_r \) remain fixed, so the \( \epsilon - CS \) conditions are hold.

**Case 2:** \( \delta = 0 \); since in this case, the value of \( \pi_k \) and \( p_r \) remain fix, it is easily seen that \( \epsilon - CS \) conditions are true.

To show termination, note that in an iteration when a path \( r \) receives a bid from depot \( k_r, \pi_{k_r} \) is set to \( \lambda \) or is increased by at least \( \delta \). So after a finite number of iterations, \( \pi_{k_r} \) is set to \( \lambda \). After \( \pi_k \) is set to \( \lambda \), in each iteration that a path receives a bid from depot \( k \), the corresponding path is assigned to depot \( k \) without any trip that is already assigned to depot \( k \) becoming unassigned. So the algorithm terminates after a finite number of iterations. ■

When the problem data are integer, Proposition 3.3 implies that cost-directed method terminates with an optimal multi-assignment provided that \( \epsilon < \frac{1}{n} \).
Figure 5. A state-of-the-art greedy method for multi-depot vehicle scheduling problem.

3.2.2. Schedule-Based Heuristic with Greedy (SBHG) Method

In this method, a greedy approach is used for assigning paths to the depots. In this respect, a greedy parameter is defined. Sort the paths in the set $\Omega_1 = \{r_1, r_2, \ldots, r_{|\Omega_1|}\}$ in any order and kept fixed until all paths is assigned. In step $l$, the path $r_l$ should be assigned to a depot. The greedy parameter for assigning path $r_l$ to depot $k$ is defined as follows:

$$g_{r_l}^k = \frac{c_{r_l}^k}{\left(1 - \frac{v_l^k}{v_k}\right)}, \quad (17)$$

in which $v_l^1, \ldots, v_l^K$ are the number of assigned paths to depots 1, $\ldots$, $K$ respectively after step $l$. Based on the greedy parameter, we assign path $r_l$ to the depot with the least cost and least deviation from workloads balancing constraints. The greedy method is described in Figure 5.

**Proposition 3.3:** The complexity of schedule-based heuristic with greedy (SBHG) method for MDBS problem with depot workload balancing constraints is $O(|T||A^1 \cup \cdots \cup A^K|\log(|T|\omega) + |\Omega|K\log(K))$. 
Table 1. Summary of state-of-the-art algorithms for MDBS problem.

| Algorithm | Complexity |
|-----------|------------|
| SBHCD     | $O(|T||A^1 \cup \cdots \cup A^K| \log(|T|\Omega) + |\Omega||\Omega \times K| \log(|\Omega|\Omega'))$ |
| SBHBD     | $O(|T||A^1 \cup \cdots \cup A^K| \log(|T|\Omega) + |\Omega||\Omega \times K| \log(|\Omega|\Omega'))$ |
| SBHG      | $O(|T||A^1 \cup \cdots \cup A^K| \log(|T|\Omega) + |\Omega|K \log(K))$ |
| CBH       | $O(|T||T \times 1, 2, \ldots, K| \log(|T|\Omega) + K(|T||A^1 \cup \cdots \cup A_K| \log(|T|\Omega)))$ |

Table 2. Trips which be done.

| Trips | Start time | Finish time |
|-------|------------|-------------|
| 1     | 5          | 6           |
| 2     | 14         | 20          |
| 3     | 30         | 37          |
| 4     | 31         | 38          |
| 5     | 63         | 69          |
| 6     | 48         | 52          |
| 7     | 75         | 80          |

Table 3. Depots capacities.

| Depot index | 1 | 2 |
|-------------|---|---|
| Depot capacity | 4 | 6 |

Proof: In every iteration of the greedy method, the depots are sorted according to greedy parameter in $O(K \log(K))$. So the complexity of greedy method is $O(|\Omega|K \log(K))$. Now it is trivial that the complexity of the heuristic is $O(|T||A^1 \cup \cdots \cup A_K| \log(|T|\Omega) + |\Omega|K \log(K))$.

