VALUE OF ITS INFORMATION FOR CONGESTION AVOIDANCE IN INTER-MODAL TRANSPORTATION SYSTEMS[1]

FINAL REPORT

Focus Area: Infrastructure Utilization
Year 4

Principal Investigators:

Wayne State University
Industrial & Manufacturing Engineering
4815 Fourth Street, Detroit, MI 48202, USA

- Alper E. Murat (PI), Assistant Professor
  Tel: 313-577-3872; Fax: 313-577-3388
  E-mail: amurat@wayne.edu
- Ratna Babu Chinnam (Co-PI), Associate Professor
  Tel: 313-577-4846; Fax: 313-578-5902
  E-mail: r_chinnam@wayne.edu

Wayne State University
Civil & Environmental Engineering
5050 Anthony Wayne Drive, Detroit, MI 48202, USA

- Snehamay Khasnabis (Co-PI), Professor
  Tel: 313-577-3915 Fax: 313-577-3881
  E-mail: skhas@eng.wayne.edu

Richard Martinko
Director, University Transportation Center
University of Toledo

Christine Lonsway
Assistant Director, University Transportation Center
University of Toledo

[1] Throughout this document "Inter-modal Freight" refers to the "Shipment of freight involving more than one mode of transportation (road, rail, air, and sea) during a single, seamless journey".

Table of Contents Page No

SUMMARY OF RESULTS

Our project has three major mile-stones for the third year:

- **Mile-stone #1:** Compare the performance of static and dynamic models through the case studies
    a. SDP vs. AO* for dynamic routing on intermodal networks
    b. Deterministic equivalent vs. PHA for stochastic programming for operational response
- **Mile-stone #2:** Develop extensive scenarios based on loading levels at inter-modal facilities and transportation network, disruption and incident states, and fidelity of real-time information, etc.
- **Mile-stone #3:** Apply Static and Dynamic models for the different scenarios in a simulation test-bed to perform cost/benefit analysis

The Research Team, made up of Dr. Alper Murat (Project PI), Dr. Ratna Babu Chinnam (Project Co-PI), Dr. Snehamay Khasnabis (Project Co-PI), doctoral students – Farshid Azadian and Nezir Aydin, has made very good progress with respect to all these three milestones over the third year.

In what follows, we summarize our achievement with respect to these milestones.

Mile-stone #1:

Majority of our efforts for this milestone has been devoted in developing novel and technically rigorous methods for tackling hard-to-solve dynamic models encountered in practice and comparison their solutions with those of the static counterparts. We have met and exceeded our goals in terms of the practicality of the models and algorithms developed.

While we have initially aimed developing AO* algorithm based methods for solving routing models on stochastic time-varying schedule-based networks, the practical models proved to be far more complex than those that can be tackled with AO*. In practice, there are such complicating factors as the multiple shipments (pickup and delivery) and carrier schedules which

not only increase the complexity of the models but also make the solution intractable. As a result, we have developed mathematical programming based heuristics which are not only novel for the dynamic routing problems studied but also contribute to the methodological body of knowledge in solving similar problems. The models we developed focused on an increasingly important area of intermodal transportation which is the air-road air-cargo transportation. We considered scheduling and dynamic routing of a single truck as well as a fleet of trucks. In addition, we considered single access airports as well as multi-airport regions with alternative access airports. Lastly, we also compared static and dynamic routing strategies where the static routing implies the decision makers commit to a shipment scheduling and routing and does not change it. In comparison, the dynamic routing corresponds to the case where the decisions are made online, based on the real time information available, congestion state of the intermodal network, as well as other factors such as cargo characteristics, customer characteristics, carrier and flight characteristics and airport factors.

In terms of the operational response models, we have developed novel disruption response models. We have met and well exceeded this part of the milestone such that we developed not only deterministic equivalent and Progressive Hedging Algorithm (PHA) based approaches but also developed Sample Average Approximation (SAA) based approaches. One major contribution is the development of an extended version of the widely-used SAA approach. We are able to attain 5-10 folds of speed improvement while finding operational response solutions with same or better performance. One particular operational response model considered is the reliable intermodal facility selection and transportation flow allocation problem. In this model, we consider random disruptions causing failure of intermodal facilities (for a given duration), hence requiring re-routing of the transportation flow commodities through surviving facilities as well as through alternative means such as emergency facilities (e.g., long hauling rather than shipping through rail).

Mile-stones #2 & #3:

These two milestones aims at investigating the benefits of the models and methods developed while considering loading levels at inter-modal facilities and transportation network, disruption and incident states, and fidelity of real-time information. In these investigative studies, we

compared the impact of utilizing dynamic versus static routing and operational response models. Specifically, we sought answers for such questions as,

1. What are the benefits of dynamic routing based on real-time information for a freight forwarder picking up multiple customer shipments, scheduling and assigning them to different flights, and delivering them to the corresponding airports on time by routing on the road network?
2. What are the potential savings in responding intermodal facility disruption through re-allocating the flow of freight to available and emergency facilities? And under what disruption scenarios the impact is highest?

For the benefits of dynamic routing, we have studied the transportation of multi-customer air-cargo shipment from a freight forwarder's perspective. In evaluating different scenarios, we considered different distributions of customer locations, varying cargo characteristics, various flight itinerary options, and depot location of the freight forwarder. We have developed a case study using a multi-airport region in the Southern California. Similarly, for the benefit of using operational response models in coping with disruption in the intermodal facility network, we have experimented with different network scenarios, failure/disruption probabilities and severities, and cost of response and recovery recourse actions.

Description of Sections A and B

In Section A, we consider a freight forwarder's operational implementation of alternative access airport policy in a multi-airport region for air cargo transportation to evaluate static and dynamic routing policies. Given a set of heterogeneous air cargo customers and their air-cargo characteristics, 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) such that the cargo is delivered to the airport on-time for the assigned flight itineraries. 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. The contributions of this study are as follows:

1. Formulate a novel model for the operational air cargo pickup and delivery and flight assignment problem of freight forwarders.
   - This problem generalizes the well-known pickup and delivery problems (PDP) since the delivery cost of customer air cargo is both destination and time dependent.
2. Develop a novel and highly efficient solution method based on the Lagrangian decomposition.
   - A successive subproblem solution (SSS) technique is developed to overcome the challenges associated with "identical subproblems" when standard Lagrangian decomposition method is applied. This method is further enhanced by developing a variable target value method with backtracking for the subgradient optimization.
3. Demonstrate the benefits of using dynamic routing with alternative access airports for air cargo transportation through a case study in Southern California multi-airport region.

This study further opened up new research venues:

1. The freight forwarders can use the proposed approach as a what-if tool to determine the depot locations to best serve their customers. They can also use it to determine which airports to frequent more often and which carriers to contract with in the long-term and make spot market purchases. The airports and airlines can use the proposed approach for competitiveness analysis for air cargo shipment.

2. The proposed model generalizes the PDP and can be used to study similar problems in other application areas by other researchers. The methodology developed can be used for other vehicle routing problems where standard Lagrangian decomposition leads to identical subproblems. Similarly, existing solution methods for PDPs can be adapted to tackle the model and its extensions.

In section B, we consider disruptions in intermodal facilities used by a company to transport it freight. Facing a disruption scenario, the company re-allocates the flow of goods through those facilities that survive as well as resort to emergency alternatives (e.g., expediting). For this problem, we present a novel hybrid method, swarm intelligence based sample average approximation (SIBSAA), for solving the capacitated reliable facility location problem (CRFLP). The CRFLP extends the well-known capacitated fixed-cost facility problem by accounting for the unreliability of facilities. The standard SAA procedure, while effectively used in many applications, can lead to poor solution quality if the selected sample sizes are not sufficiently large. With larger sample sizes, however, the SAA method is not practical due to the significant computational effort required. The proposed SIBSAA method addresses this limitation by using smaller samples and repetitively applying the SAA method while injecting social learning in the solution process inspired by the swarm intelligence of particle swarm optimization. We report on experimental study results showing that the SIBSAA improves the computational efficiency significantly while attaining same or better solution quality than the SAA method. The results of computational experiments also indicate that the benefit of having flexibility in the inter-modal transportation system increases with increasing failure likelihood and severity. We also note that the flexibility levels depend on the capacity as well as various cost factors such as recourse costs.

A. Operational Dynamic Routing in Inter-modal Schedule-based Networks Using Real-time Congestion Information: Case of Air-Road Transportation with Alternative Access Airports

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 (Başar 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 pickup and delivery problems in terms of model structure and objective function. For instance, in ATD-PDP, the delivery cost of customer air cargo is both destination and time dependent. Second, we propose a novel and highly efficient solution method based on the Lagrangian decomposition. This method overcomes the challenges associated with *'identical subproblems'* when standard Lagrangian decomposition method is applied. For subgradient optimization, we

also develop a novel variable target value method with backtracking. Third, we demonstrate the benefits of using alternative access airports for air cargo transportation through a case study in 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 CTSPPD 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 cus-

tomer'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 (θ_d^{op}) and closing hours (θ_d^{cl}). 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 $c_{dj},\ \forall j \in C \cup H$. 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_{ir}^h, 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., $Q_r^h \geq \theta_d^{op}$.

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, $t > \max_{r \in R_h} \left\{Q_r^h\right\}$, 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 $F_{i0}^h > F_{ir}^h \ \forall r \in R_h$, 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:

$$f(h,i,t) = \begin{cases} \min\limits_{r \in R_h} \{F_{ir}^h | t \leq Q_r^h\} & if \ t \leq \max\limits_{r \in R_h} \{Q_r^h\} \\ F_{i0}^h, & otherwise \end{cases}$$

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 flights 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) $F_{ir'}^h \leq F_{ir}^h$ and $Q_r^h < Q_{r'}^h$ or (b) $F_{ir'}^h < F_{ir}^h$ and $Q_r^h \leq Q_{r'}^h$. Moreover, if (c) $F_{ir'}^h = F_{ir}^h$ and $Q_{r'}^h = Q_r^h$, considering either one is sufficient.*

Proof. 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 $F_{ir'}^h = F_{ir}^h$.*

Theorem 1 characterizes the airport delivery cost function after eliminating the dominated itineraries.

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. Proof Let us first consider single flight itinerary case where $f(h,i,t) = F_{ir}^h, \quad \forall t \leq Q_r^h$ and F_{i0}^h otherwise. Since $F_{i0}^h > F_{ir}^h$, the $f(h,i,t)$ is a step-function, which is non-decreasing and has a single discontinuity at Q_r^h. 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 $Q_{r'}^h < Q_r^h$ and $F_{ir'}^h < F_{ir}^h$. 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_r^h and $Q_{r'}^h$. 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_2^h for customer i (Q_1^h for customer j) will result in the penalty cost of F_{i0}^h (F_{j0}^h for customer j).

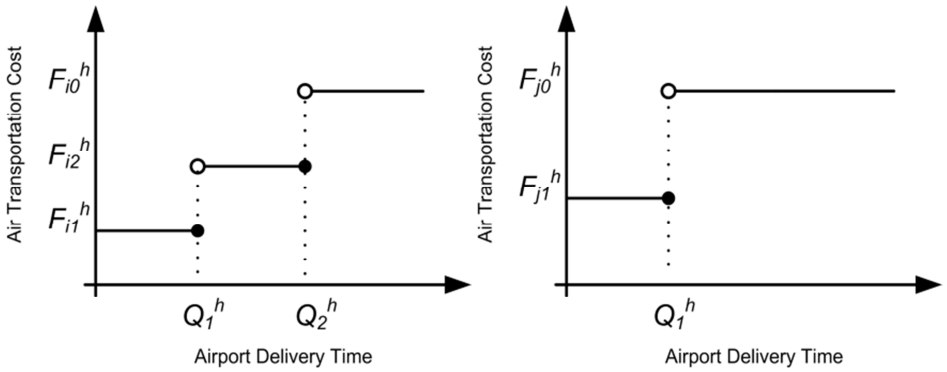


Figure 1: Illustrative airport h delivery cost function for customers $i,\ j \in C$; customer i has two flight itinerary options (left) and customer j has a single flight itinerary option (right).

