Air Cargo Pickup and Delivery Problem with Alternative Access Airports

Farshid Azadian, Alper Murat, Ratna Babu Chinnam
Department of Industrial and System Engineering, Wayne State University, 4815 Fourth Street, Detroit, MI, 48201, f_azadian@wayne.edu, amurat@wayne.edu, r_chinnam@wayne.edu

Abstract
Continuous growth in air cargo deliveries and increasing competition is necessitating freight forwarders to develop innovative solutions to improve performance and reduce costs. This study considers a freight forwarder's operational implementation of alternative access airport policy in a multi-airport region for air cargo transportation. Given a set of heterogeneous air cargo customers, the forwarder's problem is to simultaneously select air cargo flight itineraries and schedule the pickup and delivery of customer loads to the airport(s). This problem is formulated as a novel pickup and delivery problem, where the delivery cost is both destination and time dependent. An efficient solution method based on Lagrangian decomposition and variable target method with backtracking is developed. Results of computational experiments and a practical case study in the Southern California demonstrate the merits of the model and show that the proposed algorithm is very efficient and obtains near-optimal solutions.

Key words: air cargo, pickup and delivery problem, vehicle routing, Lagrangian decomposition, multi-airport region;

1. Introduction
This paper considers a freight forwarder's problem of selecting air cargo flight itineraries to a given set of heterogeneous customers and, simultaneously, planning the pickup and airport delivery schedule of customer loads. The air cargo flight itinerary options for each customer consist of a set of flights departing from the origin airport(s) and arriving to the destination at different times. For each customer, the forwarder selects an itinerary considering flight and delivery service level related costs, such as tardiness penalties. Given the air cargo itinerary assignments, the forwarder performs the customer pickup and airport deliveries via a fleet of trucks originating from a depot. In this paper, we formulate and develop an efficient solution approach for freight forwarders to concurrently plan the air cargo flight itinerary selection and pickup and delivery scheduling of multiple customer loads to minimize the total cost of air and road transportation and service.

Over the past decade, there has been consistent growth in demand for air cargo deliveries. According to the Bureau of Transportation Statistics (BTS), in 2007, the value of air cargo shipment goods in the US is over $1.8 trillion, a 31% increase in just five years (Margreta et al., 2009). Annual forecast reports by both Airbus (2010) and Boeing (2010) predict a 5.9% annual growth rate for global air cargo tonnage over the next 20 years. In response to this growth, the air transportation network has been steadily expanding its capacity over the past two decades. However, this capacity expansion through new airports, offering more flights options, and investing in road connectivity cause the service zones of airports to expand and overlap. This has resulted in the creation of Multi-Airport Regions (MARs) where several airports accessible in
a region substitute and supplement each other in meeting the region's demand for air transportation (Loo, 2008). These MARs provide alternative access options for passengers as well as air cargo shippers and forwarders. For instance, air travelers consider MARs in a region and select airports and flights primarily based on airport access time, flight itinerary options, and frequency factors (Basar and Bhat, 2004). These factors are also important concerns for the air cargo transportation. The shippers are mainly concerned with the on-time delivery performance and the shipping costs, and thereby leave the flight itinerary decisions to forwarders. The freight forwarders, intermediaries between shippers and carriers, constitute more than 90% of air cargo shipments (Hellermann, 2006). In the case of MAR, the forwarders decide on which origin airport to use given the flight itinerary options and costs. Their decisions are primarily based on such factors as airport accessibility, proximity to the origin of the loads, flight itinerary options (e.g., frequency, destinations). Hall (2002) proposed the Alternative Access Airport Policy (AAAP) where considering multiple airports (and subsequently flight itinerary options) in a MAR can be beneficial to reduce truck mileage, decrease sorting and handling costs, improve delivery service level, and avoid congestion on both road and air network. The author discussed the merits of AAAP for air cargo transportation using the case study of the Southern California region.

In this paper, we consider a freight forwarder's operational implementation of AAAP for air cargo transportation. While Hall (2002) outlined and discussed the advantages of the AAAP, to the best of our knowledge, there is no study on its modeling and implementation. We model the forwarder's problem of selecting flight itineraries for a given set of air cargo customers, picking up their loads via a fleet of vehicles and then delivering to the airports in the region. One decision component in this problem is the flight itinerary assignment of the air cargo of different customers that are geographically dispersed in the MAR. These decisions are driven by the availability of flight itinerary options, cargo drop-off cutoff times, destination arrival times, flight itinerary costs, and tardiness penalties. The other decision component is the multi-vehicle routing to pick up customer loads and deliver to the airports prior to the starting time of the selected flight itineraries. These routing decisions are affected by the locations (depot, customers and airports), starting times of the selected flight itineraries, and the vehicle fleet size. This operational implementation of AAAP generalizes the Many-to-Many Pickup and Delivery Problems (M-M-PDP) in several aspects. For instance, the delivery cost of customer air cargo is both destination and time dependent. We hereafter refer to this problem as PDP with Assignment and Time-Dependent delivery cost (ATD-PDP). Our contribution in this research is three fold. First, we model the operational implementation of AAAP for freight forwarders which generalizes several known pickup and delivery problems in terms of model structure and objective function. Second, we develop a novel and highly efficient solution method based on the Lagrangian decomposition. Finally, we present the results of a case study implementation of the AAAP in a Southern California MAR.

The rest of this paper is organized as follows. We briefly describe the relevant literature in Section 2. In Section 3, we present the problem formulation, network transformation and preprocessing. The solution method is developed and properties such as convergence are discussed in Section 4. In Section 5, we report on the results of the computational study with experimental problem instances and a case study implementation. Section 6 concludes with discussion and future research directions.
2. Related Literature

The freight forwarder's operational implementation of the AAAP is closely related to the pickup and delivery problem. Pickup and delivery problems have been extensively studied in past decades; for a comprehensive survey see (Berbeglia et al., 2007; Berbeglia et al., 2010; Laporte, 1992, 2009; Parragh et al., 2008a, b; Toth and Vigo, 2001). Generally, the PDP involves routing a fleet of vehicles to satisfy a set of transportation requests between the given origins and destinations. In the PDP, all the origin pickups must precede the destination deliveries and be performed by the same vehicle. Moreover, each route must start and terminate at the same location (i.e., depot). The PDP usually considers capacitated vehicles and the goal is to minimize criteria related to a travel measure. The travel measure can be as simple as the total travel distance for urban commercial vehicles (Miguel Andres, 2007) or more complex as the total excess riding time over the direct ride time in passenger transportation (Diana and Dessouky, 2004). The PDP can be classified into two categories: transportation between customers and the depot, and transportation between the pickup and delivery locations (Parragh et al. 2008a). The proposed problem is in the latter category, which can be further classified into paired and unpaired pickup and delivery locations.

In the paired PDP, also known as One-to-One PDP (1-1-PDP), the load picked up from a customer location can only be delivered to one of the delivery locations. Some customers, however, may share the same delivery location. In the stacker-crane problem (SCP), unit loads of non-identical commodities have to be transported from the origin to the destination using a unit capacity vehicle (see Frederickson, 1978). In the Vehicle Routing Problem with Pickup and Delivery (VRPPD), the unit capacity requirement of SCP is relaxed and replaced with a set of constraints based on the load properties (e.g., weight, volume, or unit count). A special case of the VRPPD is the VRPPD with Time Windows (VRPPDTW) where visiting the pickup or delivery location is only allowed during a time window. While the VRPPD generally concerns goods transportation, the dial-a-ride problem (DARP) addresses the passenger transportation and therefore includes additional side constraints (e.g., maximum ride time, time windows, or service quality). Accordingly, the objective function measures customers (in)convenience; see (Cordeau and Laporte, 2007) for a comprehensive survey on the modeling and solution algorithms for DARP.

In comparison, the unpaired PDPs, also known as Many-to-Many PDP problems (M-M-PDP), consider the case where any commodity can be picked up and delivered to delivery locations that accept the commodity. The M-M-PDP was initiated with Anily and Hassin (1992) that introduced the swapping problem (SP) for moving n-commodity objects between customers with a single unit capacity vehicle. In the SP, each customer supplies one type of commodity and demands a different type. In addition to the n-commodity case of the SP, there are several other single commodity problems that are studied under the M-M-PDP where picked up loads are homogenous. Hernandez-Perez and Salazar-Gonzalez (2004a, b, 2007) introduced and studied the one-commodity pickup and delivery traveling salesman problem (1-PDTSP). The 1-PDTSP is the more general case of the Q-delivery traveling salesman problem (Q-DTSP) by Chalasani and Motwani (1999) and the capacitated traveling salesman problem with pickup and deliveries (CTSPPD) by Anily and Bramel (1999). In the 1-PDTSP, a single vehicle, starting from a depot, transports goods from pickup nodes to delivery nodes without exceeding the vehicle
capacity; the objective is to minimize the total traveling cost. Q-DTSP and CTSPD are special cases of 1-PDTSP where the pickup and deliver quantities are all one unit and the vehicle capacity is restricted (i.e., Q units). Hernandez-Perez and Salazar-Gonzalez (2009) later extend their 1-PDTSP to the Multi-Commodity One-to-One Pickup and Delivery Traveling Salesman Problem (m-PDTSP); however, with this extension, the problem is not an M-M-PDP anymore.

The proposed problem is essentially a PDP as it consists of transporting loads from customer sites (pickup locations) to the airports (delivery locations) in the MAR. The depot is both the origin and destination of the vehicles; however, it is neither pickup nor a delivery point. The proposed problem differs from the 1-1-PDP in that a customer load can be accepted by more than a single delivery location (airport). Further, it differs from the general M-M-PDP in that the delivery cost of customer loads is time and destination dependent. Moreover, the delivery cost structure is different than those proposed for PDPs. Accordingly, we denote this problem as the PDP with Assignment and Time-Dependent delivery cost (ATD-PDP). The use of term "assignment" indicates that the delivery cost of a customer's load depends on the airport and flight itinerary selected. The proposed problem's characteristics have not been studied in the literature and, to the best of our knowledge, this is the first research on PDPs with assignment and time dependent delivery costs. The proposed problem is clearly an NP-hard problem in the strong sense as it coincides with the VRPPD when there is only one airport and a single itinerary (accepted by all customers), which departs late enough to complete all pickups and delivery to the airport prior to the departure.

3. Model Formulation

In this section, we develop the model formulation of the ATD-PDP. We first discuss the time dependent delivery cost. Next, we describe the graph transformation and present the mixed integer programming model formulation. Last, we introduce and discuss pre-processing steps and valid inequalities to strengthen the formulation.