At the end of this section, the proposed state-of-the-art algorithms and their complexity are summarised in Table 1.

4. Simulation Results

In this section, several numerical examples are illustrated to demonstrate the efficiency of the proposed algorithms for solving MDBS problem with depot workload balancing constraints. Also the validation of the model is discussed. The algorithms are implemented on a computer with 6 GB RAM and 2.53 GHz CPU. In the next, the application of the proposed method for solving MDBS problem with depot workload balancing constraints in a small example and some benchmark examples are presented. Also some computation aspects of the algorithms are analysed.

At first, for illustrating the proposed algorithms, a simple example is considered. A schedule network with $K = 2$ and $T \in \{1, 2, 3, 4, 5, 6, 7\}$ is considered. Tables 2–4 show the parameter of trips, depots and travel time, respectively.

The connection networks $G^1$ and $G^2$ are shown in Figures 6 and 7. The multi-commodity model of this problem is solved with AMPL software. The objective function is 145. Two optimal schedule is:

- $o(2) \rightarrow 3 \rightarrow d(2)$,
- $o(2) \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7 \rightarrow d(2)$. 
Table 4. Travel time between each couple of nodes.

| Example | d(1) | d(2) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------|------|------|---|---|---|---|---|---|---|
| o(1)    | –    | –    | 5 | 13| 29| 30| 60| 40| 70|
| o(2)    | –    | –    | 4 | 14| 25| 30| 45| 48| 75|
| 1       | 70   | 40   | – | 5 | 21| 20| 57| 36| – |
| 2       | 60   | 75   | 5 | – | 15| 11| 38| – | 53|
| 3       | 45   | 20   | 21| 9 | – | 10| – | 10| – |
| 4       | 30   | 60   | 20| 11| 10| – | 26| 9 | 34|
| 5       | 40   | 50   | 57| 38| – | 26| – | 70| 6 |
| 6       | 90   | 30   | 36| – | 10| 9 | 10| – | 24|
| 7       | 55   | 55   | – | 53| – | 34| 6 | 24| – |

Figure 6. The connection network $G^1$ for seven trips in which $o(1)$ and $d(1)$ represent the unique depot 1.

The example is also solved with SBH and the objective function and the optimal schedules are exactly the same as AMPL results. Also the results of SBHBD algorithm are as follows. Two optimal schedule is:

$$o(2) \rightarrow 3 \rightarrow d(2),$$

$$o(1) \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7 \rightarrow d(1).$$

It is interesting to note that SBHBD algorithm assigns one of the paths to depot 1 and the other to depot 2 to satisfy depot workload balancing constraints. Also the cost of the schedules of SBHBD algorithm is 146 which is a good approximate of the optimal cost without noting to depot workload balancing constraints.

4.1. Benchmark Examples

In order to investigate the efficiency of the state-of-the-art algorithms, some benchmark examples for MDBS problem are considered. The utilised benchmark examples consist of some schedule networks with $|T| \in \{500, 1000, 1500\}$ and $K \in \{4, 8\}$. These networks which is downloadable from [31] have been used to compare different algorithms for MDBS problem. Also for each benchmark, the best solution and the minimum number of required buses are available. To compare the results with previous algorithms, Table 5 presents upper bound on the average optimality gap. The formula $100 \times (\text{Cost} –$
Figure 7. The connection network $G^2$ for the same trips depicted in Figure (6) where the schedules are started from $o(2)$ and terminated in $d(2)$.