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. 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. 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 $G_o = (V_o, E_o)$. 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^h_{ir}) is replaced with the delivery cost (F_{ir}) 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 $\forall h \in H$ edges $\forall (j, h) \in E_o,\ \forall j \in V_o \backslash \{h\}$ with

new flight node edges $(j, r) \in,\ \forall j \in V, \forall r \in R_h$ 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.

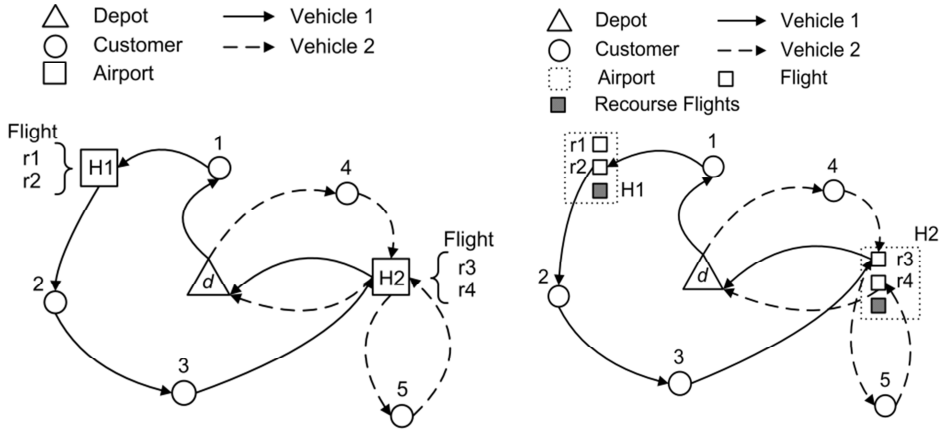


Figure 2: Illustration of a sample feasible solution in the original (a) and transformed (b) graphs.

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_{ij}^k denote the binary decision variable indicating whether vehicle k travels from node i directly to node j. Let y_{ir}^k 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_j^k. For the depot, we set $a_d^k = \theta_d^{op}$ for any vehicle $k \in K$. The formulation of the ATD-PDP, labeled (MP), is as follows.

$$\text{(MP)} \qquad z_{MP}^* = \min_{x,y} \sum_{k \in K} \left[\sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} x_{ij}^k + \sum_{i \in C} \sum_{r \in R} F_{ir} y_{ir}^k \right] \tag{1}$$

Subject to

$$\sum_{k \in K} \sum_{r \in R} y_{ir}^k = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{j \in V \setminus \{i\}} x_{ij}^k \leq 1 \qquad \forall i \in V, \forall k \in K \tag{3}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij}^k + \sum_{i \in V \setminus \{j\}} x_{ji}^k = 0 \qquad \forall j \in V, \forall k \in K \tag{4}$$

$$\left(x_{ij}^k - 1\right) M + a_i^k + t_{ij} \leq a_j^k \qquad \forall i \in V, \forall j \in V \setminus \{d, i\}, \forall k \in K \tag{5}$$

$$a_i^k \geq \theta_d^{op} \qquad \forall i \in C, \forall k \in K \tag{6}$$

$$a_r^k \leq min\left\{\theta_d^{cl} - t_{rd}, Q_r\right\} \qquad \forall r \in R, \forall k \in K \tag{7}$$

$$\left(y_{ir}^k - 1\right) M + a_i^k + t_{ir} \leq a_r^k \qquad \forall i \in C, \forall r \in R, \forall k \in K \tag{8}$$

$$2y_{ir}^k \leq \sum_{j \in V \setminus \{i\}} x_{ij}^k + \sum_{j \in V \setminus \{r\}} x_{rj}^k \qquad \forall i \in C, \forall r \in R, \forall k \in K \tag{9}$$

$$y_{ir}^k, x_{ij}^k \in \{0, 1\} \quad a_i^k, a_r^k \geq 0 \quad \forall i, j \in V | i \neq j, \quad \forall r \in R, \quad \forall k \in K \tag{10}$$

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.

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

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

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 ($\underline{a}_i^k$) is attained via a direct travel from the depot,

$$a_i^k \geq \underline{a}_i^k = \theta_d^{op} + t_{di} \quad \forall i \in C, \quad \forall k \in K, \tag{13}$$

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,

$$a_i^k \leq \overline{a}_i^k = \max_{r\in R}\left\{\min\left(Q_r, \theta_d^{cl} - t_{rd}\right) - t_{ir}\right\} \qquad \forall i \in C, \forall k \in K. \tag{14}$$

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

$$a_r^k \geq \underline{a}_r^k = \theta_d^{op} + \min_{i\in C}(t_{di} + t_{ir}) \qquad \forall r \in R, \quad \forall k \in K. \tag{15}$$

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},

$$M_{ij} = \overline{a}_i^k + t_{ij} \qquad \forall i \in V,\, \forall j \in V \backslash \{d, i\},\, \forall k \in K. \tag{16}$$

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 $\underline{a}_i^k + t_{ij} > \overline{a}_j^k$, 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.

$$x_{ji}^k \left(M - t_{ij} - t_{ji}\right) + \left(x_{ij}^k - 1\right) M + a_i^k + t_{ij} \leq a_j^k \ \ \forall i, j \neq i \in V \backslash \{d\}, \forall k \in K \tag{17}$$

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.

$$\sum_{j \in V \backslash \{i\}} x_{ij}^k = \sum_{r \in R} y_{ir}^k \quad \forall i \in C,\ \ \forall k \in K \tag{18}$$

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),

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

where $\lambda = \left(\lambda_1, \cdots, \lambda_{|C|}\right) \in \Re^{|C|}$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}.

$$\text{(LD)} \qquad \Phi^*_{LD} = \max_{\lambda} (\Phi(\lambda)). \tag{20}$$

The set splits into $|K|$ disjoint subsets, i.e. $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_{|\mathrm{K}|}$, where each Ω_{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,

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

where $\Phi_k(\lambda) = \min_{(x,y)\in\Omega_{\mathrm{k}}} L_k(\lambda, x, y)$ 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. $\Omega_1 = \Omega_2 = \cdots = \Omega_{|\mathrm{K}|}$. Hence, all subproblems have the same optimal solution with identical objective value. Accordingly, solving (LD) is equivalent to solving the following,

$$|K| \max_{\lambda} \left(\min_{(x,y)\in\Omega_{\mathrm{k}}} L_k(\lambda, x, y) \right) + \sum_{i\in C} \lambda_i \tag{22}$$

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,

$$\begin{aligned}\hat{L}(\omega,\lambda,x,y) &= \sum_{k\in K} J_k(x,y) + \sum_{i\in C}\lambda_i\left(1-\sum_{k\in K}\sum_{r\in R} y^k_{ir}\right)\\ &\quad +\omega\sum_{i\in C}\left|1-\sum_{k\in K}\sum_{r\in R} y^k_{ir}\right|, \end{aligned} \tag{23}$$

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

$$\text{(PS)} \qquad \Phi^*_{PS}(\omega) = \max_{\lambda}\hat{\Phi}(\omega,\lambda) = \max_{\lambda}\left(\min_{(x,y)\in\Omega}\hat{L}(\omega,\lambda,x,y)\right). \tag{24}$$

The $\Phi^*_{PS}(\omega)$ is the optimum dual solution with penalty weight ω. 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* ω *and* λ, $\hat{\Phi}(\omega,\lambda) \le \Phi^*_{PS}(\omega) \le z^*_{MP} \le J(x,y)$.

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

$$z^*_{MP} = \sum_{k\in K} J_k(x^*,y^*) + \sum_{i\in C} \lambda^*_{\mathrm{i}} \left(1 - \sum_{k\in K}\sum_{r\in R} y^{k*}_{ir}\right) + \omega \sum_{i\in C} \left|1 - \sum_{k\in K}\sum_{r\in R} y^{k*}_{ir}\right| = \hat{L}(\omega,\lambda^*,x^*,y^*).$$

From the definition (24),

$$\Phi^*_{PS}(\omega) = \min_{(x,y)\in\Omega} \hat{L}(\omega,\lambda^*,x,y) \leq \hat{L}(\omega,\lambda^*,x^*,y^*) = z^*_{MP}. \qquad \square$$

$\square$

Revised Lagrangian function (23) cannot be decomposed into k subproblems due to the penalty term. Hence, to calculate the subgradient of $\hat{\Phi}(\omega,\lambda)$ with respect to λ, we now need to solve the integrated low-level problem $\min_{(x,y)\in\Omega} \hat{L}(\omega,\lambda,x,y)$, 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:

$$\hat{L}_k(\omega,\lambda,x,y) = J_k(x,y) - \sum_{i\in C}\sum_{r\in R} \lambda_i y^k_{ir} + \omega \sum_{i\in C} \left| q_k(i) - \sum_{r\in R} y^k_{ir} \right|, \tag{25}$$

where,

$$q_k(i) = 1 - \sum_{\substack{s\in K \\ s\neq k}} \sum_{r\in R} y^s_{ir}. \tag{26}$$

It can be verified that the Lagrangian function (23) can be expressed in terms of $\hat{L}_k(\omega,\lambda,x,y)$ and $q_k(i)$ as follows:

$$\hat{L}(\omega,\lambda,x,y) = \hat{L}_k(\omega,\lambda,x,y) + \sum_{\substack{s\in K \\ s\neq k}} J_s(x,y) + \sum_{i\in C} \lambda_i q_k(i). \tag{27}$$

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:

$$(\mathrm{PS_k}) \quad \hat{\Phi}_k(\lambda) = \min_{(x,y)\in\Omega_{\mathrm{k}}} \hat{L}_k(\omega,\lambda,x,y). \tag{28}$$

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 $\hat{L}(\omega, \lambda, x, y)$ 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,

$$g_i^j = 1 - \sum_{k \in K} \sum_{r \in R} \left(y_{ir}^k\right)^j \qquad \forall i \in C. \tag{29}$$

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., $\hat{\lambda}^0 = \left\{\hat{\lambda}_i^0, \forall i \in C\right\}$

λ^j: Lagrangian multipliers at iteration j, i.e., $\lambda^j = \left\{\lambda_i^j, \forall i \in C\right\}$

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

δ^j: step-size at iteration j

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

β: step-size update parameter, $0 < \beta < 1$

α: initialization factor for Lagrangian multipliers

ϵ: threshold for Lagrangian multiplier convergence criteria, $\epsilon > 0$

SSS Procedure:

Initialization.

I.1. Given $\hat{\lambda}^0$, e.g., $\hat{\lambda}^0 = 0$, solve (LD) using (22) and obtain (x^0, y^0)

I.2. Calculate,

$$\lambda_i^0 = \alpha \left(1 - \sum_{k \in K} \sum_{r \in R} \left(y_{ir}^k \right)^0 \right) \qquad \forall i \in C, \tag{30}$$

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

I.3. Calculate $L_\omega^0 = \hat{L} (\omega, \lambda^0, x^0, y^0)$ and update Lagrangian multipliers:

$$\lambda^1 = \lambda^0 + \delta^0 g^0 \ ,$$

where $0 < \delta^0 = \beta (\Phi_{PS}^* (\omega) - L_\omega^0) / \| g^0 \|^2$ 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 ($\mathrm{PS_k}$) in (28) by setting

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

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

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

$$L_\omega^j = \hat{L} \left(\omega, \lambda^j, x^j, y^j \right) < \hat{L} \left(\omega, \lambda^j, x^{j-1}, y^{j-1} \right), \tag{31}$$

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

Step 3. Check the stopping criteria.