Let $G_o = (V_o, E_o)$ be an undirected graph representing the network topology of the problem where $V_o$ is the set of nodes and $E_o$ is the set of connecting edges. The set $V_o$ consists of the depot $d$, the set of customers (pickup locations) $C$, and the set of airports (delivery locations) $H$; i.e., $V_o = \{d\} \cup C \cup H$. Let $K$ be the set of uncapacitated homogeneous vehicles (trucks) that originate from the depot and operate during the depot’s opening ($\theta^o_d$) and closing hours ($\theta^c_d$). A cost $c_{ij}$ and a travel time $t_{ij}$ is associated with each edge $\forall (i,j) \in E_o$, $i \neq j$ of the network, where $c_{ij} \geq 0$ and $t_{ij} \geq 0$. We assume that the triangle property holds for the travel times and travel costs; i.e., we have $t_{ij} + t_{jg} \geq t_{ig}$ and $c_{ij} + c_{jg} \geq c_{ig}$, $\forall i,j,g \in V_o$. Note that, if needed, the fixed cost of utilizing a vehicle can be captured by adjusting the cost parameter $c_{d,j}$, $\forall j \in C \cup H$. We further assume that travel time $t_{ij}$ is deterministic and time-independent. Without loss of generality, we assume that there are no time windows for customers' pickups and the service (e.g., loading and unloading) times are negligible. The formulation can be easily extended to incorporate these considerations as the methods presented do not rely on their absence. Let $R_h$ be the set of flight itinerary options available at airport $h \in H$ on the day of operation. The cost of assigning a flight itinerary $r \in R_h$ to customer $i$ is $F_{ij}^h r$, which accounts for the flight cost of the carrier as well as the delivery service level related costs, such as tardiness penalties. The starting time
of the flight itinerary \( r \) is denoted by \( Q_r^h \) (i.e., the cargo drop-off cutoff time for the first flight of the itinerary). We only consider those flights that can be used on the day of operation, e.g., \( Q_r^h \geq \theta_d^\text{op} \).

### 3.1. Time Dependent Delivery Cost

In assigning the customer \( i \)'s cargo to a flight itinerary \( r \in R_h \), the freight forwarder accounts for the airport \( h \) arrival time. The assignment is feasible only if the airport delivery time \( t \) is on or before the flight itinerary starting time, i.e., \( t \leq Q_r^h \). When a customer's load is delivered to an airport \( h \) at time \( t \) and there are no flights available, \( t > \max_{r \in R_h} \{Q_r^h\} \), then the air cargo is assigned to a recourse flight itinerary \( r_0 \notin R_h \), e.g., a next day itinerary. We assign a penalty cost \( F_{i_0}^h > F_{i_r}^h \ \forall r \in R_h \), for airport delivery after the departure time of the last flight on the day of operation. Accordingly, we define the time dependent airport delivery cost of delivering customer \( i \)'s load to airport \( h \) at time \( t \), \( f(h,i,t) \), as follows:

\[
    f(h,i,t) = \begin{cases} 
    \min_{r \in R_h} \{F_{i_r}^h | t \leq Q_r^h\} & \text{if } t \leq \max_{r \in R_h} \{Q_r^h\} \\
    F_{i_0}^h & \text{otherwise}
    \end{cases}
\]

The definition above indicates that for each customer, not all the itinerary options need to be considered and we can identify the potential set of itinerary options that are dominated by at least another itinerary option from the same airport. The flight itineraries that are dominated for all customers are removed from further consideration. The flight itineraries that are dominated only for a subset of customers are preprocessed such that their assignment to that subset of customers is precluded. Lemma 1 provides the conditions necessary to identify the dominated flight itineraries from airport \( h \) for customer \( i \).

**Lemma 1.** Given two flight itineraries \( r, r' \in R_h \), \( r \neq r' \), itinerary \( r \) is dominated by itinerary \( r' \) if (a) \( F_{i_r}^{h} \leq F_{i_r'}^{h} \) and \( Q_r^h < Q_{r'}^h \), or (b) \( F_{i_r}^{h} < F_{i_r'}^{h} \) and \( Q_r^h \leq Q_{r'}^h \). Moreover, if (c) \( F_{i_r}^{h} = F_{i_r'}^{h} \) and \( Q_r^h = Q_{r'}^h \), considering either one is sufficient.

**Proof.** The proof is evident from the definition of \( f(h,i,t) \).

Upon the elimination of dominated itineraries, the following corollary states that there exist no two flight itineraries for customer \( i \) at airport \( h \) that either depart at the same time or have the same cost.

**Corollary 1.** After eliminating the dominated flight itineraries, there are no two flight itineraries such that \( r, r' \in R_h \), \( r \neq r' \) in \( f(h,i,t) \) with \( Q_{r'}^h = Q_r^h \) or \( F_{i_r}^{h} = F_{i_r'}^{h} \).

**Theorem 1.** Airport delivery cost function \( f(h,i,t) \) based on non-dominated flight itineraries is a non-decreasing step function with discontinuities at every \( Q_r^h \ \forall r \in R_h \).
Proof. Let us first consider single flight itinerary case where \( f(h,i,t) = F^h_{it}, \ \forall t \leq Q^h_i \) and \( F^h_{it} \) otherwise. Since \( F^h_{it} > F^h_{it'}, \) the \( f(h,i,t) \) is a step-function, which is non-decreasing and has a single discontinuity at \( Q^h_i. \) In the case of more than one flight itinerary, let us consider any two itineraries \( r, r' \in R_h. \) From Lemma 1 and Corollary 1, we have \( Q^h_i < Q^h_{i'} \) and \( F^h_{it} < F^h_{it'} \). Therefore, for any two delivery times \( t_1 \) and \( t_2 \) where \( t_1 < t_2, \) we have \( f(h,i,t_1) \leq f(h,i,t_2). \) In this case, \( f(h,i,t) \) is a non-decreasing step-function with discontinuities at \( Q^h_i \) and \( Q^h_{i'}. \) The case for more than two itineraries follows from the induction. Thus, the airport delivery cost function is a non-decreasing step-function with discontinuities at the starting times of the non-dominated itineraries.

Figure 1 illustrates a typical airport delivery cost function at airport \( h \) for two customers \( i, j \in C. \) There are two flight itinerary options available \( r = 1 \) and \( 2. \) While customer \( i \) can use both \( r = 1 \) and \( 2, \) customer \( j \) can only use the flight itinerary \( r = 1 \) and its load cannot be shipped by itinerary \( r = 2, \) e.g., destination of itinerary \( r = 2 \) is different than the customer \( j \)'s destination. Note that airport delivery after \( Q^h_2 \) for customer \( i \) \( (Q^h_1 \) for customer \( j) \) will result in the penalty cost of \( F^h_{i0} \) \( (F^h_{j0} \) for customer \( j). \)

![Illustrative airport h delivery cost function for customers i, j ∈ C; customer i has two flight itinerary options (left) and customer j has a single flight itinerary option (right).](image)

Given the time-independent and deterministic edge travel times, we can infer the following two corollaries from Theorem 1.

**Corollary 2.** Waiting at any node or delaying any airport delivery is suboptimal for ATD-PDP.

**Corollary 3.** All used vehicles start at their earliest time from the depot.

### 3.2 Graph Transformation

In ATD-PDP, each airport can be visited multiple times by a vehicle to deliver loads from different customers. Consequently, a feasible solution may not be a Hamiltonian cycle. Hence, in modeling the ATD-PDP, we need to keep track of the order of these airport visits for each vehicle by introducing additional variables. Moreover, another set of additional variables is needed to handle the step-function characteristic of the airport delivery cost. To reduce the complexity of the ATD-PDP’s formulation and eliminate the need for these additional variables, we perform a graph transformation of the original network graph \( G_o = (V_o, E_o). \)

We now describe important properties of the optimal solutions of ATD-PDP used in the
graph transformation. The first property relates to preemption, which is the act of temporarily leaving the previously picked up load at a location that is not its destination for retrieving it for delivery at a later time.

**Lemma 2.** There is an optimal solution for ATD-PDP that is non-dominated by a solution with preemption.

**Proof.** First, vehicles have no capacity restrictions to motivate preemptive solutions. In addition, a preemptive solution potentially prolongs deliveries by introducing additional node visits, shown to be suboptimal in Corollary 2. For any solution with preemption, we can identify a similar solution without preemption where the return visit for picking up the dropped load is eliminated while the remainder of the decisions remains the same. Since this elimination does not increase the airport arrival time, then, from Theorem 1, the non-preemptive solution has same or better objective function than that of the preemptive solution.

**Corollary 4.** In ATD-PDP, there is an optimal solution where all customer nodes are visited at most once.

Based on the above corollary, we can restrict the visit of each customer to at most once. The airport nodes, in contrast, can be visited more than once by each vehicle. However, the following theorem establishes that each vehicle visits an airport only once for each itinerary.

**Theorem 2.** There exists an optimal solution of ATD-PDP where each vehicle delivers customers’ load to an airport for each flight itinerary only once.

**Proof.** Consider an optimal solution in which a set of customers ($S$) are assigned to a given flight itinerary $r'$ at airport $h$. Assume that these customers are delivered to the airport by one vehicle but in two visits at times $t_1$ and $t_2$ consecutively, where $t_1 < t_2$. Clearly, $t_1 < t_2 \leq Q^h_{r'}$. Let us denote the set of delivered customers at each visit as two distinctive and non-empty sets of $S_1$ and $S_2$ respectively; i.e., $S_1 \cup S_2$. To prove the theorem it is sufficient to show that moving all the customers in set $S_1$ to set $S_2$ will results in a feasible solution with the objective value the same as the optimal objective value. First, since set $S_2$ is not empty and vehicles are uncapacitated, the proposed solution is feasible. Moreover, since the same itinerary is used the objective value is the same as the original optimal value. In other words, although the vehicle may still visit the airport at time $t_1$ for other itineraries, since $S_1$ is empty in the proposed solution, itinerary $r'$ is used only once in the second visit.

Theorem 2 states that we can restrict the solution of ATD-PDP to those solutions where each flight itinerary requires at most one visit to the airport. The following corollary establishes that we only need to consider visits to an airport $h$ equal to the number of flight itineraries $\forall r \in R_h$ plus an additional visit for the recourse flight $r_0 \notin R_h$.

**Corollary 5.** In ATD-PDP, there is an optimal solution where any airport $h$ is visited, at most, $|R_h| + 1$ times.

We use this property to perform the graph transformation. In our graph transformation
scheme, we partition each airport node $h$ into $|R_h| + 1$ nodes, each node representing a single flight itinerary. In the remainder, we refer to these nodes as *flight nodes*.