Table 5. Upper bound on the optimality gap percentage for different heuristics.

| Heuristic     | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SBHCD         | 1.220       | 1.638       | 1.520       | 1.869       | 2.868       | 2.494       |
| SBHG          | 5.682       | 5.913       | 5.065       | 7.873       | 6.956       | 7.765       |
| Branch and cut| 0.298       | 0.171       | 0.217       | 0.684       | 0.581       | –           |
| Lagrangian heuristic | 0.170   | 0.329       | 0.405       | 0.547       | 0.672       | 0.837       |
| Column generation | 0.298  | 0.171       | 0.217       | 0.684       | 0.581       | –           |
| LNS           | 2.184       | 1.414       | 2.050       | 3.179       | 3.245       | 3.690       |
| Tabu search   | 10.919      | 8.309       | 10.779      | 17.865      | 19.647      | 20.816      |

Table 6. Average CPU time for different heuristics.

| Heuristic     | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SBH           | 25          | 191         | 570         | 74          | 148         | 469         |
| SBHG          | 5           | 22          | 57          | 7           | 19          | 59          |
| Branch and cut| 81          | 1287        | 4149        | 612         | 6207        | –           |
| Lagrangian heuristic | 85     | 700         | 2300        | 125         | 900         | 3200        |
| Column generation | 77     | 651         | 2203        | 119         | 857         | 3085        |
| LNS           | 85          | 700         | 2300        | 125         | 900         | 3200        |
| Tabu search   | 85          | 700         | 2300        | 125         | 900         | 3200        |

Best solution) / Best solution is used for computing optimality gap percentage. The results are compared with five different heuristics including branch and cut, Lagrangian heuristic, column generation, large neighbourhood search (LNS) and tabu search (see [4] for the results of these heuristic algorithms.). Also Table 6 shows the average CPU time for different algorithms. Since both CPU time and optimality gap are important for obtaining an efficient algorithm an aggregate parameter is defined as Optimality gap × CPU − time to compare different algorithms. This parameter is computed for each benchmark in Table 7. The minimum value of the aggregate parameter is underlined in Table 7. It is interesting to note that the number of buses required to perform the schedule in each benchmark is minimum for the proposed algorithms.
Table 7. Average value of aggregate parameter $Optimality\ gap \times CPU -$ time for different heuristics.

| Heuristic          | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ |
|--------------------|------------|------------|------------|------------|------------|------------|
| SBHCD              | 30.5       | 312.858    | 866.4      | 138.306    | 424.464    | 1169.686   |
| SBHG               | 29.925     | 55.616     | 204.573    | 42.546     | 205.903    | 454.182    |
| Branch and cut     | 24.138     | 220.077    | 900.333    | 412.608    | 3606.267   | –          |
| Lagrangian heuristic | 14.47     | 230.3      | 931.5      | 68.375     | 2920.5     | 11808      |
| Column generation  | 22.946     | 111.321    | 478.051    | 81.396     | 497.917    | –          |
| LNS                | 185.64     | 989.8      | 4715       | 2920.5     | 11808      | –          |
| Tabu search        | 928.115    | 5816.3     | 24791.7    | 17682.3    | 66611.2    | –          |

Table 8. The average values of parameter $var = \frac{\sum_{k=1}^{K} (\frac{v_k}{v_{k'}} - \bar{\frac{v_k}{v_{k'}}})^2}{\bar{\frac{v_k}{v_{k'}}}}$ for different heuristics.

| Heuristic          | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ | $|T| = 500$ | $|T| = 1000$ | $|T| = 1500$ |
|--------------------|------------|------------|------------|------------|------------|------------|
| SBH                | 0.0791     | 0.0520     | 0.1497     | 0.2764     | 0.2361     | 0.3305     |
| SBHCD              | 0.0791     | 0.0519     | 0.1497     | 0.2763     | 0.2360     | 0.3215     |
| SBHBD              | 0.0000     | 0.0000     | 0.0000     | 0.0001     | 0.0000     | 0.0000     |
| SBHG               | 0.0001     | 0.0001     | 0.0000     | 0.0002     | 0.0001     | 0.0001     |

The results of Tables 5 and 6 show that SBHG algorithm can be used for generating near optimal solution at a few computational time. Also, SBHCD algorithm obtains high-quality solutions at relatively few computational time. In all benchmarks, the CPU time of the proposed methods is less than five already developed methods. Also Table 5 shows that SBH, SBHCD and SBHBD algorithms obtain better quality solutions in the less computation time with respect to LNS and tabu search algorithms. All of the proposed methods outperform tabu search algorithm. The results of Table (6) show that CPU time for our proposed algorithms is decreased substantially with respect to other existing heuristics. So, the proposed methods can be used for obtaining good solutions when the system requires a solution in a few seconds.