3.1. If $\left\| \lambda^{j+1} - \lambda^j \right\| \leq \epsilon$, 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 α 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 ω, at each iteration i, $L_\omega^j < \Phi_{PS}^*(\omega)$.*

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

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

Further,

$$\begin{aligned}\hat{L}\left(\omega,\lambda^{j},x^{j-1},y^{j-1}\right) &= \hat{L}\left(\omega,\lambda^{j-1},x^{j-1},y^{j-1}\right)+\hat{L}\left(\omega,\lambda^{j},x^{j-1},y^{j-1}\right)-\hat{L}\left(\omega,\lambda^{j-1},x^{j-1},y^{j-1}\right)\\ &= L_{\omega}^{j-1}+\sum_{i\in C}\left(\lambda_{i}^{j}-\lambda_{i}^{j-1}\right)\left(1-\sum_{k\in K}\sum_{r\in R}y_{ir}^{k}\right)=L_{\omega}^{j-1}+\delta^{j-1}\left\|g^{j-1}\right\|^{2}.\end{aligned}$$

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

□

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

Lemma 3. ***(Direction).*** *Let λ^{*} be the optimal multiplier vector, then $\Phi_{\mathrm{PS}}^{*}\left(\omega\right)-L_{\omega}^{j}\leq\left(\lambda^{*}-\lambda^{j}\right)g^{j},\ \forall j$.*

Proof. Proof Based on (23) and (24), we have

$$\begin{aligned}\Phi_{\mathrm{PS}}^{*}\left(\omega\right)=\hat{\Phi}(\omega,\lambda^{*})= &\quad \hat{L}\left(\omega,\lambda^{*},x^{*},y^{*}\right)\leq\hat{L}\left(\omega,\lambda^{*},x^{j},y^{j}\right)=L_{\omega}^{j}+\hat{L}\left(\omega,\lambda^{*},x^{j},y^{j}\right)-L_{\omega}^{j}\\ = &\quad L_{\omega}^{j}+\left(\lambda^{*}-\lambda^{j}\right)g^{j}.\end{aligned}$$

Last step follows from the definition of g^{j} in (29) and Lagrangian function $\hat{L}\left(\omega,\lambda,x,y\right)$ in (23). From Theorem 4, we have $\Phi_{\mathrm{PS}}^{*}\left(\omega\right)-L_{\omega}^{j}>0$, 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; that is,*

$$\left\|\lambda^{*}-\lambda^{j+1}\right\|^{2}<\left\|\lambda^{*}-\lambda^{j}\right\|^{2}\quad\forall j,$$

where λ^{*} is the optimal multiplier vector.

Proof. Proof From (32) we have

$$\begin{aligned}\left\|\lambda^{*}-\lambda^{j+1}\right\|^{2} &= \left\|\lambda^{*}-\lambda^{j}-\delta^{j}g^{j}\right\|^{2}\\ &= \left\|\lambda^{*}-\lambda^{j}\right\|^{2}+\left(\delta^{j}\right)^{2}\left\|g^{j}\right\|^{2}-2\delta^{j}\left(\lambda^{*}-\lambda^{j}\right)g^{j}.\end{aligned}$$

Using result from Lemma 3, we have,

$$\left\|\lambda^* - \lambda^{j+1}\right\|^2 \leq \left\|\lambda^* - \lambda^j\right\|^2 + \left(\delta^j\right)^2 \left\|g^j\right\|^2 - 2\delta^j \left(\Phi^*_{\mathrm{PS}}(\omega) - L^j_\omega\right).$$

Then, from the definition of δ^j in Step 2 of SSS procedure,

$$\left\|\lambda^* - \lambda^{j+1}\right\|^2 \leq \left\|\lambda^* - \lambda^j\right\|^2 - \delta^j \left(\Phi^*_{\mathrm{PS}}(\omega) - L^j_\omega\right),$$

and using the result of Theorem 4, we obtain $\left\|\lambda^* - \lambda^{j+1}\right\|^2 \leq \left\|\lambda^* - \lambda^j\right\|^2$. □

□

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

Theorem 6. *For any two penalty weight ω_1 and ω_2, where $0 < \omega_1 < \omega_2$,*

$$\hat{\Phi}(\omega_1, \lambda) \leq \hat{\Phi}(\omega_2, \lambda) \leq \Phi^*_{\mathrm{PS}}(\omega_2) \leq z^*_{MP}.$$

Proof. Proof From (23), we have $L^j_{\omega_2} - L^j_{\omega_1} = (\omega_2 - \omega_1)\left|\sum_{i \in C} g^j_i\right| \geq 0$.Thus, $L^j_{\omega_2} \geq L^j_{\omega_1}$. Subsequently from (24), we have,

$$\min\ L^j_{\omega_2} \geq \min\ L^j_{\omega_1}\ ,$$

$$\max\ \hat{\Phi}(\omega_2, \lambda) \geq \max\ \hat{\Phi}(\omega_1, \lambda)\ ,$$

$$\Phi^*_{\mathrm{PS}}(\omega_2) \geq \Phi^*_{\mathrm{PS}}(\omega_1).$$

From Theorem 5, we already have $\Phi^*_{\mathrm{PS}}(\omega_2) \leq z^*_{MP}$. □

□

While Theorem 6 states that the solution quality of SSS improves with penalty parameter, we note that choosing ω 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 $\Phi^*_{\mathrm{PS}}(\omega)$ 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 $\Phi^*_{PS}(\omega)$ with a dynamically adjusted estimate Φ^j_{PS} (target value). Analogous to Theorem 3, it can be shown that $L^j_\omega < \Phi^j_{PS}$ holds true for each iteration j. In choosing the estimate Φ^j_{PS}, the goal is to approximate z^*_{MP} as close as possible. In standard VTVM method, the target value Φ^j_{PS} 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 Φ^j_{PS} 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 $\Phi^*_{PS}(\omega)$. 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 $\Phi^*_{PS}(\omega)$ with Φ^j_{PS} in the remainder steps of the SSS procedure.

Notation for SSS with Backtracking VTVM:

Φ^j_{PS}: target value at iteration j

$\overline{\Phi}^j_{PS}$: upper bound on the optimal solution at iteration j

$\underline{\Phi}_{PS}$: lower bound on the optimal solution value

(x^*, y^*) : an optimal solution to (MP)

$\triangle_j$: accumulated improvements since the last Lagrangian function improvement until the beginning of iteration j

ε_j: acceptance tolerance that the current incumbent value L^j_ω is close to the target value Φ^j_{PS} in iteration j

σ: acceptance interval parameter

η_j: fraction of cumulative improvement that is used to increase the target value in iteration j

ϵ_{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 $\Phi_{PS}^{j=1} = \underline{\Phi}_{PS}$, $\overline{\Phi}_{PS}^{j=1} = +\infty$, $\eta_{j=1}$= 0.35, σ= 0.2,$\triangle_{j=1}$= 0, and $\varepsilon_{j=1}$=$\sigma\left(\Phi_{PS}^{j=1} - L_{\omega}^{j=0}\right)$.

Step 1. Subproblem Solution & Backtracking:

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

1.1.a. Solve subproblem ($\mathrm{PS_k}$) in (28) by setting $(x^j, y^j)_s = (x^{j-1}, y^{j-1})_s$ for $s \in K,\ s \neq k$

and obtain $(x^j, y^j)_k$. Denote (x^j, y^j) =$(x^j, y^j)_k \cup \{(x^{j-1}, y^{j-1})_s | s \in K, s \neq k\}$.

1.1.b. If (x^j, y^j) is primal feasible, then

i. Set $(x^*, y^*) = (x^j, y^j)$,

ii. Set $\overline{\Phi}_{PS}^{j} = L_{\omega}^{j} = J\left(x^j, y^j\right)$,

iii. Set algorithm parameters, variables, and solutions back to iteration v, i.e., where $v = max\left\{l : \Phi_{PS}^{l} < \overline{\Phi}_{PS}^{j} = L_{\omega}^{j}\right\}$,

iv. Set $j := v$,

v. Set $\beta = \beta/2$ and repeat iteration j with updated Lagrange multipliers $\lambda^j = \lambda^{j-1} + \delta^{j-1} g^{j-1}$.

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

$$L_{\omega}^{j} = \hat{L}\left(\omega, \lambda^j, x^j, y^j\right) < \hat{L}\left(\omega, \lambda^j, x^{j-1}, y^{j-1}\right),$$

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 & VTVM:

2.1. If $L_{\omega}^{j} > \Phi_{PS}^{j} - \varepsilon_j$, then

2.1.a. Update the target value $\Phi_{PS}^{j+1} = \min\left\{L_{\omega}^{j} + \varepsilon_j + \eta_{\mathrm{j}} \triangle_{\mathrm{j}}, \overline{\Phi}_{PS}^{j}\right\}$,

2.1.b. Update the threshold $\varepsilon_{j+1} = \sigma\left(\Phi_{PS}^{j+1} - L_{\omega}^{j}\right)$,

2.1.c. Reset $\triangle_{\mathrm{j}} = 0$,

2.1.d. Update $\eta_{\mathrm{j}+1} = min\left\{2\eta_{\mathrm{j}}, 1\right\}$,

otherwise set $\Phi_{PS}^{j+1} = \Phi_{PS}^{j}$, $\overline{\Phi}_{PS}^{j+1} = \overline{\Phi}_{PS}^{j}$, $\eta_{\mathrm{j}+1} = \eta_{\mathrm{j}}$ and $\varepsilon_{j+1} = \varepsilon_j$.

2.2. Update $\triangle_{\mathrm{j}+1} = \triangle_{\mathrm{j}} + \left(L_{\omega}^{j} - L_{\omega}^{j-1}\right)$

2.3. Update Lagrangian multipliers:

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

where $0 < \delta^j = \beta\left(\Phi_{PS}^{j} - L_{\omega}^{j}\right) / \left\|g^j\right\|^2$ and $0 < \beta < 1$.

Step 3. Check the stopping criteria:

3.1. If $\left(\overline{\Phi}_{PS}^{j} - \ L_{\omega}^{j}\right) \leq \epsilon_{\text{GAP}}$ or $\left\|\lambda^{j+1} - \lambda^{j}\right\| \leq \epsilon$, 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 Φ_{PS}^{j} with an underestimation $\underline{\Phi}_{PS}$ 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_{ω}^{j} 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 the an iteration v satisfying $\Phi_{PS}^{v} < L_{\omega}^{j}$ and repeat the iteration j with smaller step size. As the L_{ω}^{j} closes in on the target value such that L_{ω}^{j} is within ε_j threshold of Φ_{PS}^{j}, then Step 2.1.a updates the target value based on the accumulated improvement $\triangle_{\text{j}}$ and $\overline{\Phi}_{PS}^{j}$. This update guarantees that the dual solution and the target value is separated by at least ε_j while ensuring that the target value does not exceed the upper bound. The threshold ε_j is updated in Step 2.1.b.

Choosing large values for σ increases ε_j. With higher ε_j values, we are more likely to consider that the L_{ω}^{j} 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 $\overline{\Phi}_{PS}^{j}$ might not have decreased sufficiently. In contrast, lower σ values reduce the convergence rate. The required ranges for acceptance interval parameter and fraction of cumulative improvement are $\sigma \in (0, 1/3]$ and $\eta_{\text{j}} \in (0, 1]$ (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 (ϵ_{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 based 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_r^h of flights are independent and identically distributed according to a uniform distribution $U[\varphi/|K|, \theta]$ where $|K|$ is the number of available vehicles; φ 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_{ir}^h 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.

Description of column headings in Table 2.

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

Comparative performance of CPLEX and SSS.