Let $G = (V, E)$ be the transformed graph of the original graph $G_0 = (V_o, E_o)$. In this transformation, each airport node $h \in H$ is replaced by $|R_h| + 1$ flight nodes, $|R_h|$ nodes each corresponding to a flight itinerary plus another node for the recourse flight. Consequently, the airport set $H$ is replaced with a new set of flight nodes $r \in R$, where $|R| = \sum_{h \in H} |R_h| + |H|$. The geographical locations of the flight nodes are identical to that of their respective airport nodes. Then, we have $V = \{d\} \cup C \cup R$. The cost of assigning flight itinerary $r \in R_h$ to customer $i$ $(F_{ir}^h)$ is replaced with the delivery cost $(F_{i0})$ to flight node $r \in R$. Note that we are using the same index $r$ for itineraries and flight nodes. Further, we introduce a hard upper time window $Q_r$ for flight node $r$, i.e., it cannot be visited after $Q_r$. The flight node for recourse flights has the delivery cost of $F_{i0}$ and upper time window of infinity.

As for the edges, we replace the airport edges $\{j, h\} \in E_o$, $\forall j \in V_o \setminus \{h\}$ with new flight node edges $\{(j, r) \in E\} \forall j \in V, \forall r \in R_h$ and assign edge travel times $t_{jr} = t_{jh}$ and costs $c_{jr} = c_{jh}$. Similar procedure is repeated for the outgoing links. In addition, a new set of links interconnecting the flight nodes are added with zero travel time and cost for the flight nodes generated from the same airport. The transformed graph $G = (V, E)$ inherits all the edges connecting depot to customers and customers to customers.

While a feasible solution in the original graph may not be a Hamiltonian cycle, the same solution is represented with one or more Hamiltonian cycles on the sub-graphs of the transformed graph. Indeed, any solution in graph $G$ can be easily transferred back to a solution in original graph $G_o$ by collapsing the flight nodes back to their original airport node.

Figure 2 illustrates the graph transformation on a network with 5 customers and 2 airports, each with 2 flights. In the feasible solution illustrated in Figure 2a, loads from customer(s) \{1\},\{2,3,4\}, and \{5\} are assigned to flight itineraries $r2$ at airport $H1$, $r3$ at airport $H2$, and $r4$ at airport $H2$, respectively. While vehicle 1's trip is a Hamiltonian cycle, vehicle 2 visits the airport $H2$ twice. In the transformed graph in Figure 2b, this solution is represented in a single Hamiltonian cycle as vehicle 1 visiting flight node $r1$ and vehicle 2 visiting flight node $r3$ and then subsequently $r4$. In Figure 2b, the shaded flight nodes correspond to flight nodes for recourse flights.

![Figure 2](image-url)  
*Figure 2.* Illustration of a sample feasible solution in the original (a) and transformed (b) graphs.
The graph transformation eliminates the need for additional variables for tracking the order of vehicle visits to airports as well as handling the step-function characteristic of the time dependent delivery cost. This transformation further reduces the complexity of the ATD-PDP’s formulation. In particular, it allows network preprocessing and introducing valid inequalities to strengthen the formulation as described in Section 3.4.

### 3.3 Formulation

The objective of the ATD-PDP is to pick up all customer loads, assign loads to flight itineraries and deliver loads to the airports on time while minimizing the total cost. We now formulate the ATD-PDP using the transformed graph as a mixed-integer programming model. Let $x^k_{ij}$ denote the binary decision variable indicating whether vehicle $k$ travels from node $i$ directly to node $j$. Let $y^k_{ir}$ be the binary decision variable indicating whether the load of customer $i$ is shipped by flight itinerary $r \in R$ with vehicle $k \in K$. The arrival time of the vehicle $k$ at node $j \in V$ is denoted as $a^k_j$. For the depot, we set $a^k_d = \theta^o_d$ for any vehicle $k \in K$. The formulation of the ATD-PDP, labeled (MP), is as follows.

\[
\text{(MP)} \quad z_{MP} = \min_{x, y} \sum_{k \in K} \left[ \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c^k_{ij} x^k_{ij} + \sum_{i \in C} \sum_{r \in R} F^r_{ir} y^k_{ir} \right] \tag{1}
\]

Subject to

\[
\sum_{k \in K} \sum_{r \in R} y^k_{ir} = 1 \quad \forall i \in C \tag{2}
\]

\[
\sum_{j \in V \setminus \{i\}} x^k_{ij} \leq 1 \quad \forall i \in V, \forall k \in K \tag{3}
\]

\[
\sum_{i \in V \setminus \{j\}} x^k_{ij} + \sum_{i \in V \setminus \{j\}} x^k_{ji} = 0 \quad \forall j \in V, \forall k \in K \tag{4}
\]

\[
(x^k_{ij} - 1)M + a^k_i + t_{ij} \leq a^k_j \quad \forall i \in V, \forall j \in V \setminus \{d, i\}, \forall k \in K \tag{5}
\]

\[
a^k_i \geq \theta^o_d \quad \forall i \in C, \forall k \in K \tag{6}
\]

\[
a^k_r \leq \min\{\theta^c_d - t_{rd}, Q_r\} \quad \forall r \in R, \forall k \in K \tag{7}
\]

\[
(y^k_{ir} - 1)M + a^k_i + t_{ir} \leq a^k_r \quad \forall i \in C, \forall r \in R, \forall k \in K \tag{8}
\]

\[
2y^k_{ir} \leq \sum_{j \in V \setminus \{i\}} x^k_{ij} + \sum_{j \in V \setminus \{r\}} x^k_{rj} \quad \forall i \in C, \forall r \in R, \forall k \in K \tag{9}
\]

\[
y^k_{ir}, x^k_{ij} \in \{0, 1\} \quad a^k_i, a^k_r \geq 0 \quad \forall i, j \in V|i \neq j, \forall r \in R, \forall k \in K \tag{10}
\]

The objective (1) minimizes the total cost of delivery including flight itineraries, service level and road travel cost. Constraint set (2) ensures that every customer’s load is assigned to a flight itinerary. Constraint set (3) guarantees that each node is visited at most once by each vehicle. Constraint set (4) is the flow conservation at each node for each vehicle. Constraint sets (5) and
(6) calculate the arrival time at every node while also preventing sub-tours. Constraint set (7) prohibits visiting a flight node after the starting time of the flight itinerary while ensuring that the vehicle can also return to the depot before the depot’s closing time. Constraint set (8) guarantees that a customer load pickup precedes its delivery to the selected flight node. Constraint set (9) ensures that both pickup and delivery of a customer load is performed by a same vehicle. Constant $M$ is a big number corresponding to arrival times and can be calculated as summation of all the links' travel times.

For brevity, let $J_k(x,y)$ denote the objective function for vehicle $k$ and $J(x,y)$ denote objective for all vehicles.

$$J_k(x,y) = \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} x_{ij}^k + \sum_{i \in C} \sum_{r \in R} F_{ir} y_{ir}^k \quad \forall k \in K$$  \hspace{1cm} (11)

$$J(x,y) = \sum_{k \in K} J_k(x,y)$$  \hspace{1cm} (12)

### 3.4 Network Preprocessing and Valid Inequalities

We strengthen the formulation (MP) by network preprocessing and introducing valid inequalities. First preprocessing step is to tighten the upper and lower bounds on the node arrival times. For a customer node, the earliest arrival time ($a_i^k$) is attained via a direct travel from the depot,

$$a_i^k \geq a_i^k = \theta_d + t_{dl} \quad \forall i \in C, \quad \forall k \in K,$$  \hspace{1cm} (13)

and, the latest arrival time ($\overline{a}_i^k$) is the latest time that allows a vehicle to pick up the customer load, deliver it to a flight node, and return to the depot before the closing time,

$$a_i^k \leq \overline{a}_i^k = \max_{r \in R} \left\{ \min_{c \in C} \left( Q_r, \theta_d + t_{rd} - t_{ir} \right) \right\} \quad \forall i \in C, \quad \forall k \in K.$$  \hspace{1cm} (14)

For a flight node, the earliest arrival time ($a_r^k$) is attained by the shortest travel from the depot after visiting a customer,

$$a_r^k \geq a_r^k = \theta_d + \min_{r \in C} \left( t_{dl} + t_{ir} \right) \quad \forall r \in R, \quad \forall k \in K.$$  \hspace{1cm} (15)

The latest arrival time to a flight node ($\overline{a}_r^k$) is already included in (MP) as the constraint set (7). Next preprocessing step is determining the lowest value for $M$ in constraints (5) and (8). In particular, we replace $M$ with edge specific $M_{ij}$,

$$M_{ij} = \overline{a}_i^k + t_{ij} \quad \forall i \in V, \quad \forall j \in V \setminus \{d, i\}, \quad \forall k \in K.$$  \hspace{1cm} (16)

The final preprocessing step is the elimination of the inadmissible edges that can never be traversed in a feasible solution. We remove edges $(i, d) \forall i \in C$ since vehicles must return to the depot empty. The edges $(d, r) \forall r \in R$ are eliminated since vehicles leave the depot empty. Lastly, we remove any edge $(i, j) \forall i, j \in V$ if $a_i^k + t_{ij} > \overline{a}_j^k$, i.e., the vehicle cannot traverse an edge if it cannot arrive its destination before the latest allowed arrival time.

We tighten the constraints set (5) by using the following lifting scheme from Desrochers and Laporte (1991) by taking the reverse arcs into account.

$$x_{ij}^k (M - t_{ij} - t_{ji}) + (x_{ij}^k - 1) M + a_i^k + t_{ij} \leq a_j^k \quad \forall i, j \neq i \in V \setminus \{d\}, \forall k \in K$$  \hspace{1cm} (17)

In addition, we introduce the following cut set that ensures that a vehicle visits a customer...
only if it delivers the customer’s load to a flight node.

\[
\sum_{j \in V \setminus \{i\}} x_{ij}^k = \sum_{r \in R} y_{ir}^k \quad \forall i \in C, \quad \forall k \in K
\] (18)

4 Methodology

First, we briefly present the standard Lagrangian Decomposition approach. Next, we introduce the Successive Subproblem Solution (SSS) method for solving ATD-PDP problem using the (MP) formulation with preprocessing and valid inequalities described in Section 3.4. We also provide convergence results and a method to estimate the bound used in subgradient optimization to improve the convergence to quality primal feasible solutions.

4.1 Standard Lagrangian Decomposition Approach

The standard Lagrangian Decomposition (LD) approach is commonly used for formulations composed of two or more intertwined subproblems that are easier to solve independently through specialized algorithms. In fact, the LD approach is commonly used for the vehicle routing problems (Kohl and Madsen, 1997). The (MP) formulation is a candidate for LD approach since constraints (2) are the only coupling constraints for vehicles and the rest of the constraints and the objective is separable by vehicle. Hence, by relaxing the constraints (2) through Lagrangian relaxation, (MP) can be decomposed to \(|K|\) subproblems, each corresponding to a single vehicle.