In Section 3, different approaches for satisfying depot workload balancing constraints are proposed. In the next experiment, these three approaches are compared. For this aim, a parameter is defined as a degree of workload balancing constraint deviation for each depot as follows:

$$var = \frac{\sum_{k=1}^{K} (\frac{v_k}{v_{k'}} - \bar{\frac{v_k}{v_{k'}}})^2}{\bar{\frac{v_k}{v_{k'}}}}$$

in which $v_k^k$ is the number of assigned paths to depot $k$ and $\bar{\frac{v_k}{v_{k'}}}$ is the average value of $\frac{v_k}{v_{k'}}$, $k = 1, \ldots, K$. Table 8 shows the values of parameter $var$ for each benchmark.

Table 8 shows that the parameter $var$ has the least value for SBHBD and SBHG algorithms. So in these two algorithms, the depot workload balancing constraints deviation is the least. Note that the values of parameter $var$ in SBHCD algorithm are close to the same values in SBH algorithm. This result is expected because of the cost preference in the SBHCD algorithm.

Next experiment shows that if the cost priority is reduced in SBHCD algorithm, the depot balancing constraints deviation is also reduced. For this aim, a parameter $\Delta$ is considered...
and the set $D_r$ is defined as $D_r = \{ k' | (a_{k',r} - \pi_k') \geq (a_{k,r} - \pi_k - \Delta), \forall k = 1, \ldots, K \}$. In fact, in SBHCD algorithm the value of $\Delta$ is zero. By increasing the parameter $\Delta$, the cost priority is reduced. On the other hand, the priority of satisfying depot workload balancing constraints is increased. The results of sensitivity analysis of parameter $\Delta$ are shown in Figures 8 and 9. In these figures, the values of cost and the variance of $\{ v_k^k | k = 1, \ldots, K \}$ are depicted for different values of parameter $\Delta$. It can be understood that by increasing the parameter $\Delta$ or equivalently decreasing the cost priority, the cost of the generated solutions is increased and depot workload balancing constraints deviation is decreased.

### 4.2. Discussion

In this paper, state-of-the-art auction algorithms are proposed to solve MDBS problem under depot workload balancing constraints. Suggested algorithms are summarised in Figure 10. These algorithms are illustrated by a small example containing two depots. Also the efficiency of the proposed algorithm is compared with five heuristic algorithms on benchmark examples. The results of these comparisons showed that we can choose the solution method according to the priorities of the system administrator. For example, if cost is the main priority, cost-directed method is used, if satisfying balancing constraints are important, balance-directed methods are used, and if the CPU time is important, greedy method is applied.
Figure 9. Analysis of cost in SBHCD algorithm with respect to deviation parameter $\Delta$ for benchmarks with $K = 8$.

Moreover, in the state-of-the-art algorithms presented in Section 3, it can be seen that the running time is strongly dependent on the value of the parameter $\epsilon$. So, the practical performance of the proposed methods can be improved significantly using $\epsilon$-scaling. In this
paper, an adaptive method for $\epsilon$-scaling is applied as follows:

$$
\epsilon(k) = \begin{cases} 
1 & \text{if the number of assignments is increased by one in iteration } k \\
\alpha^{k-k_0} & \text{if the number of assignments is not increased from iteration } k_0.
\end{cases}
$$

$\epsilon(k)$ is the value of $\epsilon$ in iteration $k$ and $\alpha > 1$ is a parameter [32].

In the following, a sensitivity analysis of parameter $\alpha$ is done to investigate the effect of this parameter on the efficiency and CPU time. In this regard, a benchmark with $K = 4$ and $|T| = 500$ is considered. Computational results for the CPU time and optimality gap for different values of parameter $\alpha$ are depicted in Figure 11.