							SSS-VTVM							SSS-B-VTVM						
			CPLEX Gap (%)				SSS Gap (%)				CPU Time Ratio			SSS Gap (%)				CPU Time Ratio		
|C|	|H|	|K|	Ave	Min	Max	Hit	Ave	Min	Max	Hit	Ave	Min	Max	Ave	Min	Max	Hit	Ave	Min	Max
5	1	3	0.0	0.0	0.0	100	1.8	0.0	5.2	40	1	0	3	0.1	0.0	0.8	70	1	0	1
		4	0.0	0.0	0.0	100	1.3	0.0	5.8	40	1	0	2	0.2	0.0	1.5	70	1	0	1
		5	0.0	0.0	0.0	100	0.7	0.0	2.1	50	1	0	3	0.3	0.0	1.8	80	1	0	2
	2	3	0.0	0.0	0.0	100	0.8	0.0	5.6	40	25	0	146	0.1	0.0	1.2	80	11	0	57
		4	0.0	0.0	0.0	100	1.5	0.0	8.2	50	8	0	28	0.0	0.0	0.2	90	4	0	12
		5	3.3	0.0	33.0	90	2.8	0.0	7.1	20	31	1	228	0.5	0.0	2.0	50	27	0	228
7	1	3	0.0	0.0	0.0	100	1.5	0.0	6.2	20	10	1	28	0.8	0.0	4.0	30	9	0	28
		4	0.0	0.0	0.0	100	0.6	0.0	2.7	50	11	0	38	0.4	0.0	2.7	70	10	0	38
		5	0.0	0.0	0.0	100	1.9	0.1	5.7	0	9	1	34	0.5	0.0	1.8	30	7	0	34
	2	3	0.0	0.0	0.0	100	1.4	0.0	3.3	10	102	6	344	0.9	0.0	3.3	30	91	4	344
		4	1.9	0.0	14.9	80	4.2	0.0	14.9	20	54	4	346	2.9	0.0	14.9	30	50	4	346
		5	1.5	0.0	7.3	78	3.2	0.0	11.6	11	99	18	309	1.8	0.0	6.8	22	73	9	207
10	1	3	0.2	0.0	2.3	82	2.0	0.0	6.7	18	431	7	2,205	0.8	0.0	2.3	27	239	5	966
		4	0.9	0.0	5.2	70	3.7	0.1	7.5	0	446	14	1,549	1.9	0.0	6.3	10	281	14	1,549
		5	0.7	0.0	3.2	70	4.0	0.0	6.9	0	195	0	645	1.4	0.0	4.9	0	143	0	645
	2	3	2.9	0.0	9.6	46	5.0	0.0	15.6	8	172	1	753	3.4	0.0	9.9	23	81	1	252
		4	9.1	1.1	34.4	0	10.8	2.8	19.5	0	234	2	748	7.9	2.3	14.6	0	120	2	255
		5	9.5	0.9	26.1	0	11.1	3.4	20.2	0	299	27	1,581	8.8	2.6	16.1	0	145	25	688
15	1	3	4.5	0.0	35.9	23	6.7	0.3	14.1	0	520	2	1,892	5.2	0.2	14.1	0	313	2	1,382
		4	4.7	0.0	26.6	10	5.1	1.6	15.4	0	198	64	687	3.9	1.6	8.2	0	169	53	687
		5	9.6	0.0	27.6	7	8.8	3.9	20.3	0	289	5	1,048	6.4	0.0	12.8	7	195	2	1,048
	2	3	13.1	2.8	51.4	0	9.5	4.4	16.7	0	43	2	143	8.9	4.4	16.7	0	40	2	143
		4	33.0	4.0	69.1	0	15.2	7.4	25.3	0	62	1	287	12.5	3.4	25.3	0	28	1	113
		5	31.8	0.1	66.4	0	12.5	3.8	20.5	0	72	3	535	11.2	2.0	19.6	0	42	1	244

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

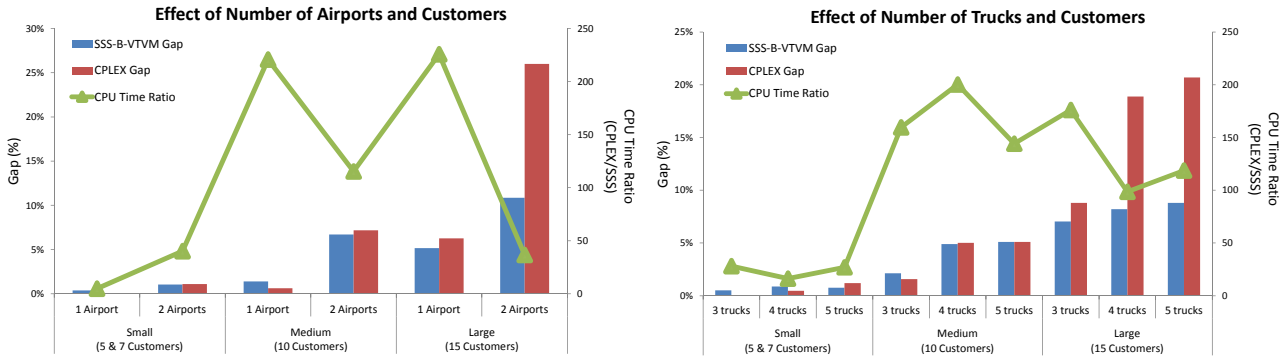
better than that of SSS's, the difference is small. Further, SSS is able to attain good quality solutions up to 346 times faster and on the average 29 times faster.

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.



(a) (b)

Figure 3: Effect of number of customers, (a) number of airports and (b) number of vehicles 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.

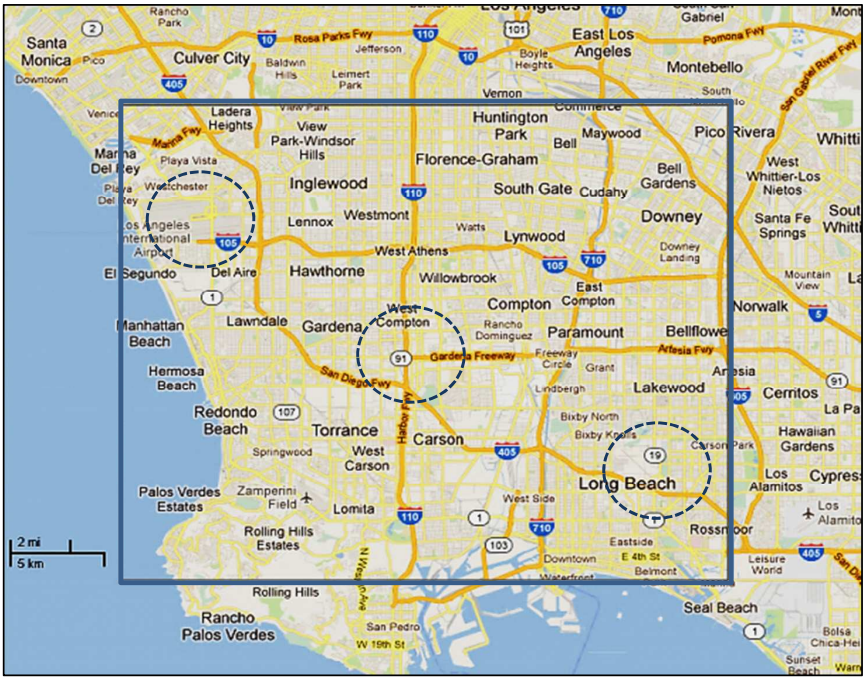


Figure 4: Southern California MAR used in the case study

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.

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.

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

[1]Bureau of Transportation Statistics, U.S. Department of Transportation, Last Accessed November 2011, http://www.transtats.bts.gov/

Table 1: Case study flight itinerary options from LAX and LGB airports.

Dest.	Prob.	Origin	Airline	Departure Time		Mean Delay (min)	Mean Elapsed Time (min)
				1st	2nd		
BOS	19%	LAX	AA	22:15	07:15*	7	324
		LGB	B6	22:59	22:40*	11	324
FLL	13%	LAX	DL	21:05	09:45*	10	324
		LGB	B6	21:15	23:59	10	307
IAD	24%	LAX	AA	21:00	21:00*	11	293
		LGB	B6	21:08	21:08*	23	310
JFK	44%	LAX	AA	21:20	09:05*	10	312
		LGB	B6	21:00	08:05*	17	315

* Next day departure

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 poli-

cies 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.

Table 2: Case study results for three depot location scenarios (DLGB, DMID, DLAX) and three airport access policies (AAAP, LGB, LAX)

	DLGB Depot					**DMID Depot**					**DLAX Depot**				
No	LB	AAAP	LGB	LAX	ρ	LB	AAAP	LGB	LAX	ρ	LB	AAAP	LGB	LAX	ρ
1	6,843	7,012	7,111	7,899	63%	6,800	6,989	7,108	7,362	61%	6,860	7,088	7,137	7,605	82%
2	6,776	6,981	7,181	7,358	51%	6,787	6,979	7,138	7,248	55%	6,799	6,986	7,127	7,404	57%
3	6,823	6,994	7,056	7,817	73%	6,837	7,036	7,115	7,535	72%	6,809	6,923	7,054	7,627	47%
4	6,833	7,011	7,133	7,797	59%	6,821	7,088	7,198	7,474	71%	6,860	7,075	7,107	7,664	87%
5	6,769	6,961	7,177	7,426	47%	6,808	6,999	7,077	7,561	71%	6,833	6,984	7,246	7,604	37%
6	6,765	6,932	7,080	7,501	53%	6,823	6,981	6,997	7,801	91%	6,741	6,904	7,127	7,315	42%
7	6,852	7,061	7,125	7,661	77%	6,746	6,948	7,147	7,251	50%	6,839	7,058	7,164	7,440	67%
8	6,819	7,000	7,172	7,514	51%	6,812	6,932	7,065	7,635	47%	6,781	6,988	7,181	7,461	52%
9	6,763	6,961	7,123	7,390	55%	6,832	7,042	7,158	7,614	64%	6,793	7,013	7,158	7,375	60%
10	6,804	6,996	7,151	7,797	55%	6,749	6,955	7,205	7,299	45%	6,786	6,918	7,153	7,917	36%
Ave.	6,805	6,991	7,131	7,616	58%	6,802	6,995	24%	7,121	63%	6,810	7,075	7,107	7,664	57%
Sdev.	34	35	41	201.5	10%	32	48	2%	63	14%	38	66	50	178	18%

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:

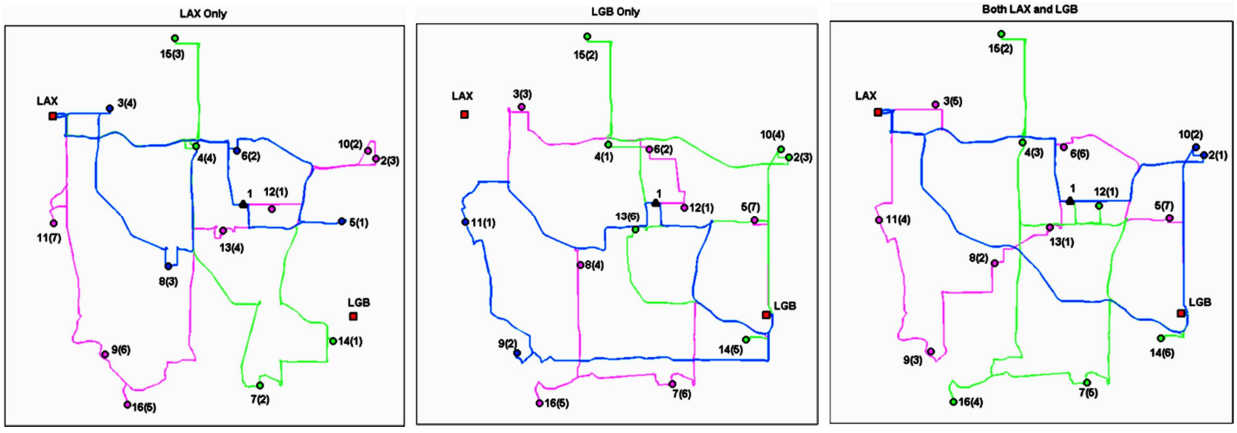
$$\rho = \frac{z_{AAAP} - LB}{\min\{z_{LGB}, z_{LAX}\} - LB}\% \tag{32}$$