The Lagrangian relaxation of MP with respect to constraints (2) results in the following relaxed problem (LR),

\[
\Phi(\lambda) = \min_{(x,y) \in \Omega} \left[ \sum_{k \in K} J_k(x,y) + \sum_{i \in C} \lambda_i \left(1 - \sum_{k \in K} \sum_{r \in R} y_{ir}^k\right) \right],
\] (19)

where \(\lambda = (\lambda_1, \ldots, \lambda_{|C|}) \in \mathbb{R}^{|C|}\) is the vector of Lagrangian multipliers associated with constraints (2). The set denotes all feasible solutions of the (LR). Then, the Lagrangian Dual (LD) problem maximizes the (LR) solution, which is a lower bound on \(z_{MP}^\star\).

\[
\Phi_{LD}^\star = \max_\lambda (\Phi(\lambda)).
\] (20)

The set splits into \(|K|\) disjoint subsets, i.e. \(\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_{|K|}\), where each \(\Omega_k\) is defined by constraints (3)–(10) for a given \(k \in K\). Further, the objective of (LD) is additive, thus leading to the following decomposition,

\[
\Phi(\lambda) = \sum_{k \in K} \Phi_k(\lambda) + \sum_{i \in C} \lambda_i.
\] (21)

where \(\Phi_k(\lambda) = \min_{(x,y) \in \Omega_k} L_k(\lambda, x, y)\) and \(L_k(\lambda, x, y) = J_k(x,y) - \sum_{i \in C} \sum_{r \in R} \lambda_i y_{ir}^k\) is the Lagrangian function of the \(k^{th}\) subproblem. To solve (LD), we solve the primal subproblem \(\Phi_k(\lambda)\) for each vehicle \(k\) at the low-level and update the Lagrangian multipliers at the high-level, e.g., using subgradient optimization (Conejo et al., 2006; Fisher, 2004; Geoffrion, 1974). The optimization at both levels is performed iteratively until the dual solution converges.

However, since the vehicles are homogeneous, the subproblems \(\Phi_k(\lambda)\) are identical; i.e. \(\Omega_1 = \Omega_2 = \cdots = \Omega_{|K|}\). Hence, all subproblems have the same optimal solution with identical objective value. Accordingly, solving (LD) is equivalent to solving the following,
\[
|K| \max_{\lambda} \left( \min_{(x,y) \in \Omega_k} L_k(\lambda, x, y) \right) + \sum_{i \in C} \lambda_i
\]  

(22)

where \( k \in K \) is any one of the subproblems. This case of identical subproblems presents challenges in the solution process. In particular with discrete decisions, it leads to oscillating dual solutions, affecting the convergence rate. Further, the solutions converged are primal infeasible and provide a lower bound on \( z^*_{MP} \) that can be weak. Lastly, the primal infeasibility of the solutions requires integration with an exact (heuristic) method such as branch-and-bound (Lagrangian heuristic) to obtain optimal (good quality) solutions (Kohl and Madsen, 1997).

### 4.2 Successive Subproblem Solving Method

We adapt the Successive Subproblem Solving (SSS) method to avoid the challenges associated with the standard Lagrangian Decomposition method due to the identical subproblems. This approach is introduced by Zhai et al. (2002) to solve the unit commitment problem in electrical power generator scheduling. The SSS approach extends and improves over the standard Lagrangian Decomposition method by addressing the dual solution oscillation. However, it does not guarantee either the primal feasibility or the quality of feasible solutions. We address these issues in Section 4.4 by developing a modified variable target value method for subgradient optimization for SSS approach.

In SSS, we introduce an absolute penalty term that helps to reduce the oscillation and constraint violations more rapidly. Accordingly, the Lagrangian function is revised to the following augmented form,

\[
\tilde{L}(\omega, \lambda, x, y) = \sum_{k \in K} f_k(x, y) + \sum_{i \in C} \lambda_i \left( 1 - \sum_{k \in K} \sum_{r \in R} y^k_{ir} \right) \\
+ \omega \sum_{i \in C} \left| 1 - \sum_{k \in K} \sum_{r \in R} y^k_{ir} \right|, 
\]  

(23)

where \( \omega > 0 \) is the penalty parameter. The revised dual problem (PS) and dual function \( \Phi(\lambda) \) are then expressed as,

\[
(PS) \quad \Phi^*_PS(\omega) = \max_{\lambda} \tilde{\Phi}(\omega, \lambda) = \max_{\lambda} \left( \min_{(x,y) \in \Omega} \tilde{L}(\omega, \lambda, x, y) \right). 
\]  

(24)

The \( \Phi^*_PS(\omega) \) is the optimum dual solution with penalty weight \( \omega \). The optimum solution (PS) can be either a feasible or infeasible solution to the original problem (MP). If the solution is feasible, it can be shown that it is also optimum, i.e., no duality gap \( \Phi^*_PS(\omega) = z^*_MP \). Following theorem establishes that the \( \Phi^*_PS(\omega) \) is a lower bound on the primal optimum solution \( z^*_MP \).

**Theorem 3.** For any \( \omega \) and \( \lambda^* \), \( \tilde{\Phi}(\omega, \lambda) \leq \Phi^*_PS(\omega) \leq z^*_MP \leq f(x, y) \).

**Proof.** By definition from (1) and (24), we have \( z^*_MP \leq f(x, y) \) and \( \tilde{\Phi}(\omega, \lambda) \leq \Phi^*_PS(\omega) \), respectively. Let \( (x^*, y^*) \) be the primal optimum solution to problem (MP) and \( \lambda^* \) denote the optimum multipliers. The primal optimum solution is feasible, thus, satisfies constraint set (2). Accordingly, we have

\[
z^*_MP = \sum_{k \in K} f_k(x^*, y^*) + \sum_{i \in C} \lambda^*_i \left( 1 - \sum_{k \in K} \sum_{r \in R} y^k_{ir} \right) + \omega \sum_{i \in C} \left| 1 - \sum_{k \in K} \sum_{r \in R} y^k_{ir} \right| = \tilde{L}(\omega, \lambda^*, x^*, y^*).
\]
From the definition (24),
$$
\Phi_{PS}^*(\omega) = \min_{(x,y) \in \Omega} \tilde{L}(\omega, \lambda^*, x, y) \leq \tilde{L}(\omega, \lambda^*, x^*, y^*) = z_{MP}^* .
$$

Revised Lagrangian function (23) cannot be decomposed into \( k \) subproblems due to the penalty term. Hence, to calculate the subgradient of \( \tilde{\Phi}(\omega, \lambda) \) with respect to \( \lambda \), we now need to solve the integrated low-level problem \( \min_{(x,y) \in \Omega} \tilde{L}(\omega, \lambda, x, y) \), which is computationally inefficient. Revised Lagrangian function in (23), however, can be reformulated as an additive function. Let us redefine the Lagrangian function for \( k^{th} \) vehicle as follows:
$$
\tilde{L}_k(\omega, \lambda, x, y) = J_k(x, y) - \sum_{i \in C} \sum_{r \in R} \lambda_i y_{ir}^k + \omega \sum_{i \in C} | q_k(i) - \sum_{r \in R} y_{ir}^k | ,
$$
where,
$$
q_k(i) = 1 - \sum_{s \in K \setminus k} \sum_{r \in R} y_{ir}^s .
$$

It can be verified that the Lagrangian function (23) can be expressed in terms of \( \tilde{L}_k(\omega, \lambda, x, y) \) and \( q_k(i) \) as follows:
$$
\tilde{L}(\omega, \lambda, x, y) = \tilde{L}_k(\omega, \lambda, x, y) + \sum_{s \in K \setminus k} J_s(x, y) + \sum_{i \in C} \lambda_i q_k(i) .
$$

Since (27) is additive, we can now solve the (PS) in parts, e.g., for each vehicle. The subproblem for vehicle \( k \) is then defined as follows:
$$
(PS_k) \quad \tilde{\Phi}_k(\lambda) = \min_{(x,y) \in \Omega_k} \tilde{L}_k(\omega, \lambda, x, y) .
$$

The variable \( q_k(i) \) is fixed for the \( k^{th} \) subproblem. The \( q_k(i) \) links subproblem \( k \) to other subproblems by conveying the information about customer \( i \)'s assignment to other vehicles. Hence, the solutions of the subproblems are likely to be different from each other, thus alleviating the issues associated with identical subproblems.

In solving (PS), the SSS method solves the vehicle subproblems one at a time, while calculating the \( q_k(i) \) \( \forall i \in C \) using the solution from other vehicles. The SSS method updates the Lagrangian multipliers after solving any of the subproblems. Note that this is needed since solving subproblems one after another using the same multipliers improves the \( \tilde{L}(\omega, \lambda, x, y) \) at a decreasing rate because the subgradient directions are not being updated. In SSS, the Lagrangian multipliers are updated using the surrogate subgradient (SSG) approach introduced by Zhao et al. (1999). The standard subgradient approach requires solving all subproblems to obtain the subgradient direction (Geoffrion, 1974; Fisher, 2004). In the SSG approach, however, the solution to only one of the subproblems is sufficient to obtain a proper surrogate subgradient direction. Let \( g_i^j \) denote the surrogate subgradient for customer \( i \) at any iteration \( j \) and is calculated as,
$$
g_i^j = 1 - \sum_{k \in K} \sum_{r \in R} (y_{ir}^k)^j \quad \forall i \in C .
$$

We first introduce the notation used in the SSS method and then present its algorithmic steps.