Figure 11 shows that by decreasing parameter $\alpha$, the CPU time is increased while the optimality gap is decreased. This results show that if a near optimal solution in a few computational time is requested, the large value of $\alpha$ should be selected. While solution with high quality of objective value is attained by small value of parameter $\alpha$.

5. Conclusion and Future Direction

In this paper, an extended bus scheduling problem is described by introducing depot workload balancing constraints. For this variant, a two-objective integer multi-commodity flow formulation with soft constraints is developed. Two state-of-the-art heuristic algorithms based on auction algorithm including schedule-based heuristic (SBH) and cluster-based heuristic (CBH) are proposed. Also three different approaches including cost-directed, balance-directed and greedy methods are investigated to satisfy depot workload balancing constraints in SBH. For each algorithm, termination conditions of the algorithm and its complexity are investigated. A new adaptive method for $\epsilon$-scaling in auction algorithm is also proposed and the sensitivity analysis of this method is presented. The results are compared with five existing heuristic algorithms including branch and cut, Lagrangian heuristic, column generation, large neighbourhood search and tabu search algorithms. The results show that in all benchmarks the CPU time to generate near optimal solutions for the proposed state-of-the-art algorithms are less than the same values in the existing algorithms. Also the quality of the generated solutions of the proposed algorithms is better than large neighbourhood search and tabu search algorithms. Also different approaches for satisfying depot workload balancing constraints are compared. When the cost priority is more
than the priority of depot workload balancing constraints deviation, cost-directed method is useful. While when the satisfaction of depot workload balancing constraints is important, the balance-directed and greedy methods are promising.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Notes on contributor**

*Malihe Niksirat* received her PhD degree in 2016 on Applied Mathematics and Computer Sciences from Amirkabir University of Technology, and since 2018 she has been a faculty member at the Department of Computer Sciences in Birjand University of Technology, Birjand, Iran. She was a member of the Scientific Committee of the 9th International Conference on Fuzzy Information and Engineering in 2018. Now, she is a member of Iranian Operations Research Society. Her research interests are in the areas of Fuzzy Mathematical Models and Methods, Fuzzy Arithmetic, Fuzzy Optimisation and Decision Making, Operations Research, Transportation problems, Meta-heuristic optimisation, Gray Systems, Neural Networks, Logistic and Uncertainty Analysis.

**References**

[1] Desaulniers G, Hickman M. Public transit. In: Laporte G, Barnhart C, editors. Handbooks in operations research and management science. Vol. 14, 2007. p. 69–127.

[2] Guihaire V, Hao JK. Transit network design and scheduling: a global review. Transp Res A Policy Pract. 2008;42:1251–1273.

[3] Shafahi Y, Khani A. A practical model for transfer optimization in a transit network: model formulations and solutions. Transp Res A Policy Pract. 2010;44:377–389.

[4] Pepin AS, Desaulniers G, Hertz A, et al. A comparison of five heuristics for the multiple depot vehicle scheduling problem. J Sched. 2010;12:17–30.

[5] Hadjar A, Marcotte O, Soumis F. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Oper Res. 2006;54:130–149.

[6] Kliewer N, Mellouli T, Suhl L. A time-space network based exact optimization model for multi-depot bus scheduling. Eur J Oper Res. 2006;175:1616–1627.

[7] Kulkarni S, Krishnamoorthy M, Ranade A, et al. A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem. Transp Res B Meth. 2018;118:457–487.

[8] Faiz TI, Vogiatzisb C, Noor-E-Alam M. A column generation algorithm for vehicle scheduling and routing problems. Comput Ind Eng. 2019;130:222–236.

[9] Wei M, Sun B, Jin W. A bi-level programming model for uncertain regional bus scheduling problems. J Transport Syst Eng Inf Technol. 2013;13:106–112.