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 improve-

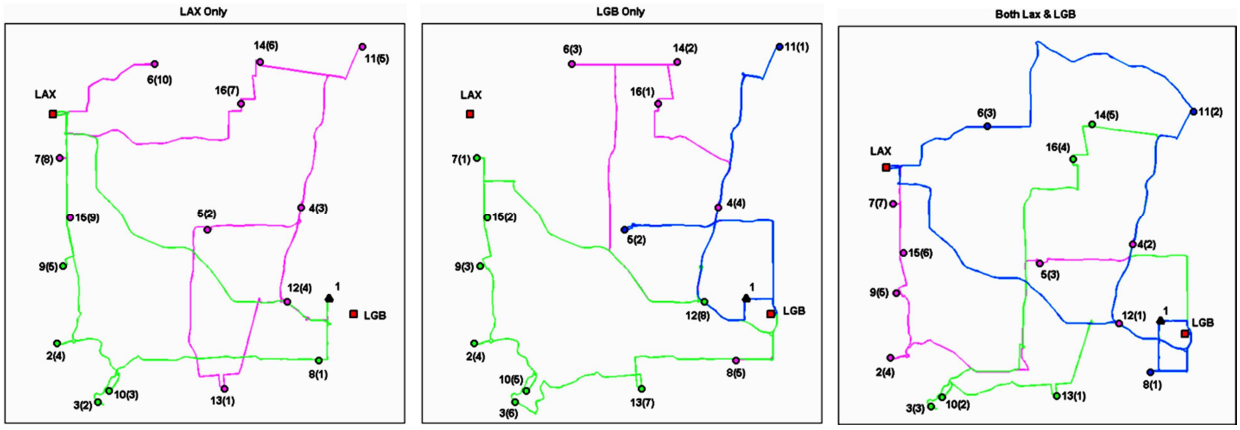
ment of the AAAP policy over single airport policies. In the case of DLGB depot location, the AAAP policy improves the total delivery time performance on the average by 58%. The improvements range between 47% and 77%. Similarly, for the DMID depot location, the average improvement of AAAP is 63% and 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.

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.



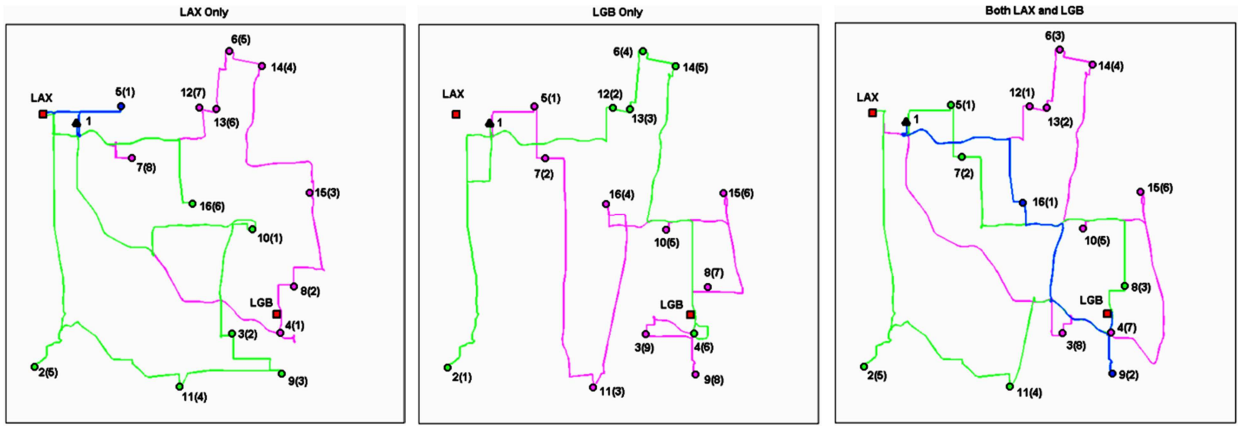
(a) (b) (c)

Figure 5: Routes for problem instance #10 with DMID depot



(a) (b) (c)

Figure 6: Routes for problem instance #1 with DLGB depot



(a) (b) (c)

Figure 7: Routes for problem instance #6 with DLAX depot

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.

The proposed model and solution methodology provide small- and medium-sized freight forwarders an operational decision support tool to improve their customer service levels and reduce transportation cost in road and air modes. This research can be extended in multiple directions. Since the ATD-PDP generalizes the PDP, it thus can be used to study similar problems in other application areas. The heuristic methods, e.g., in Ropke and Pisinger (2006) and Gendreau et al. (1999), can be adapted to solve ATD-PDP. The SSS-B-VTVM

methodology can be applied to a broad range of vehicle routing problems where standard Lagrangian decomposition leads to identical subproblems. Further research can investigate, using the ATD-PDP model, the forwarder's problem of determining the depot locations to best serve customers. Similarly, it can also be used to assess the competitiveness of multiple airports in a MAR for air cargo shipments under various flight itinerary (e.g., frequency, schedule, cost) scenarios.

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:

Airbus, 2010. Global Market Forecast 2010-2029, In: Airbus (Ed.), Toulouse.

Anily, S., Bramel, J., 1999. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. Nav Res Log 46(6), 654-670.

Anily, S., Hassin, R., 1992. The swapping problem. Networks 22(4), 419-433.

Azadian, F., Murat, A.E., Chinnam, R.B., 2012. Dynamic routing of time-sensitive air cargo using real-time information. Transportation Research Part E: Logistics and Transportation Review, 48(1), 355-372.

Başar, G., Bhat, C., 2004. A parameterized consideration set model for airport choice: an application to the San Francisco Bay Area. Transportation Research Part B: Methodological 38(10), 889-904.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15(1), 1-31.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. Eur J Oper Res 202(1), 8-15.

Boeing, 2010. World air cargo forecast 2010-2011, In: Company, T.B. (Ed.), Seattle, WA.

Chalasani, P., Motwani, R., 1999. Approximating capacitated routing and delivery problems. SIAM Journal on Computing 28(6), 2133-2149.

Chayanupatkul, A., Hall, R., Epstein, D., 2004. Freight routing and containerization in a package network that accounts for sortation constraints and costs. METRANS Transportation Center, U. of Southern California.

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

New York.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of Operations Research 153(1), 29-46.

Desrochers, M., Laporte, G., 1991. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. Operations Research Letters 10(1), 27-36.

Diana, M., Dessouky, M.M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B: Methodological 38(6), 539-557.

Fisher, M.L., 2004. The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science 50(12), 1861-1871.

Frederickson, G., 1978. Approximation Algorithms for Some Routing Problems. SIAM J. Comput. 7(2), 178-193.

Gendreau, M., Laporte, G., Vigo, D., 1999. Heuristics for the traveling salesman problem with pickup and delivery. Computers & Operations Research 26(7), 699-714.

Geoffrion, A.M., 1974. Lagrangean relaxation for integer programming, In: Balinski, M.L. (Ed.), Approaches to Integer Programming. Springer Berlin Heidelberg, 82-114.

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

Held, M., Wolfe, P., Crowder, H.P., 1974. Validation of subgradient optimization. Mathematical Programming 6(1), 62-88.

Hellermann, R., 2006. Capacity options for revenue management theory and applications in the air cargo industry. Springer, Berlin; Heidelberg; New York.

Hernández-Pérez, H., Salazar-González, J.-J., 2004a. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics 145(1), 126-139.

Hernández-Pérez, H., Salazar-González, J.-J., 2004b. Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. Transportation Science 38(2), 245-255.

Hernández-Pérez, H., Salazar-González, J.-J., 2007. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. Networks 50(4), 258-272.

Hernández-Pérez, H., Salazar-González, J.-J., 2009. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. Eur J Oper Res 196(3), 987-995.

Kohl, N., Madsen, O.B.G., 1997. An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation. operations research 45(3), 395-406.

Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. Eur J Oper Res 59(3), 345-358.

Laporte, G., 2009. Fifty Years of Vehicle Routing. Transportation Science 43(4), 408-416.

Lim, C., Sherali, H., 2006. Convergence and Computational Analyses for Some Variable Target Value and Subgradient Deflection Methods. Computational Optimization and Applications 34(3), 409-428.

Loo, B.P.Y., 2008. Passengers' Airport Choice within Multi-Airport Regions (MARs): Some Insights from a Stated Preference Survey at the Hong Kong International Airport. Journal of Transport Geography 16(2), 117-125.

Margreta , M., Ford , C., Dipo , M.A., 2009. U.S. Freight on the Move: Highlights from the 2007 Commodity Flow Survey Preliminary Data. US Department of Transportation, Research and Innovative Technology Administration, Bureau of Transportation Statistics

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.

Parragh, S., Doerner, K., Hartl, R., 2008a. A survey on pickup and delivery problems (Part I: Transportation between customers and depot). Journal für Betriebswirtschaft 58(1), 21-51.

Parragh, S., Doerner, K., Hartl, R., 2008b. A survey on pickup and delivery problems (Part II: Transportation between pickup and delivery locations). Journal für Betriebswirtschaft 58(2), 81-117.

Ropke, S., Pissinger, D., 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40(4), 455-472

Solomon, M.M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35(2), 254-265.

Toth, P., Vigo, D., 2001. The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia.

Zhai, Q., Guan, X., Cui, J., 2002. Unit commitment with identical units successive subproblem solving method based on Lagrangian relaxation. Power Systems, IEEE Transactions on 17(4), 1250-1257.

Zhao, X., Luh, P.B., Wang, J., 1999. Surrogate Gradient Algorithm for Lagrangian Relaxation. J Optimiz Theory App 100(3), 699-712.

B. Operational Response Model and Novel Solution Methodology for Managing Disruptions at Inter-modal Facilities and Transportation Networks

1. Introduction

Facility location decisions, as strategic supply chain decisions, require significant investment and forward looking to anticipate and plan for uncertain future events. An important category of such uncertain events is the disruption of facilities which are critical for the ability to efficiently serve the customer demand (Schütz, 2009). These disruptions can be natural disasters or man-made (such as terrorist attacks, labor strikes, etc.). In certain cases, the disruption at a region may extend or migrate through the network and affect other parts of the supply chain network (Masihtehrani, 2011). Recent examples of such disruptions are the 2011 earthquake in Japan affecting Toyota's ability to ship parts and finished vehicles (The Guardian, 2011 & Brennan, 2011), hurricanes Katrina and Rita in 2005 disrupting the nation's oil refineries, and the 2000 fire at the Philips plant halting the production of Ericsson and Nokia (Snyder et al., 2006).

Following a disruption event, there is hardly any recourse action to change the supply chain substructure rapidly (Snyder et al., 2006). Instead, a common recourse is to reassign customers to other facilities or arrange alternative sources of supply. In either case, the cost of serving the customer demand increases e.g., due to higher transportation cost. Over the past decade, the consideration of such disruptions affecting the supply chain network design has received significant attention from both the researchers and practitioners. An exemplary earlier study is Snyder & Daskin (2005) where authors develop a reliability based formulation for Uncapacitated Facility Location Problem (UFLP) and propose a Lagrangean relaxation based algorithm. More recently, Shen et al. (2011) study a variant of the reliable UFLP in Snyder & Daskin (2005) called uncapacitated reliable facility location problem (URFLP). Authors propose highly efficient approximation algorithms for URFLP by exploiting the special structure of the problem. However these approximations cannot be applied to the general class of facility location problems such as the capacitated reliable facility location problems (CRFLP).