**Notation:**
- \( (x^j, y^j)_k \): solution of \( k^{th} \) subproblem at iteration \( j \)
- \( (x^j, y^j) \): solution at iteration \( j \)
\( \hat{\lambda}^0 \): initial Lagrangian multipliers, i.e., \( \hat{\lambda}^0 = \{ \hat{\lambda}_i^0, \forall i \in C \} \)

\( \lambda^j \): Lagrangian multipliers at iteration \( j \), i.e., \( \lambda^j = \{ \lambda_i^j, \forall i \in C \} \)

\( g^j \): surrogate subgradients at iteration \( j \), i.e., \( g^j = \{ \sum_{i \in C} g_i^j, \forall i \in C \} \)

\( \delta^j \): step-size at iteration \( j \)

\( L^j_\omega \): Lagrangian function value at iteration \( j \) with penalty \( \omega \), i.e., \( L^j_\omega = \hat{L}(\omega, \lambda^j, x^j, y^j) \)

\( \beta \): step-size update parameter, \( 0 < \beta < 1 \)

\( \alpha \): initialization factor for Lagrangian multipliers

\( \varepsilon \): threshold for Lagrangian multiplier convergence criteria, \( \varepsilon > 0 \)

**SSS Procedure:**

**Initialization.**

1.1. Given \( \hat{\lambda}^0 \), e.g., \( \hat{\lambda}^0 = 0 \), solve (LD) using (22) and obtain \( (x^0, y^0) \)

1.2. Calculate,

\[
\hat{\lambda}_i^0 = \alpha \left( 1 - \sum_{k \in K} \sum_{r \in R} (y_{ir}^k)^0 \right) \quad \forall i \in C,
\]

where, \( 0 < \alpha < \left( \Phi_{PS}^*(\omega) - \hat{L}(\omega, 0, x^0, y^0) \right) / \sum_{i \in C} \left\| 1 - \sum_{k \in K} \sum_{r \in R} (y_{ir}^k)^0 \right\|^2 \)

1.3. Calculate \( L^0_\omega = \hat{L}(\omega, \lambda^0, x^0, y^0) \) and update Lagrangian multipliers:

\[
\hat{\lambda}^1 = \lambda^0 + \delta^0 g^0,
\]

where \( 0 < \delta^0 = \beta (\Phi_{PS}^*(\omega) - L^0_\omega) / \| g^0 \|^2 \) and \( 0 < \beta < 1 \). Set \( j = 1 \).

**Step 1. Subproblem Solution:**

1.1. For \( k = 1, 2, \ldots, |K| \), Repeat:

1.1.a. Solve subproblem (PS\(_k\)) in (28) by setting

\( (x^j, y^j)_s = (x^{j-1}, y^{j-1})_s \) for \( s \in K \) \( s \neq k \)

to obtain \( (x^j, y^j)_k \).

1.1.b. If the following improvement condition is satisfied,

\[
L^j_\omega = \hat{L}(\omega, \lambda^j, x^j, y^j) < \hat{L}(\omega, \lambda^j, x^{j-1}, y^{j-1}),
\]

where \( (x^j, y^j) = (x^j, y^j)_k \cup \{(x^{j-1}, y^{j-1})_s | s \in K, s \neq k \} \),

then go to Step 2, otherwise continue with the next \( k \).

1.2. Set \( (x^j, y^j) = (x^{j-1}, y^{j-1}) \).

**Step 2. Subgradient Optimization:**

2.1. Update Lagrangian multipliers:

\[
\lambda^{j+1} = \lambda^j + \delta^j g^j,
\]

where \( 0 < \delta^j = \beta (\Phi_{PS}^* - L^j_\omega) / \| g^j \|^2 \) and \( 0 < \beta < 1 \).

**Step 3. Check the stopping criteria.**

3.1. If \( \| \lambda^{j+1} - \lambda^j \| \leq \varepsilon \), then go to Step 4; otherwise set \( j = j + 1 \) and return to Step 1.

**Step 4. Terminate with solution** \( (x^j, y^j) \).
The SSS method is initialized by solving (LD) to obtain initial solutions to estimate the starting values for Lagrangian multipliers. The bounding of $\alpha$ in the initialization ensures that $L_\omega^0 = \hat{L}(\omega, \lambda^0, x^0, y^0) < \Phi_{PS}^*(\omega)$. This inequality is important for convergence analysis as explained in the next section. The subproblems in Step 1 are sequentially solved until the improvement condition in (31) is attained. In each subproblem solution, the previous iteration's solutions are used to calculate $q_k(i) \forall i \in C$. When none of the vehicle $k$'s subproblem solution satisfies (31), then the previous iteration's solution is maintained. The multipliers are updated using the surrogate gradient in Step 2.1. The SSS method terminates when multipliers converge.

The SSS method requires $\Phi_{PS}^*(\omega)$. This value, however, is generally unknown in advance and needs to be estimated. A poor underestimation may result in convergence to a primal infeasible solution with large duality gap (see Theorem 4). In the standard Lagrangian method, the value used in place of $\Phi_{PS}^*(\omega)$ is an overestimation of $z_{MP}$, which affects the convergence rate. However, the solutions converged are either primal infeasible or optimal (Held et al., 1974). In comparison, SSS method, using an overestimation of $\Phi_{PS}^*(\omega)$, may converge to a primal feasible but not optimal solution. Hence, SSS differs from the standard Lagrangian method, as it may converge to a suboptimal primal feasible solution without a feasibility recovery heuristic. The reason for this is that the SSS minimizes the augmented Lagrangian relaxation in (25) by solving decomposed subproblems in Step 1. The bound estimate of $\Phi_{PS}^*(\omega)$ in SSS is therefore critical affecting both the convergence rate and the solutions converged, i.e., primal feasible or infeasible. We present the bound estimation procedure in Section 4.2.2.

### 4.2.1 Convergence Analysis

In this section, we provide convergence results for SSS method with subgradient optimization using $\Phi_{PS}^*(\omega)$. The following theorem establishes that the Lagrangian function value at each iteration of SSS underestimates the optimal solution to the (PS).

**Theorem 4. (Solution Bounding)** For a given $\omega$, at each iteration $i$, $L_{\omega}^i < \Phi_{PS}^*(\omega)$.

**Proof.** For $j = 0$, the condition on $\alpha$ suffices. In the case of $j \geq 1$, from (31) we have,

$$L_{\omega}^i = \hat{L}(\omega, \lambda^j, x^j, y^j) \leq \hat{L}(\omega, \lambda^j, x^{j-1}, y^{j-1}).$$

Further,

$$L(\omega, \lambda^j, x^{j-1}, y^{j-1}) = L(\omega, \lambda^{j-1}, x^{j-1}, y^{j-1}) + L(\omega, \lambda^j, x^{j-1}, y^{j-1}) - L(\omega, \lambda^{j-1}, x^{j-1}, y^{j-1})
= L_{\omega}^{j-1} + \sum_{i \in C} (\lambda_i^j - \lambda_i^{j-1})(1 - \sum_{k \in K} \sum_{r \in R} y_{kr}^j) = L_{\omega}^{j-1} + \delta^j \|g^{j-1}\|^2.$$

From the definition of $\delta^j$ in Step 3 of SSS procedure we have, $L_{\omega}^j \leq L_{\omega}^{j-1} + \beta(\Phi_{PS}^*(\omega) - L_{\omega}^{j-1})$. Since $\beta < 1$, we obtain, $L_{\omega}^j < L_{\omega}^{j-1} + \Phi_{PS}^*(\omega) - L_{\omega}^{j-1} \leq \Phi_{PS}^*(\omega)$. □

The following lemma states that the search direction of the Lagrangian multipliers in any iteration is always a proper direction, i.e., $(\lambda^* - \lambda^j) g^j > 0$.

**Lemma 3. (Direction).** Let $\lambda^*$ be the optimal multiplier vector, then $\Phi_{PS}^*(\omega) - L_{\omega}^j \leq (\lambda^* - \lambda^j) g^j, \forall j$. 
Proof. Based on (23) and (24), we have
\[
\Phi^*_\text{PS}(\omega) = \Phi(\omega, \lambda^*) = \hat{L}(\omega, \lambda^*, x^*, y^*) \leq \hat{L}(\omega, \lambda^*, x^j, y^j) = L^j_{\omega} + \hat{L}(\omega, \lambda^*, x^j, y^j) = L^j_{\omega} + (\lambda^* - \lambda^j)g^j.
\]
Last step follows from the definition of \(g^j\) in (29) and Lagrangian function \(\hat{L}(\omega, \lambda, x, y)\) in (23). From Theorem 4, we have \(\Phi^*_\text{PS}(\omega) - L^j_{\omega} > 0\), thus the theorem's result follows.■

The convergence of the Lagrangian multipliers is established by the following theorem.

Theorem 5. (Convergence) In the SSS algorithm, the Lagrangian multipliers are converging; i.e,
\[
\|\lambda^* - \lambda^{j+1}\|^2 < \|\lambda^* - \lambda^j\|^2 \quad \forall j,
\]
where \(\lambda^*\) is the optimal multiplier vector.

Proof. From (32) we have
\[
\|\lambda^* - \lambda^{j+1}\|^2 = \|\lambda^* - \lambda^j - \delta^j g^j\|^2 = \|\lambda^* - \lambda^j\|^2 + (\delta^j)^2\|g^j\|^2 - 2\delta^j(\lambda^* - \lambda^j)g^j.
\]
Using result from Lemma 3, we have,
\[
\|\lambda^* - \lambda^{j+1}\|^2 \leq \|\lambda^* - \lambda^j\|^2 + (\delta^j)^2\|g^j\|^2 - 2\delta^j(\Phi^*_\text{PS}(\omega) - L^j_{\omega}).
\]
Then, from the definition of \(\delta^j\) in Step 2 of SSS procedure,
\[
\|\lambda^* - \lambda^{j+1}\|^2 \leq \|\lambda^* - \lambda^j\|^2 - \delta^j(\Phi^*_\text{PS}(\omega) - L^j_{\omega}),
\]
and using the result of Theorem 4, we obtain \(\|\lambda^* - \lambda^{j+1}\|^2 \leq \|\lambda^* - \lambda^j\|^2\). ■

Increasing the penalty parameter improves the quality of the solution converged as established by the following theorem.

Theorem 6. For any two penalty weight \(\omega_1\) and \(\omega_2\), where \(0 < \omega_1 < \omega_2\),
\[
\Phi(\omega_1, \lambda) \leq \Phi(\omega_2, \lambda) \leq \Phi^*_\text{PS}(\omega_2) \leq z^*_\text{MP}.
\]

Proof. From (23), we have \(L^j_{\omega_2} - L^j_{\omega_1} = (\omega_2 - \omega_1)|\sum_{l \in E} g^j_l| \geq 0\). Thus, \(L^j_{\omega_2} \geq L^j_{\omega_1}\). Subsequently from (24), we have,
\[
\min L^j_{\omega_2} \geq \min L^j_{\omega_1},
\]
\[
\max \Phi(\omega_2, \lambda) \geq \max \Phi(\omega_1, \lambda),
\]
\[
\Phi^*_\text{PS}(\omega_2) \geq \Phi^*_\text{PS}(\omega_1).
\]
From Theorem 5, we already have \(\Phi^*_\text{PS}(\omega_2) \leq z^*_\text{MP}\). ■

While Theorem 6 states that the solution quality of SSS improves with penalty parameter, we note that choosing \(\omega\) very large may cause ill-conditioning and numerical instability.
4.2.2 Bound Estimation: Variable Target Value Method

The SSS procedure uses an estimate of $\Phi^*_P(\omega)$ for the surrogate subgradient optimization. Rather than using a static estimate, we dynamically change this estimate in order to obtain a good quality primal feasible solution. Specifically, we modify the variable target value method (VTVM) presented in Lim and Sherali (2006) and incorporate backtracking to improve the target value estimation. Since the SSS method can converge to primary feasible but suboptimal solution, we integrated a backtracking phase within the VTVM to improve the quality of the feasible solution.