[10] Guedes PC, Borenstein D. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. Comput Ind Eng. 2015;90:361–370.

[11] Laurent B, Hao JK. Iterated local search for the multiple depot vehicle scheduling problem. Comput Ind Eng. 2009;57:277–286.

[12] Wen M, Linde E, Ropke S. An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. Comput Oper Res. 2016;76:73–83.

[13] Shui X, Zuo X, Chen C. A clonal selection algorithm for urban bus vehicle scheduling. Appl Soft Comput. 2015;36:36–44.

[14] Li L, Lo HK, Xiao F. Mixed bus fleet scheduling under range and refueling constraints. Transp Res C Emer Tech. 2019;104:443–462.

[15] Kang L, Chen S, Meng Q. Bus and driver scheduling with mealtime windows for a single public bus route. Transp Res C Emer Tech. 2019;101:145–160.
[16] Boyer V, Ibarra-Rojas OJ, Ríos-Solis YA. Vehicle and crew scheduling for flexible bus transportation systems. Transp Res B Meth. 2018;112:216–229.
[17] Carosi S, Frangioni A, Galli L, et al. A matheuristic for integrated timetabling and vehicle scheduling. Transp Res B Meth. 2019;127:99–124.
[18] Chatterjee A, Mukherjee S, Kar S. A rough approximation of fuzzy soft set-based decision-making approach in supplier selection problem. Fuzzy Inf Eng. 2018;10:178–195.
[19] Ibarra-Rojas OJ, Giesen R, Rios-Solis YA. An integrated approach for timetabling and vehicle scheduling problems to analyze the trade-off between level of service and operating costs of transit networks. Transp Res B Meth. 2014;70:35–46.
[20] Petersen HL, Larsen A, Madsen OBG, et al. The simultaneous vehicle scheduling and passenger service problem. Transp Sci. 2013;47:603–616.
[21] Laporte G, Ortega FA, Pozo MA. Multi-objective integration of timetables, vehicle schedules and user routings in a transit network. Transp Res B Meth. 2017;98:94–112.
[22] He F, Yang J, Li M. Vehicle scheduling under stochastic trip times: an approximate dynamic programming approach. Transp Res Part C Emerg Technol. 2018;96:144–159.
[23] Wang C, Shi H, Zuo X. A multi-objective genetic algorithm based approach for dynamical bus vehicles scheduling under traffic congestion. Swarm Evol Comut. 2020;54:100667.
[24] Nagy V, Horváth B. The effects of autonomous buses to vehicle scheduling system. Procedia Comput Sci. 2020;170:235–240.
[25] Alizadeh Z, Nasseri SH, Mahdavi I. A genetic algorithm for supply chain configuration with new product development. Comput Ind Eng. 2017;101:440–454.
[26] Alizadeh Z, Paydar MM, Nasseri SH, et al. A meta-heuristic approach supported by NSGA-II for the design and plan of supply chain networks considering new product development. J Ind Eng Int. 2017;13:1–12.
[27] Mehrabian A, Tavakkoli-Moghaddam R, Khalili-Damaghani K. Multi-objective routing and scheduling in flexible manufacturing systems under uncertainty. Iran J Fuzzy Syst. 2017;14(2):45–77.
[28] Bertsekas DP. Linear network optimization: algorithms and codes. MIT Press: Cambridge (MA); 1991.
[29] Nasseri SH, Darvishi salokolaei D, Yazdani Cherati AB. A new approach for solving grey assignment problems. Optim Control Appl Met. 2017;2:15–28.
[30] Freling R, Wagelmans APM, Paixio JMP. Models and algorithms for single-depot vehicle scheduling. Transp Sci. 2001;35:165–180.
[31] http://people.few.eur.nl/huisman/instances.htm.
[32] Nasseri SH, Mahdavi I, Afrouzy ZA, et al. A fuzzy mathematical multi-period multi-echelon supply chain model based on extension principle. Ann Univ Craiova Math Comput Sci Ser. 2015;42:384–401.