In practice, capacity decisions are considered jointly with the location decisions. Further, the capacity of facilities often cannot be changed in the event of a disruption. Following a facility failure, customers can be assigned to other facilities only if these facilities have sufficient available capacity. Thus capacitated reliable facility location problems are more

complex than their uncapacitated counterparts (Shen et al., 2011). The studies considering capacitated reliable facility location problem are limited. Snyder & Ulker (2005) study the CRFLP and propose an algorithm based on Sample Average Approximation (SAA) embedded with Lagrangean relaxation. Gade (2007) apply the SAA method in combination with a dual decomposition method to solve CRFLP. Peng et al. (2011) propose a hybrid metaheuristic based on genetic algorithm to solve a related problem where the objective is to minimize the total fixed and transportation cost while limiting the disruption risk based on the p-robustness criterion. In summary, the earlier work on CRFLP uses either SAA based approximation or metaheuristic methods to overcome the computational complexity associated with large number of scenario realizations.

In this study we develop a novel technique, called Swarm Intelligence Based Sample Average Approximation (SIBSAA), hybridizing the swarm intelligence and SAA method to efficiently solve the CRFLP. While the standard SAA procedure is effective with sufficiently large samples, the required sample size can be quite large for the desired confidence level. Further, the SAA procedure selects the best performing sample solution and discards the remaining sample solutions which contain valuable information about the problem's uncertainty. The main idea of the proposed hybrid method is to re-use all the information embedded in sample solutions by iteratively solving the sample problems while injecting social learning in the solution process. The swarm intelligence injection is based on the Particle Swarm Optimization (PSO). Our experimental results indicate that the proposed hybrid method improves the computational efficiency significantly while attaining same or better solution quality than that of the SAA method.

The rest of this paper is organized as follows. In §2, we review the relevant literature. In §3, we present capacitated reliable facility location problem and its formulation, briefly summarize the SAA and PSO methods, and then describe the proposed hybrid algorithm in detail. In §4, we report on the computational experiments comparing the solution quality and CPU time efficiency of the SAA algorithm and the proposed hybrid algorithm. We conclude with discussion and future research directions in §5.

2. Literature Review

Developing efficient methods for solving large-scale stochastic problems is a challenge in optimization. There are several exact methods such as Progressive Hedging Algorithm (Rockafeller & Wets, 1991), and L-shaped Decomposition (Van Slyke & Wets, 1969) for solving stochastic programming problems. However, these exact methods become impracti-

cal when the number of scenarios is very large. Therefore, we herein focus on two types of approximate methods: sampling based methods and metaheuristics. Sampling methods can be applied in either interior or exterior mode (Linderoth et al., 2006). In the interior mode, the algorithm aims to solve the full problem and only select a sample when an approximate value is needed (Higle and Sen, 1991). In the exterior mode, a sample is randomly selected among all possible scenarios and an approximation of the objective value for the true problem is determined by solving the sample. The SAA method uses exterior sampling and has become a popular technique in solving large-scale stochastic programming problems. This is primarily due to its ease of application. It has been shown that the solutions obtained by the SAA converge to the optimal solution as the sample size is sufficiently large (Ahmed & Shapiro, 2002). However these sample sizes could be quite large and the actual rate of convergence depends on the problem conditioning. Several studies reported successful application of SAA to various stochastic programs (Verweij et al., 2002, Kleywegt et al., 2001, Shapiro & Homem-de-Mello, 1998). The SAA approach is also extensively used in solving stochastic facility location and network design problems (Snyder & Ulker, 2002, Gade, 2007, Santoso et al., 2005, Schütz et al., 2009, Chouinard et al., 2008).

Alternative to sampling methods, the metaheuristic methods such as such as Genetic Algorithms (GA), Tabu Search (TS), and Simulated Annealing (SA) have been used to solve stochastic programming problems. Kratica et al. (2001) applied GA to solve simple facility location problem. Wang et al. (2008) proposed a stochastic programming-based genetic algorithm to determine profitable capacity planning and task allocation plan for resource portfolio planning problem. Authors reported that using a stochastic sampling procedure improves the effectiveness of the genetic algorithm. Arostegui et al. (2006) compared the TS, SA, GA methods' performances by solving various facility location problems under time-limited, solution limited and unrestricted conditions. They reported that the TS method gives better performance overall than both the SA and the GA methods. In this study, we integrate swarm intelligence within the SAA algorithm in an effort to reduce the need to increase sample sizes for improved solution quality. The proposed hybrid method utilizes a swarm intelligence concept similar to that of PSO and enables the swarm learning to take place between the sample solutions of the SAA (i.e.,swarm) . This combined approach differs from the previous studies using metaheuristics to solve stochastic programming problems in two ways. First, the solution methodology is based on SAA where the solutions in the swarm are obtained by exact solution of the sample subproblems (e.g., rather than an update mechanism such as random crossover operation in GA and velocity update in PSO). Second,

the swarm learning is incorporated into the objective function by penalizing deviations from a balanced solution combining the swarm's best solution found thus far and swarm's average solution. Hence, the swarm learning is an integral part of the solution generation process in the proposed SIBSAA method.

3. Problem Statement and Methodology

In this section, we first present Capacitated Reliable Facility Location Problem (CRFLP) formulation. Next, we summarize the standard SAA and PSO heuristic methods and describe the proposed hybrid SIBSAA methodology in detail.

3.1. ***Capacitated Reliable Facility Location Problem(CRFLP)***

We now introduce the notation used throughout this paper. Let D denote the set of customers (i.e., demand points) and F denote the set of possible facility sites. The fixed cost for facility $i \in F$ is denoted by f_i and incurred if the facility is opened. Let d_j be the demand for customer $j \in D$ and c_{ij} denote the cost of satisfying each unit demand of customer j from facility i and include such variable cost drivers as transportation and production costs. Each facility i has a limited capacity and can serve at most b_i units of demand. Facilities are subject to failure and can become unavailable in the event of a disruption. A customer j cannot be served by any of the facilities whenever all facilities fail, there is not sufficient capacity among the surviving facilities to meet customer j's demand, or the cost of serving customer j's demand via surviving facilities is prohibitive. In such cases, the customer j is assigned to an emergency facility and a large penalty cost is incurred for each unit of unsatisfied demand. The emergency facility can represent an alternative supply source and the large penalty cost then represents the outsourcing cost. In the absence of an alternative supply, the emergency facility corresponds to the lost-sale with the penalty cost of forfeited profit. For simplicity, we denote the last facility in F as the emergency facility and the cost $c_{|F|j}$ as the customer j's unit demand penalty cost.

We formulate the CRFLP as a two-stage stochastic programming problem. In the first stage, the location decisions are made before random failures of the located facilities. In the second stage, following the facility failures, the customer-facility assignment decisions are made for every customer given the facilities that have survived. The goal is to identify the set of facilities to be opened while minimizing the total cost of open facilities and the expected cost of meeting demand of customers from the surviving facilities and the emergency facility. In the scenario based formulation of CRFLP, let s denote a failure scenario and the set of

all failure scenarios is S, where $s \in S$. Let p_s be the probability that scenario s occurs and $\sum_{s \in S} p_s = 1$. Further let k_i^s denote whether the facility i survives (i.e., $k_i^s = 1$) and $k_i^s = 0$ otherwise. For instance, in case of independent facility failures, we have $|S| = 2^{|F|-1}$ possible failure scenarios for $|F| - 1$ facilities and the last facility is the emergency facility which is perfectly reliable. Note that our proposed method does not require any assumption on independence and distribution of each facility's failure.

The binary decision variable x_i specifies whether facility i is opened or not, and the binary variable y_{ij}^s specifies whether customer j assigned to facility i in scenario s or not. We note that the single sourcing assumption, while a preferred method in practice, is not restrictive for the proposed method. The scenario based formulation of the CRFLP as a two stage stochastic program is as follows.

CRFLP:

$$\text{Minimize} \qquad \sum_{i \in F} f_i x_i + \sum_{s \in S} p_s \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^s \tag{1}$$

Subject to

$$\sum_{i \in F} y_{ij}^s = 1 \qquad \forall j \in D, s \in S \tag{2}$$

$$y_{ij}^s \leq x \qquad \forall j \in D,\ i \in F,\ s \in S \tag{3}$$

$$\sum_{j \in D} d_j y_{ij}^s \leq k_i^s b_i \qquad \forall i \in F, s \in S \tag{4}$$

$$x_i, y_{ij}^s \in \{0, 1\} \qquad \forall j \in D,\ i \in F,\ s \in S \tag{5}$$

The objective function in (1) minimizes the total fixed cost of opening facilities and the expected second stage cost of satisfying customer demand through lasting and emergency facility. Constraints (2) ensure that each customer is assigned to either an open facility or the emergency facility in every failure scenario. Constraints (3) ensure that a customer's demand cannot be served from a facility that is not opened in every failure scenario. Constraints (4) prevent the assignment of any customer to a facility if it has that is failed and also ensure the total demand assigned the facility does not exceed its capacity in every failure scenario. Constraints (5) are integrality constraints.

As stated in the literature review, the main disadvantage of scenario based formulations is that the number of scenarios grows exponentially with number of facilities. Since enumerating all scenarios is not practical, approximation algorithms are often employed (Snyder et

al., 2006). In the next subsection, we discuss two approximation methods (SAA and PSO) and describe the hybrid SIBSAA methodology.

3.2. ***Swarm Intelligence based SAA Method***

The hybrid SIBSAA algorithm integrates a swarm intelligence strategy inspired by PSO with the classical SAA method in order to improve the solution quality and efficiency. For completeness of the paper, we briefly review the SAA and PSO algorithms in the next two subjections.

3.2.1. ***Sample Average Approximation (SAA)***

The SAA method is commonly used method to solve large-scale stochastic optimization problems (Ahmed & Shapiro, 2002, Kleywegt et al., 2001, Schütz et al., 2009). The main idea of the SAA is to approximate the objective function value of the stochastic program by solving the problem for a sample of scenarios. After solving the problem for a sufficient number of samples, the first-stage solutions of each sample is tested in a large sample by solving for only the second-stage decisions. The best performing sample solution is then selected as the final solution. In our computational experiments, we compare the performance of the proposed hybrid method with that of SAA. The steps of the SAA procedure to solve the CRFLP are as follows.

SAA Procedure for CRFLP

Initialize: Generate M independent random samples $m = 1, 2, \ldots, M$ with scenario sets N_m where $|N_m| = N$. Each sample m consists of N realizations of independently and identically distributed (i.i.d.) random scenarios. We also select a reference sample which is sufficiently large, e.g., $|N'| \gg N$.

Step 1: For each sample m, solve the following problem and record the sample optimal objective function value v^m and the sample optimal solution $\mathbf{x^m}$=$\{x_i^m\}_{\forall i \in F}$.

SAA-CRFLP (m):

$$\text{Minimize} \qquad \sum_{i \in F} f_i x_i^m + \sum_{s \in N_m} \frac{1}{|N_m|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{sm} \tag{6}$$

Subject to

$$\sum_{i \in F} y_{ij}^{sm} = 1 \qquad \forall j \in D, s \in N_m \tag{7}$$

$$y_{ij}^{sm} \leq x_i^m \qquad \forall j \in D,\ i \in F,\ s \in N_m \tag{8}$$

$$\sum_{j \in D} d_j y_{ij}^{sm} \leq k_i^s b_i \qquad \forall i \in F, s \in N_m \tag{9}$$

$$x_i^m, y_{ij}^{sm} \in \{0,1\} \qquad \forall j \in D,\ i \in F, s \in N_m \tag{10}$$

Step 2: Calculate the average $\overline{v}^M$ of the sample optimal objective function values obtained in Step 1 as follows.

$$\overline{v}^M = \frac{1}{M} \sum_{m=1}^{M} v^m \tag{11}$$

Step 3: Estimate the true objective function value $\hat{v}^m$ of the original problem for each sample's optimal solution. Solve the following problem for each sample using the optimal first stage decisions $\mathbf{x^m}$ from Step 1.

$$\hat{v}^m = \text{Minimize} \sum_{i \in F} f_i x_i^m + \sum_{s=1}^{|N'|} \frac{1}{|N'|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{sm} \tag{12}$$

Subject to constraint sets (2), (3), (4), (5), and using $N_m \equiv N'$.
Step 4: Select the solution $\mathbf{x^m}$ with the best $\hat{v}^m$, i.e. $\mathbf{x}^{SAA} = argmin_{m=1,\ldots,M}\ \hat{v}^m$ as the solution and $v^{SAA} = \min_{m=1,\ldots,M} \hat{v}^m$, as the solution value of SAA.