We modify the SSS method by replacing $\Phi^*_P(\omega)$ with a dynamically adjusted estimate $\Phi^j_P$ (target value). Analogous to Theorem 3, it can be shown that $L^j_\omega < \Phi^j_P$ holds true for each iteration $j$. In choosing the estimate $\Phi^j_P$, the goal is to approximate $z^*_MP$ as close as possible. In standard VTVM method, the target value $\Phi^j_P$ is increased as long as the convergence rate is satisfactory and then decreased to close in on an optimal solution. In our adaptation, we increase the target value $\Phi^j_P$ with a controlled rate until we find a primal feasible solution. Finding a primal feasible solution, as explained in Section 4.2.1, indicates that the target value is an overestimation of $\Phi^*_P(\omega)$. This primal feasible solution, however, maybe a low quality suboptimal solution. Therefore, with a backtracking phase, we revise the latest target value to obtain a better primal feasible solution. Specifically, after encountering with a primal feasible solution, we return back to a past iteration where the target value underestimates the current solution’s objective value. Then, the modified SSS repeats the iteration with a smaller step size in an effort to find an improved primal feasible solution.

We first provide the notation used in VTVM with backtracking and then present the modified steps of the SSS procedure. Next, we briefly discuss the convergence behavior of the SSS with backtracking. Note that we replace $\Phi^*_P(\omega)$ with $\Phi^j_P$ in the remainder steps of the SSS procedure.

**Notation for SSS with Backtracking VTVM:**

- $\Phi^j_P$: target value at iteration $j$
- $\Phi^{\overline{u}}_P$: upper bound on the optimal solution at iteration $j$
- $\Phi^{\overline{b}}_P$: lower bound on the optimal solution value
- $(x^*, y^*)$: an optimal solution to (MP)
- $\Delta_j$: accumulated improvements since the last Lagrangian function improvement until the beginning of iteration $j$
- $\varepsilon_j$: acceptance tolerance that the current incumbent value $L^j_\omega$ is close to the target value $\Phi^j_P$ in iteration $j$
- $\sigma$: acceptance interval parameter
- $\eta_j$: fraction of cumulative improvement that is used to increase the target value in iteration $j$
- $\varepsilon_{\text{GAP}}$: optimality gap threshold
Modified Steps of the SSS Procedure with Backtracking VTVM:

Initialization. Execute Steps I.1, I.2, I.3 of the original SSS procedure, and,

I.4. Set \( \Phi_{PS}^{j=1} = \Phi_{PS}, \ \bar{\Phi}_{PS}^{j=1} = +\infty, \ \eta_j = 0.35, \ \sigma = 0.2, \Delta_j = 0, \) and \( \varepsilon_j = \sigma(\Phi_{PS}^{j=1} - L_{\omega}^{j=0}). \)

Step 1. Subproblem Solution & Backtracking:

1.1. For \( k = 1,2,\ldots,|K|, \) Repeat:
   1.1.a. Solve subproblem \( (PS_k) \) in (28) by setting \( (x^l,y^l)_s = (x^{j-1},y^{j-1})_s \) for \( s \in K, s \neq k \) and obtain \( (x^l,y^l)_k. \) Denote \( (x^l,y^l) = (x^l,y^l)_k \cup \{(x^{j-1},y^{j-1})_s | s \in K, s \neq k\}. \)
   1.1.b. If \( (x^l,y^l) \) is primal feasible, then
      i. Set \( (x^*,y^*) = (x^l,y^l), \)
      ii. Set \( \bar{\Phi}_{PS}^j = L_{\omega}^j = f(x^l,y^l), \)
      iii. Set algorithm parameters, variables, and solutions back to iteration \( v, \) i.e.,
          where \( v = \max\{i: \Phi_{PS}^i < \bar{\Phi}_{PS}^j = L_{\omega}^j\}, \)
      iv. Set \( j = v, \)
      v. Set \( \beta = \beta/2 \) and repeat iteration \( j \) with updated multipliers \( \lambda^j = \lambda^{j-1} + \delta^{j-1}g^{j-1}. \)
   1.1.c. If the following improvement condition is satisfied,
         \[
         L_{\omega}^j = \hat{L}(\omega,\lambda^j,x^l,y^l) < \hat{L}(\omega,\lambda^j,x^{j-1},y^{j-1}),
         \]
         where \( (x^l,y^l) = (x^l,y^l)_k \cup \{(x^{j-1},y^{j-1})_s | s \in K, s \neq k\}, \)
         then go to Step 2, otherwise continue with the next \( k. \)
1.2. Set \( (x^l,y^l) = (x^{j-1},y^{j-1}). \)

Step 2. Subgradient Optimization & VTVM:

2.1. If \( L_{\omega}^j > \Phi_{PS}^j - \varepsilon_j, \) then
   2.1.a. Update the target value \( \Phi_{PS}^{j+1} = \min\{L_{\omega}^j + \varepsilon_j + \eta_j\Delta_j , \bar{\Phi}_{PS}^j\}, \)
   2.1.b. Update the threshold \( \varepsilon_{j+1} = \sigma(\Phi_{PS}^{j+1} - L_{\omega}^j), \)
   2.1.c. Reset \( \Delta_j = 0, \)
   2.1.d. Update \( \eta_{j+1} = \min\{2\eta_j, 1\}, \)
       otherwise set \( \Phi_{PS}^{j+1} = \Phi_{PS}, \ \bar{\Phi}_{PS}^{j+1} = \bar{\Phi}_{PS}, \ \eta_{j+1} = \eta_j \) and \( \varepsilon_{j+1} = \varepsilon_j. \)
2.2. Update \( \Delta_{j+1} = \Delta_j + (L_{\omega}^j - L_{\omega}^{j-1}) \)
2.3. Update Lagrangian multipliers:
         \[
         \lambda^{j+1} = \lambda^j + \delta^j g^j,
         \]
         where \( 0 < \delta^j = \beta(\Phi_{PS}^j - L_{\omega}^j)/\|g^j\|^2 \) and \( 0 < \beta < 1. \)

Step 3. Check the stopping criteria:

3.1. If \( \left(\bar{\Phi}_{PS}^j - L_{\omega}^j\right) \leq \varepsilon_{\text{GAP}} \) or \( \|\lambda^{j+1} - \lambda^j\| \leq \varepsilon, \) then terminate with Step 4;
       otherwise set \( j = j + 1 \) and go to Step 1.

Step 4. Terminate with \( (x^*,y^*). \)
The SSS with backtracking VTVM initializes the target value $\Phi_{PS}^j$ with an underestimation $\Phi_{PS}$ of the dual optimal value, e.g., linear programming relaxation. From Lemma 3, it can be shown that the Lagrangian multipliers provide a proper direction and thus the dual solution $L^j_{\omega}$ is non-decreasing. When the dual solution is primal feasible, we perform the backtracking phase in Step 1.1b. This backtracking helps improve the quality of the subsequent feasible solutions by reverting to an iteration $v$ satisfying $\Phi_{PS}^v < L^j_{\omega}$ and repeat the iteration $j$ with smaller step size. As the $L^j_{\omega}$ closes in on the target value such that $L^j_{\omega}$ is within $\varepsilon_j$ threshold of $\Phi_{PS}^j$, then Step 2.1.a updates the target value based on the accumulated improvement $\Delta_j$ and $\Phi_{PS}^j$. This update guarantees that the dual solution and the target value is separated by at least $\varepsilon_j$ while ensuring that the target value does not exceed the upper bound. The threshold $\varepsilon_j$ is updated in Step 2.1.b.

Choosing large values for $\sigma$ increases $\varepsilon_j$. With higher $\varepsilon_j$ values, we are more likely to consider that the $L^j_{\omega}$ is close to the target value and thus update the target value more frequently and with larger increments (Step 2.1.a). This can result in poor feasible solutions as the upper bound $\Phi_{PS}^j$ might not have decreased sufficiently. In contrast, lower $\sigma$ values reduce the convergence rate. The required ranges for acceptance interval parameter and fraction of cumulative improvement are $\sigma \in (0, 1/3]$ and $\eta_j \in (0, 1]$ (Lim and Sherali, 2006). The algorithm terminates and returns the best primal feasible solution when the gap between the best feasible solution and the Lagrangian dual function value falls below the optimality gap threshold ($\varepsilon_{\text{gap}}$).

5 Computational Experiments

We report on the results of two computational experiments. First, we investigate the computational and solution quality performance of the proposed approach for solving the ATD-PDP. Next, we present the results of implementing AAAP in a real-world case study using the Southern California region discussed in Hall (2002). The SSS with backtracking VTVM is programmed in Matlab R2008a and integer programs are solved with CPLEX 12.1. All experimental runs are conducted on a PC with Intel(R) Core 2 CPU, 1.66 GHz processor and 1 GB RAM running on Windows XP Professional. In the following section, we report on the computational results of the two variants of the SSS method, namely SSS with backtracking VTVM (SSS-B-VTVM) and VTVM base SSS without backtracking (SSS-VTVM).

5.1 Evaluation of the Solution Algorithm

We generated a set of test problems varying from small to large problem scenarios. Since the ATD-PDP is a new problem, no benchmark datasets are available. In generating the data sets, we adhered to the development procedure described in Solomon (1987). The problem scenarios have one depot and one or two airports each with three flight itinerary options for each customer. The third option represents the recourse flight itinerary option. For a problem scenario with $n = |C|$ customers and $m = |H|$ airports, we first generate $(1 + m + n)$ locations from a uniform distribution over the square bounded by $[0, 10(1 + m + n)] \times 0,$
10(1 + m + n)]. Next, we randomly label the nodes as the depot, airports and customers to avoid any association between the location and identity of a node. The travel time between nodes is calculated as the Euclidean distances between them. The travel cost between two nodes is set equal to their travel time.

For each airport $h$, the departure times $Q^h_r$ of flights are independent and identically distributed according to a uniform distribution $U[\varphi / |K|, \theta]$ where $|K|$ is the number of available vehicles; $\varphi$ is the heuristic solution to a TSP problem consisting of the depot (origin), all customers and the airport (destination) and obtained through the greedy next best routing heuristic. The cost of flight itinerary options $F^h_{ir}$ are independent and identically distributed according to a uniform distribution $U[a, b]$ where $a$ and $b$ are the bounds set as 100 and 600, respectively. The flight itinerary options are sorted from cheapest to most expensive and assigned to the flight itineraries based on the starting times such that cheaper itineraries start earlier.