Let v^* denote the optimal objective function value of the original problem CRFLP. The $\overline{v}^M$ is an unbiased estimator of $\mathbb{E}[v]$ which is the expected optimal objective function value of sample problems. Since $\mathbb{E}[v] \leq v^*$, the $\overline{v}^M$ provides a statistical lower bound on the v^* (Ahmed & Shapiro, 2002). While $\overline{v}^M$ does not always provide a lower bound on the v^*, as observed in Shen et al. (2011), it is useful in assessing the quality of the solution value of SAA $\hat{v}^{SAA}$. The reference set N' is used to estimate the objective function value of the sample problem solutions in the original problem CRFLP.

3.2.2. ***Particle Swarm Optimization (PSO)***

The PSO methodology, first proposed by Kennedy and Eberhart (1995), is a population-based metaheuristic search technique motivated by the social behavior of organisms such as bird flocking and fish schooling. It is based on the swarm intelligence idea where knowledge is optimized by social interaction and the thinking is not only personal but also social. The PSO achieves local and global search capabilities where the intensification and diversification are achieved via relative weighting of the personal and social (swarm-based) thinking. PSO

has been successfully applied in solving many combinatorial optimization problems in which the objective space possesses many local optimal solutions. In such problems where there are many multiple local optima, the PSO provides the advantage of escaping from the local optima through the communication between the members of the swarm.

In PSO, each solution is represented as a particle in a swarm. Each particle has a position and a fitness value that is evaluated by the fitness function to be optimized. The particles iterate from one position to another through a velocity vector. This velocity vector is determined by the particle's most recent displacement, its best position thus far, and the best position encountered by all particles in the swarm. The velocity vector of each particle is calculated by updating the previous velocity by two best values: *pbest* and *gbest*. The personal best (*pbest*) is the particle's best position it has visited thus far and tracked by each particle. The global best (*gbest*), tracked by the swarm, is the best position visited by any particle in the population. The influence of the personal and global bests on the velocity vector is controlled by weights called learning factors.

Let L be the set of particles, $l = 1, \ldots, L$ in swarm. In every iteration of the PSO, the particle l's velocity at iteration k, $u_l(k)$, is updated according to :

$$u_l(k+1) = \gamma u_l(k) + \delta_1 r_1 [\theta_l(k) - x_l(k)] + \delta_2 r_2 [\mu(k) - x_l(t)] \tag{13}$$

where $x_l(k)$ is the position (solution) of particle l at iteration k, and the parameters γ and (δ_1, δ_2) are inertia weight and learning factors controlled by the user, respectively. The weights r_1 and r_2 are uniform random numbers generated at every velocity update. The value $\theta_l(k)$ is the individual best solution for particle l at iteration k, and $\mu(k)$ is the swarm's global best solution at iteration k. Using this velocity, the particle's position is updated as $x_l(k+1) = x_l(k) \oplus u_l(k+1)$ where the operator $\oplus$ represents the update scheme depending on whether x_l is continuous or discrete (Kennedy & Eberhart, 1997).

The three terms in (13) represent the memory, cognitive learning and social learning of the particle, respectively. The $\gamma u_l(k)$ is called the particle's inertia which induce the particle to move in the same direction as the most recent displacement. The weights (δ_1, δ_2) are referred as the learning rates since the inertia weight controls the extent to which the memory of the previous velocity influences the new velocity. The diversification and intensification of the particle is controlled through the inertia weight as well as velocity bounds which limit velocity to preset maximum and minimum levels (Shi & Eberhart, 1998). Inertia weight, velocity bounds and learning rates jointly determine the particle's motion. Usually, a high inertia weight is used at the beginning and then gradually decreased so as to diversify the

solution.

3.2.3. ***Hybrid Method: Swarm Intelligence based SAA (SIBSAA)***

The proposed method is a hybridization of the SAA and PSO. The motivation for this hybridization originates from the last stage of the SAA method (Step 4) where the best performing solution is selected and the rest of the solutions are discarded. However, this discarding of $(M-1)$ sample solutions is a loss of valuable sample information as well as loss of effort spent in solving each sample's solution. Let's consider the implementation of the classical SAA procedure in the context of PSO and treat each sample solution as a particle. Then the implementation of SAA would correspond to iterating the particles (sample solutions) in the swarm (the set of sample solutions) only once and then selecting the best performing particle. In the PSO, however, the particle iterations are sustained with the particle's recent memory (e.g., inertia) as well as with the social learning. Hence, in the proposed hybrid approach, we modify the SAA method by continuing the solution of the sample problems (e.g. particles) while injecting the social learning in the solution process. The underlying premise of this hybridization is that, by starting with sufficient number of samples (representative of the entire scenario set), the continued iteration of the SAA method with social learning would converge the sample solutions to the optimal solution of the original problem.

An important distinction of the proposed hybrid method from the classical PSO is the movement of particles. The PSO iterates the particles according to a velocity vector and the positions of particles are suboptimal solutions (until convergence to the global optimum). In comparison, the SIBSAA solves the sample problems to optimality. However, since the samples do not correspond to the entire scenario set, these optimal solutions are sub-optimal for the original problem. As a result of this distinction, the injection of the social learning in particle's movement is different in the SIBSAA than that in PSO. The social learning in SIBSAA is achieved through dynamically penalizing the deviation of the sample solution from the swarm's balanced solution ($\overline{\overline{\mathbf{x}}}^k$) at iteration k. This balanced solution is composed of swarm's best incumbent solution and swarm's average solution at iteration k. The swarm's best incumbent solution ($\mathbf{x}_{best}$) is the solution with the best objective value ($\hat{v}_{best}$) considering the reference set N'. On the other hand, the swarm's average solution ($\overline{\mathbf{x}}^k$) at iteration k is the probability weighted average of sample solutions at iteration $(k-1)$. Similar to the PSO, the social learning is modulated with parameters ρ^k, ω_m^k, and α^k. The parameter ρ^k is the swarm learning parameter which penalizes the sample solutions' squared Euclidean norm deviation from the swarm's balanced solution at iteration k. The parameter ω_m^k penalizes

the linear deviation of the sample solutions from the swarm's balanced solution at iteration k. Lastly, the parameter α^k is a balancing weight between the swarm's average solution and the swarm's best incumbent solution, and used to calculate the swarm's balanced solution.

We first present the proposed SIBSAA algorithm and then describe its steps in detail. For brevity, we use the same notation as before and only introduce the additional notation used in the SIBSAA algorithm.

Notation:
k, k_{max} : iteration index and maximum number of iterations
$P_m, \hat{P}_m$: probability and normalized probability of realization of sample m
$\mathbf{x}^{m,k}$: solution vector for sample m at iteration k
$\overline{\mathbf{x}}^k$: swarm's average sample solution at iteration k
$\overline{\overline{\mathbf{x}}}^k$: swarm's balanced solution at iteration k
$\mathbf{x}_{best}$: best incumbent solution
$\hat{v}_{best}$: objective function value of the best incumbent solution with respect to N'
$\hat{v}^k_{best}$: objective function value of the best solution at iteration k with respect to N'
ω^k_m : dual variable vector for sample m at iteration k
ρ^k : swarm learning parameter at iteration k
β : update factor for the swarm learning parameter
α^k : weight for learning from the global best at iteration k
$\triangle_\alpha$: update parameter for global learning weight
ϵ_k : Euclidean norm distance of sample solutions from the $\overline{\overline{\mathbf{x}}}^k$ at iteration $k-1$
ε : convergence threshold for solution spread
$\mathbf{x}^{\mathbf{S}IBSAA}$: best solution found by SIBSAA
v^{SIBSAA}: objective function value of the best solution found by SIBSAA

The pseudo-code for the swarm intelligence based SAA is as follows:

Swarm Intelligence based SAA Algorithm (SIBSAA) for CRFLP
Initialize: Generate M independent random samples $m = 1, 2, \ldots, M$ with scenario sets N_m where $|N_m| = N$. Each sample m consists of N realizations of independently and identically distributed (i.i.d.) random scenarios. We also select a reference sample which is sufficiently large, e.g., $|N'| \gg N$.

$\circ$ Set $k := 0$, $\omega^{k=0}_m = 0$ for $\forall m = 1, \ldots, M$, $\alpha^{k=0}$, and $\rho^{k=0} := 1$,

∘ Calculate $P_m := \prod_{s \in N_m} p_s$ and $\hat{P}_m = \frac{P_m}{\sum_{m=1}^{M} P_m}$ and denote $\widehat{\mathbf{P}} = \left\{ \hat{P}_m \right\}_{\forall m}$.

Step 1: Execute the Steps 2-4 of the SAA Algorithm for CRFLP using the scenario sets N_m for $m = 1, 2, \ldots, M$ and the reference set N'.

1.1. Assign the solutions $\mathbf{x}_{best} := \mathbf{x}^{SAA}$ and $\mathbf{x}^{m,k=0} = \mathbf{x}^m$ for $\forall m = 1, \ldots, M$.

Step 2: Update the problem parameters as follows,

2.1. Set $k := k + 1$,

2.2. Calculate $\overline{\mathbf{x}}^k := \widehat{\mathbf{P}} \mathbf{x}^{m,k-1}$,

2.3. Calculate $\overline{\overline{\mathbf{x}}}^k := \alpha^k \overline{\mathbf{x}}^k + \left(1 - \alpha^k\right) \mathbf{x}_{best}$,

2.4. Update $\alpha^k := \alpha^{k-1} - \triangle_\alpha$,

2.5. Update ρ^k if $k > 2$,

$$\rho^k := \begin{cases} \beta \rho^{k-1} & if \;\; \epsilon_k > \epsilon_{k-1}/2 \\ \rho^{k-1} & otherwise \end{cases}$$

2.6. Calculate $\omega_m^k := \omega_m^{k-1} + \rho^k \left(\mathbf{x}^{m,k-1} - \overline{\overline{\mathbf{x}}}^k \right)$.

Step 3: For each sample $m = 1, \ldots, M$, solve the following problem and record the sample optimal objective function value $v^{m,k}$ and the sample optimal solution $\mathbf{x}^{m,k} = \left\{ x_i^{m,k} \right\}_{\forall i \in F}$.

SIBSAA-CRFLP(m):

$$\text{Minimize} \quad \sum_{i \in F} f_i x_i^{m,k} + \sum_{s \in N_m} \frac{1}{|N_m|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{smk} + \omega_m^k \mathbf{x}^{m,k} + \frac{\rho^k}{2} \left\| \mathbf{x}^{m,k} - \overline{\overline{\mathbf{x}}}^k \right\|^2 \qquad (14)$$

Subject to constraints (7), (8), (9), and (10). Update $\epsilon_k := \left(\sum_{m=1}^{M} \left\| \mathbf{x}^{m,k} - \overline{\overline{\mathbf{x}}}^k \right\| \right)^{1/2}$.