We have conducted experiments using 5, 7, 10 and 15 customer cases. For each experiment scenario, we generated 10 independent instances and solve them using CPLEX, SSS-VTVM, and SSS-B-VTVM. Since there is no prior work on ATD-PDP, we compare the proposed methods with the CPLEX solution of (MP) as an integrated model. We restricted the solution time to 3 hours for all methods and report the best feasible solution attained within the time limit for each instance. In total, we have solved 300 problem instances using both methods. We first present the results of SSS-VTVM. In this method, we terminate the solution procedure when a primal feasible solution is found. Table 2 presents the comparative solution quality and computational performance results and Table 1 describes the column headings. Table 2 optimality results are based on the gap between the best solutions found in each method and the lower bound from CPLEX. For each problem scenario, we report the average, minimum, and maximum optimality gap of the methods and the comparison of the CPU time in terms of a ratio. The CPU time ratio metric is selected since we report the performance across all the instances.

| C | Number of customers |
| H | Number of airports |
| K | Number of vehicles |

**Table 1.** Description of column headings in Table 2.

| CPLEX Gap (%) | Gap between the best feasible solution and lower bound of CPLEX |
| SSS Gap (%) | Calculated as (SSS solution - CPLEX Lower Bound)/CPLEX Lower Bound. |
| CPU Time Ratio | The ratio of CPLEX's CPU time to SSS's CPU time |
| Hit | Percentage of time that SSS or CPLEX finds an optimum solution |

We first consider the results without backtracking, i.e., SSS-VTVM. For small size problems with 5 and 7 customers, the CPLEX's average gap across all scenarios is 0.5% whereas the SSS's gap is 2.1%. On the average, CPLEX finds the optimum in 96% of the cases and SSS finds in 30% of the cases. While the CPLEX's solution quality performance is slightly better than that of SSS's, the difference is small. Further, SSS is able to attain good quality solutions up to 346 times faster and on the average 29 times faster.
<table>
<thead>
<tr>
<th>C</th>
<th>H</th>
<th>K</th>
<th>Ave</th>
<th>Min</th>
<th>Max</th>
<th>Hit</th>
<th>Ave</th>
<th>Min</th>
<th>Max</th>
<th>Hit</th>
<th>Ave</th>
<th>Min</th>
<th>Max</th>
<th>Hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.8</td>
<td>0.0</td>
<td>5.2</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0.1</td>
<td>0.0</td>
<td>0.8</td>
<td>70</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.3</td>
<td>0.0</td>
<td>5.8</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.0</td>
<td>1.5</td>
<td>70</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.7</td>
<td>0.0</td>
<td>2.1</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0.3</td>
<td>0.0</td>
<td>1.8</td>
<td>80</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.8</td>
<td>0.0</td>
<td>5.6</td>
<td>40</td>
<td>25</td>
<td>0</td>
<td>146</td>
<td>0.1</td>
<td>0.0</td>
<td>1.2</td>
<td>80</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.5</td>
<td>0.0</td>
<td>8.2</td>
<td>50</td>
<td>8</td>
<td>0</td>
<td>28</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>90</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>2.8</td>
<td>0.0</td>
<td>7.1</td>
<td>20</td>
<td>31</td>
<td>1</td>
<td>228</td>
<td>0.5</td>
<td>0.0</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.5</td>
<td>0.0</td>
<td>6.2</td>
<td>20</td>
<td>10</td>
<td>1</td>
<td>28</td>
<td>0.8</td>
<td>0.0</td>
<td>4.0</td>
<td>30</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.6</td>
<td>0.0</td>
<td>2.7</td>
<td>50</td>
<td>11</td>
<td>0</td>
<td>38</td>
<td>0.4</td>
<td>0.0</td>
<td>2.7</td>
<td>70</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.9</td>
<td>0.1</td>
<td>5.7</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>34</td>
<td>0.5</td>
<td>0.0</td>
<td>1.8</td>
<td>30</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>1.4</td>
<td>0.0</td>
<td>3.3</td>
<td>10</td>
<td>102</td>
<td>6</td>
<td>344</td>
<td>0.9</td>
<td>0.0</td>
<td>3.3</td>
<td>30</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>4.2</td>
<td>0.0</td>
<td>14.9</td>
<td>20</td>
<td>54</td>
<td>4</td>
<td>346</td>
<td>2.9</td>
<td>0.0</td>
<td>14.9</td>
<td>30</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>3.2</td>
<td>0.0</td>
<td>11.6</td>
<td>11</td>
<td>99</td>
<td>18</td>
<td>399</td>
<td>1.8</td>
<td>0.0</td>
<td>6.8</td>
<td>22</td>
</tr>
</tbody>
</table>

The CPLEX's gap for medium size problems with 10 customers averages 4.8% across all scenarios and an optimum is found for 45% of the cases. In comparison, the SSS has an average gap of 6.2% and finds an optimum for 4% of the cases. While the CPLEX's solution quality performance is slightly better than that of SSS, the difference is small. The SSS is able to attain good quality solutions up to 2,205 times faster and on the average 296 times faster. For the large size problems with 15 customers, the CPLEX's average gap is 16.2% with an optimality hit rate of 7% of the time. While the SSS's average gap is 9.8%, it is not able to find a verifiable optimal solution. Unlike small and medium size problem scenarios, SSS has a better average gap performance than that of CPLEX's for large size problems. As before, the SSS is much more efficient than CPLEX, e.g., up to 1,892 times faster and on the average 197 times faster.

The last seven columns of Table 2 present the results for SSS-B-VTVM which improves over the solution quality performance of the SSS-VTVM through the backtracking phase. For small size problems the average gap is reduced to 1.2% and the optimality hit rate is increased to 54%. These improvements are attained without sacrificing the CPU time performance advantage over CPLEX. For medium size problems, the average gap performance of SSS-B-VTVM is better than that of the CPLEX, e.g., 4.0% versus 4.8%, respectively. While this improvement comes with reduced CPU time performance advantage, the SSS-B-VTVM is still 168 times faster than CPLEX on the average. For large size problems, the average gap performance improves slightly and is about half of that of the CPLEX, e.g., 8.0% versus
16.2%, respectively. The CPU time performance is reduced by a third but still about 131 times faster than CPLEX on the average. Across all problem instances, the CPLEX, SSS-VTVM, and SSS-B-VTVM have on the average 5.3%, 4.8%, and 3.4% optimality gap, respectively. In terms of CPU performance, SSS-VTVM and SSS-B-VTVM are on the average 138 and 87 times faster than CPLEX, respectively.

Based on the results in Table 2, we study the effect of number of airports, vehicles and customers on the performance of SSS-B-VTVM (Figure 3). The effect of the number of airports is illustrated in Figure 3a. With increasing number of airports, the optimality gap of SSS-B-VTVM increases at a lower rate than that of the CPLEX. For medium and large instances, the CPU performance of SSS-B-VTVM is highest with single airport and, for small instances, highest with two airports. This is because as the problem size increases, flight itinerary assignment and routing decisions become more interrelated making it difficult to solve as an integrated model. Note that the CPU time advantage of SSS-B-VTVM is significantly reduced for two airport case in the large problem instances. This is attributable to the time limit which is mostly restrictive for CPLEX than SSS-B-VTVM.

![Figure 3a](image1.jpg)

**Figure 3a.** Effect of number of airports and customers on the performance of SSS-B-VTVM.

Figure 3b illustrates the effect of the number of vehicles. The gap performance of SSS-B-VTVM is robust with respect to the number of vehicles. This can be explained by the fact that additional vehicles are utilized to a lesser extent, hence their effect on the optimality is marginal. In comparison, the gap performance of CPLEX is reduced, especially, for large problems. This difference is due to the vehicle-based decomposition of SSS-B-VTVM, which is able to find quality solutions in the presence of underutilized fleet capacity. The CPU time advantage is relatively reduced, beginning with 4 vehicles in medium size problem instances. This is, indeed, a result of the time limit which makes the numerator of the CPU time ratio invariant to the number of vehicles.

5.2 Case Study

To assess the benefits of implementing AAAP, we conducted a case study in a Southern California MAR using real flight itinerary information and airport locations. The performance of AAAP is compared to the single airport policy where the freight forwarder can only assign customers' air cargo loads to the flights departing from one airport.
5.2.1 Alternative Access Airports and Depot Locations

The Southern California MAR used in our experiments is described in Hall (2002) and illustrated in Figure 4. In this MAR, the Los Angeles International Airport (LAX) is the largest air-freight port. Hall (2002) suggests redirecting some of the domestic freight load to Long Beach Airport (LGB) or Ontario International Airport (ONT) to reduce the load and congestion in the LAX airport. As discussed in Hall (2002) and Chayanupatkul et al. (2004), a forwarder rarely considers more than two alternative access airports. Hence, we consider LGB and LAX as the two alternative access airports. For the location of the depot, we experimented with three location scenarios: adjacent to LAX, adjacent to LGB, and in-between LAX and LGB. We denote these depot location scenarios as DLAX, DLGB, and DMID, respectively. For the two scenarios of DLAX and DLGB, we randomly and uniformly select the depot location in a one-mile radius region with the airport in the center. For DMID scenario, we select the depot location within a one-mile radius of the city of Compton such that the travel time is identical to both the LAX and LGB airports. These regions are illustrated with dashed circles in Figure 4.

5.2.2 Customer Locations

In all experiments, the fleet size is four vehicles and there are 15 customers. We consider the scenario where the air cargo loads are time-sensitive (shipped overnight). All customer loads are available for pick-up by 7:00 pm. We generate multiple case study instances, by uniformly sampling customer locations within the MAR region, i.e., rectangular region in Figure 4. The Google Maps API is used to generate the customer locations and calculate travel times. For each customer in each problem instance, we first uniformly sample a geographical coordinate (i.e., latitude and longitude) in the MAR region. Next, we determine the closest street address to this coordinate point through the Google Maps API. In case of an infeasible coordinate point (e.g., inside a lake), we re-sample for another coordinate. The travel times are estimated from the shortest paths accounting for speed limits using Google Maps API.

Figure 4. Southern California MAR used in the case study
5.2.3 Flight Itinerary Options

A forwarder, upon receiving a time-sensitive shipment order, can execute it via an integrator (e.g. FedEx, UPS), a mixed passenger-cargo (e.g. United Airlines, Delta Airlines, American Airlines), or a chartered/dedicated freighter. In this case study, we consider only the mixed passenger-cargo flight itinerary options, the most practiced option for small and mid-size forwarders.

Table 3. Case study flight itinerary options from LAX and LGB airports.

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Prob.</th>
<th>Origin</th>
<th>Airline</th>
<th>Departure Time</th>
<th>Mean Delay</th>
<th>Mean Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LAX</td>
<td>AA</td>
<td>22:15</td>
<td>7</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LGB</td>
<td>B6</td>
<td>22:59</td>
<td>11</td>
<td>324</td>
</tr>
<tr>
<td>BOS</td>
<td>19%</td>
<td>LAX</td>
<td>DL</td>
<td>21:05</td>
<td>10</td>
<td>324</td>
</tr>
<tr>
<td>FLL</td>
<td>13%</td>
<td>LGB</td>
<td>B6</td>
<td>21:15</td>
<td>10</td>
<td>307</td>
</tr>
<tr>
<td>IAD</td>
<td>24%</td>
<td>LAX</td>
<td>AA</td>
<td>21:00</td>
<td>11</td>
<td>293</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LGB</td>
<td>B6</td>
<td>21:08</td>
<td>23</td>
<td>310</td>
</tr>
<tr>
<td>JFK</td>
<td>44%</td>
<td>LAX</td>
<td>AA</td>
<td>21:20</td>
<td>10</td>
<td>312</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LGB</td>
<td>B6</td>
<td>21:00</td>
<td>17</td>
<td>315</td>
</tr>
</tbody>
</table>

* Next day departure

We assume the final destination of a customer's air cargo is a domestic destination with direct flights from both the LAX and the LGB. Accordingly, we consider four major US airports as the destinations: Boston Logan International Airport in Massachusetts (BOS), Fort Lauderdale-Hollywood International Airport in Florida (FLL), Dulles International Airport in Dallas, Texas (IAD), and John F. Kennedy International Airport in New York (JFK). As for the airlines, we considered American Airlines (AA) and Delta Airlines (DL) for the LAX airport and JetBlue Airways (B6) for the LGB airport. In determining the cargo destination for each customer, we randomly assigned each customer's load to one of the four destinations. The probability distribution used in this assignment is based on the frequency of the outgoing flights to each destination from each airport. These probabilities are presented in the second column of Table 3. For each customer, there are in total four flight itinerary options, e.g., two options from each airport.

We arbitrarily selected the operation day as August 16, 2010 and collected the flight itinerary information from the BTS database\(^1\). The flight itinerary information, including the average departure delay and elapsed time (i.e. overall taxi-out to taxi-in time) in August 2010, is listed in Table 4. The departure delays are incorporated in the total delivery time by assuming flights depart late with their respective mean delay. While the first flight departure times are rather similar in two airports, the second flight departure times are notably different for some destinations. We consider the starting time of a flight itinerary as the departure time of its first flight.

---

5.2.4 Case Study Results

We evaluate the performance of different policies based on total delivery time including road and air travel times. Note that the practical implementation of the AAAP would account for forwarder's negotiated terms with air-carriers, cost structure of road transportation operations, and pricing models (Azadian et al. 2012). However, cost performance, i.e., the total delivery time, used in this case study provides ample policy comparison opportunity. Specifically, given a solution, we calculate total delivery time as the sum of road travel times by all vehicles and the total time elapsed for each customer load from the start of the operation (19:00) until its delivery time to the destination airport. We have conducted three sets of experiments corresponding to each depot location scenario (DLAX, DLGB, and DMID). In each set, we consider three different airport access policies: AAAP (with LAX & LGB), LAX only, and LGB only. For each depot location, we generated 10 problem instances and solve them with the SSS-B-VTVM algorithm under each access policy.

The case study flight itinerary options in Table 3 show that there is no significant difference between the first flight options across the two airports. Further, the recourse flight itinerary options only differ for the loads going to BOS or FLL. Hence, in this case study, the performance differences of the three airport policies are primarily attributable to the road travel time and the small differences in the flight itinerary options. We note that the performance advantage of utilizing alternative access airports would increase when the flight itinerary options' starting times and flight itinerary durations/costs (especially for multi-leg itineraries) vary between the alternative airports. Therefore, we compare the airport policies based on the delivery time saving potential in each depot location scenario. For this, we estimate a lower bound on the total delivery time as a summation of the lower bound for flight itinerary time and road travel. The lower bound for the flight itinerary time is estimated by assigning each customer load to the cheapest itinerary accessible. The lower bound for the road travel time is calculated by solving a minimum spanning tree connecting all the nodes.

Table 4 presents the total delivery time in minutes for all problem instances in each depot location scenario and under three access policies (LAX & LGB, LGB, LAX). For AAAP policy, i.e., LAX &LGB, we report the percentage of the time that the LAX airport is selected. Last two rows in Table 4 present the average and standard deviations of the results. The column 'LB' denotes the lower bound on the total delivery time for each depot location scenario.

The AAAP policy dominates the single airport policy in all depot location scenarios and in all problem instances. The AAAP's impact on the total delivery time can be assessed through the following performance measure:

$$\rho = \frac{z_{AAAP} - LB}{\min(z_{LGB}, z_{LAX}) - LB} \%$$  \hspace{1cm} (32)

where $z_{AAAP}$, $z_{LGB}$, and $z_{LAX}$ correspond to the solutions of three airport policies.

The performance measure in (32) indicates the percentage total delivery time improvement of the AAAP policy over single airport policies. In the case of DLGB depot location, the AAAP policy improves the total delivery time performance on the average by 58% . The improvements range between 47% and 77%. Similarly, for the DMID depot location, the average improvement of AAAP is 63% and the range is between 45% and 91%. In the case of DLAX depot location, the average improvement is 57% and the range is between 36% and 87%. Overall, the AAAP's improvement over single airport policies is 59% on the
average across all depot locations.

Table 4. Case study results for three depot location scenarios (DLGB, DMID, DLAX) and three airport access policies (AAAP, LGB, LAX)

<table>
<thead>
<tr>
<th>No</th>
<th>LB</th>
<th>AAAP</th>
<th>LGB</th>
<th>LAX</th>
<th>( \rho )</th>
<th>LB</th>
<th>AAAP</th>
<th>LGB</th>
<th>LAX</th>
<th>( \rho )</th>
<th>LB</th>
<th>AAAP</th>
<th>LGB</th>
<th>LAX</th>
<th>( \rho )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave.</td>
<td>6.805</td>
<td>6.991</td>
<td>7.131</td>
<td>7.616</td>
<td>58%</td>
<td>6.802</td>
<td>6.995</td>
<td>7.121</td>
<td>7.121</td>
<td>63%</td>
<td>6.810</td>
<td>7.075</td>
<td>7.107</td>
<td>7.664</td>
<td>57%</td>
</tr>
<tr>
<td>Sdev.</td>
<td>34</td>
<td>35</td>
<td>41</td>
<td>201.5</td>
<td>10%</td>
<td>32</td>
<td>48</td>
<td>2%</td>
<td>63</td>
<td>14%</td>
<td>38</td>
<td>66</td>
<td>50</td>
<td>178</td>
<td>18%</td>
</tr>
</tbody>
</table>

In Figures 5-7, we illustrate the routes identified for each depot location and airport policy for sample problem instances. These routes are turn-by-turn routes from the Google Maps API. The labels are "1" for the location of depot, "2" to "16" for the locations of 15 customers, and LAX and LGB for the airports. The label in parenthesis denotes the order of visit by the vehicle. Each color route corresponds to a unique vehicle. For instance, in Figure 5a, the vehicle with blue color route starts its trip from the depot located in Compton (i.e. node 1), visits customers 5, 6, 8, 3, delivers loads to LAX, and returns to the depot. Accordingly, the customer 5 is labeled 5(1), customer 6 is labeled 6(2), and so forth. In all instances, at most three of the four vehicles are used, indicating absence of recourse flight usage. In Figure 5, there are three vehicles in all airport policies. In Figures 6a and 7b only two vehicles are used since the third vehicle does not provide any additional benefit in terms of improving the total delivery cost. In Figures 5c, 6c and 7c, two vehicles deliver customer loads to both the LAX and LGB airports whereas the third vehicle visits only the LGB. In Figure 7a, the third vehicle is used to pick up and deliver the load of only customer 5.
6 Conclusion

We study a freight forwarder's operational implementation of AAAP in a MAR for air cargo transportation. The forwarder's AAAP implementation involves the task of selecting flight itineraries for a given set of heterogeneous air cargo customers, picking up their loads via a fleet of vehicles and then delivering to the airports in the region. The goal is to minimize the total cost of air and road transportation and service by simultaneously selecting the air cargo flight itinerary and scheduling pickup and delivery of multiple customer loads to the airport(s).

We formulated a novel model (ATP-PDP) which extends the existing PDP models to address the case where the delivery cost is both destination and time dependent. This model is further strengthened by preprocessing steps and special cuts. To overcome the computational complexity, we adapted an efficient solution method, SSS, based on Lagrangian decomposition. The SSS method overcomes the challenges associated with identical subproblems in standard Lagrangian decomposition and iteratively solves the ATP-PDP in parts. Since Lagrangian based methods, including SSS, can converge to a primal infeasible solution, we developed a modified
variable target methodology for subgradient optimization. The integrated method, SSS-B-VTVM, converges to a primal feasible solution and the solution quality can be controlled by trading-off the quality with computational performance.

We conducted an experimental study to assess the optimality gap and CPU time performance of SSS-B-VTVM and compared with those of CPLEX. The results show that the SSS-B-VTVM yields near-optimal primal feasible solutions, i.e., on the average 3.4% optimality gap compared to 5.3% of CPLEX. Further, the SSS-B-VTVM is able to achieve this performance on the average about 87 times faster than CPLEX and more than thousand times faster for some problems. In addition, we have applied the modeling and solution methodology for a case study in a Southern California MAR and compared the AAAP performance with single airport policies considering various depot location and customer scenarios. The computational results indicate that the AAAP is able to realize savings in the order of 36% to 91% of the potential saving opportunities. This research can be extended in multiple directions. The proposed approach can be used to evaluate forwarder’s locational decisions for its depot(s). Similarly, it can be used to assess the competitiveness of multiple airports in a MAR for air cargo shipments under various flight availability schedule scenarios.

Acknowledgments
This work was supported by funding grant DTRT06-G-0039 from the US Department of Transportation through the University Transportation Center at University of Toledo (UT-UTC).

References
Conejo, A., Castillo, E., Minguez, R., Garcia-Bertrand, R., 2006. Decomposition techniques in
mathematical programming engineering and science applications. Springer, Berlin; Heidelberg; New York.


Hall, R.W., 2002. Alternative Access and Locations for Air Cargo. METRANS Transportation Center, University of Southern California.


Miguel Andres, F., 2007. Analysis of the efficiency of urban commercial vehicle tours: Data
collection, methodology, and policy implications. Transportation Research Part B: Methodological 41(9), 1014-1032.