Step 4: Using the sample solutions $\mathbf{x}^{m,k}$ for $m = 1, \ldots, M$ obtained in Step 3, estimate the true objective function value $\hat{v}^m$ of the original problem by solving the following problem for each sample.

$$\hat{v}^{m,k} = \text{Minimize} \sum_{i \in F} f_i x_i^m + \sum_{s=1}^{|N'|} \frac{1}{|N'|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{smk} \qquad (15)$$

Subject to constraints (7), (8), (9), and (10) using $N_m \equiv N'$.

Step 5: Select the solution $\mathbf{x}^{m,k}$ with the best $\hat{v}^{m,k}$. Let $\hat{v}_{best}^k = argmin_{m=1,\ldots,M} \; \hat{v}^{m,k}$, then

$$\hat{v}_{best} := \begin{cases} \hat{v}_{best}^k & if \;\; \hat{v}_{best}^k < \hat{v}_{best} \\ \hat{v}_{best} & otherwise \end{cases}$$

$$\mathbf{x}_{best} := \begin{cases} \mathbf{x}^{m',k} | m' = argmin_{m=1,\ldots,M}\ \hat{v}^{m,k} & if\ \ \hat{v}^{k}_{best} < \hat{v}_{best} \\ \mathbf{x}_{best} & otherwise \end{cases}$$

Step 6: Check for convergence: If ($\epsilon_k \geq \varepsilon$ or $\overline{\overline{\mathbf{x}}}^k \neq \mathbf{x}_{best}$) and ($k < k_{max}$), then return to Step 2, otherwise terminate with best found solution $\mathbf{x}^{SIBSAA}$=$\mathbf{x}_{best}$ and solution value $v^{SIBSAA} = \hat{v}_{best}$.

The initialization step of the SIBSAA is similar to that of SAA's and the only additional calculation is the sample m's probability and normalized probabilities, e.g., P_m and $\hat{P}_m$. The probability $\hat{P}_m$ is used to calculate the swarm's average sample solution $\overline{\mathbf{x}}^k$ at iteration k (Step 2.2). The first step in SIBSAA is to execute the standard SAA procedure (Step 1). Next, we calculate the swarm's average solution and the balanced solution. The swarm's balanced solution ($\overline{\overline{\mathbf{x}}}^k$) is a weighted average of the average solution ($\overline{\mathbf{x}}^k$) and the incumbent best solution ($\mathbf{x}_{best}$) as calculated in Step 2.3. The weight factor $\alpha^k \in [0, 1]$ used in Step 2.3 determines the bias of the social learning; whereas high values tend the sample solutions to the sample average solution, low values tend to the incumbent best solution. There are two strategies for modulating the social learning bias; α^k can be static by setting $\triangle_\alpha = 0$ or can be dynamically varied between iterations by setting $\triangle_\alpha > 0$ (see Step 2.4). The advantage of dynamic α^k is that, beginning with a large α^k, we first prioritize the sample average solution until the incumbent best solution quality improves. This approach allows guiding the sample solutions to a consensus sample average initially and then directing the consensus sample average in the direction of improved best incumbent solution.

In Step 2.5, we update the swarm learning parameter ρ^k depending whether the distance (ϵ_k) of sample solutions from the most recent balanced solution has sufficiently improved. We choose the improvement threshold as half of the distance in the previous iteration (e.g., ϵ_{k-1}). Similarly, in Step 2.6, we update the penalty parameter (ω^k_m) for the linear deviation of every sample's solution from the swarm's balanced solution at iteration k. Note that the ω^k_m are the Lagrange multipliers corresponding to the equivalence of each sample's solution to the balanced solution.

In Step 3, we solve each sample problem with additional objective function terms representing the social learning and calculate the deviation of the sample solutions from the balanced solution (i.e., ϵ_k). Step 4 estimates the objective function value of each sample solution in the original problem using the reference set N'. Step 5 identifies the sample

solution $\mathbf{x}^{m,k}$ with the best $\hat{v}^{m,k}$ in iteration k and updates the incumbent best $\hat{v}_{best}$ if there is improvement. The Steps 4 and 5 correspond to the integration of SAA method's selection of the best performing sample solution. In contrast to SAA's termination with the best performing sample solution, the proposed SIBSAA retains this information to induce social learning in the next iteration through the balanced solution. Step 6 checks whether the stopping conditions are met. If the iteration limit is reached $k \geq k_{max}$ or when the all sample solutions converged to the balanced solution within a tolerance, then the SIBSAA terminates with the best found solution. The worst-case solution of the SIBSAA is equivalent to the SAA solution with the same set of samples. This can be observed by noting that the best incumbent solution is initialized with the SAA's solution. Hence, the SIBSAA converges to a solution which has the same performance or better than that of SAA's.

4. Experimental Study

We now describe the experimental study performed to investigate the computational and solution quality performance of the proposed SIBSAA for solving CRFLP. We benchmark the results of SIBSAA with those of SAA and an exact solution. We solved the CRFLP exactly by using the deterministic equivalent formulation. All algorithms are programmed in Matlab R2010b and integer programs are solved with CPLEX 12.1. The experiments are conducted on a PC with Intel(R) Core 2 CPU, 2.13 GHz processor and 2.0 GB RAM running on Windows 7 OS. Next, we describe the experimental setting and data in detail. In Section 4.2, we report on sensitivity analysis results of SIBSAA's performance with respect to algorithm's parameters. In Section 4.3, we present and discuss the benchmarking results.

4.1. ***Experimental Setting***

We used the test data sets from Zhan (2007) that are also used in Shen et al. (2011) for the URFLP. In these data sets, the coordinates of site locations are i.i.d. and sampled from $\mathrm{U}[0,1] \times \mathrm{U}[0,1]$. The customer and facility sites are identical. The customer demand is i.i.d, sampled from $\mathrm{U}[0,1000]$, and rounded to the nearest integer. The fixed cost of opening a facility is i.i.d. and sampled from $\mathrm{U}[500,1500]$, and rounded to the nearest integer. The variable costs c_{ij} for $i = 1,\ldots,|F|-1$ and $\forall j$ are chosen as the Euclidean distance between sites. The penalty cost $c_{|F|j}$ for serving customer j from the emergency facility is i.i.d. and sampled from $\mathrm{U}[0,15]$. Since Zhan (2007) and Shen et al. (2011) consider uncapacitated RFLP, the data sets do not have facility capacities. We have selected identical capacity levels for all facilities $b_{i=1,..,|F|} = 2,000$. In generating the failure scenarios,

we assume that the facility failures are independently and identically distributed according to the Bernoulli distribution with probability q_i, i.e., the failure probability of facility i. In our experiments, we use uniform failure probability, i.e., $q_{i=1,\ldots,|F|-1} = q$, and consider the cases $q = \{0.1,\ 0.2, \ldots, 0.9\}$. The emergency facility is perfectly reliable, i.e., $q_{|F|} = 1$. Note that the case $q = 0$ corresponds to the deterministic fixed-charge facility location problem, and $q = 1$ corresponds to the case where all facilities fail. The solution of the latter one is where all customers are assigned to the emergency facility and the objective function value is then $\sum_{j\in D} d_j c_{|F|j}$. The failure scenarios $s \in S$ are generated as follows. Let's denote $F_f^s \subset F$ be the facilities that are failed, and $F_r^s \equiv F \backslash F_f^s$ be the set of facilities that are reliable (not failed) in scenario s. The facility indicator parameter in scenario s become k_i^s=1 if $i \in F_r^s$, and k_i^s=0 otherwise. The probability of scenario s is then calculated as $p_s = q^{|F_f^s|}(1-q)^{|F_f^s|-1}$.

In all experiments, we used $|D| = |F| - 1 = 10$ sites which is a medium-sized CRFLP problem and is more difficult to solve than the uncapacitated version (URFLP). The size of the failure scenario set is $|S|$=1,024. The deterministic equivalent formulation has variables x_i and y_{ij}^s totaling $|F| + |F| \times |D| \times |S| = 11 + 11 \times 10 \times 1024 = 112,651$ binary variables. Similarly, it has constraints (7), (8) and (9) totaling $|D|\times|S|+|F|\times|D|\times|S|+|F|\times|S| = 11+ 11 \times 10 \times 1024 = 134,144$ constraints. We generated sample sets for SAA and the proposed SIBSAA by randomly sampling from U $[0, 1]$ as follows. Given the scenario probabilities, p_s, we calculate the scenario cumulative probability vector $\{p_1, (p_1 + p_2), \ldots, (\ p_1 + p_2 + \ldots + p_{|S|-1}),\ 1\}$ which has $|S|$ intervals. We first generate the random number and then select the scenario corresponding to the interval containing the random number. We tested the SAA and SIBSAA algorithms with varying number of samples (M), and sample sizes (N). We used identical sample sets in all the SAA and SIBSAA comparisons where M and N are same. In selecting the reference set, we use the entire scenario set, i.e., $N' = S$, as the evaluation of a solution with $|S|$=1,024 scenarios can be easily computed. Lastly, all the datasets used in the following sections, including sample sets, are available from the authors upon request.

4.2. ***SIBSAA Parameter Sensitivity***

We evaluated the sensitivity of the SIBSAA with respect to the social learning bias parameter (α), number of samples (M), and the swarm learning parameter update factor (β). First, we tested for α which determines the bias of the social learning, e.g., biased towards swarm's best or swarm's average sample solution at iteration k. In these tests, we set $(M, N) = (5, 10)$, $q = 0.4$, and $\beta = 2$ unless otherwise is stated. Table 1 the first column shows the type of

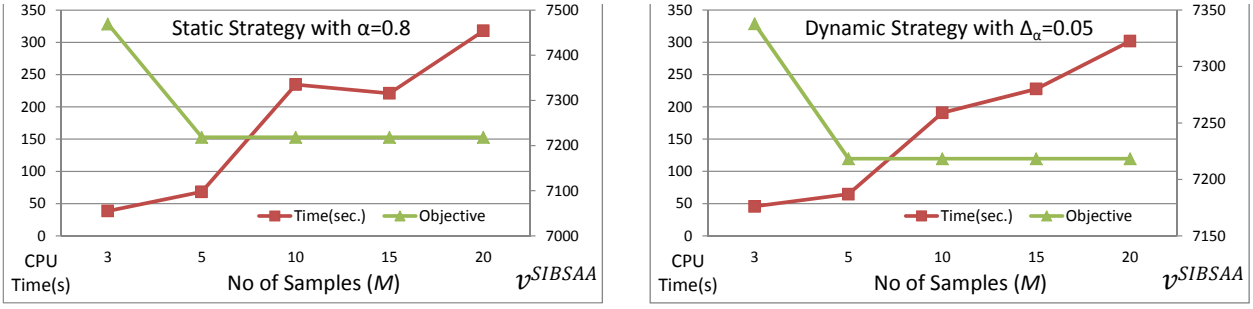
α				
Strategy	Parameter	Open Facilities	Objective	Time(sec.)
Dynamic	$\triangle_\alpha$=0.03	1,2,7,8,10	7,338	69.2
	$\triangle_\alpha$=**0.05**	**1,2,4,8,10**	**7,218**	**64.8**
	$\triangle_\alpha$=0.08	1,2,7,8,10	7,338	64.7
	$\triangle_\alpha$=0.10	1,2,7,8,10	7,338	61.1
	$\triangle_\alpha$=0.12	1,2,7,8,10	7,338	62.2
	$\triangle_\alpha$=0.15	1,2,7,8,10	7,338	59.2
Static	α=0.5	1,2,7,8,10	7,338	69.8
	α=**0.6**	**1,2,4,8,10**	**7,218**	**73.6**
	α=0.7	1,2,7,8,10	7,338	61.3
	α=**0.8**	**1,2,4,8,10**	**7,218**	**68.3**
	α=0.9	1,2,7,8,10	7,338	69.8
	α=1	1,2,7,8,10	7,338	72.5
Exact Solution		**1,2,4,8,10**	**7,218**	$\gg$**21,600**

Table 1. Sensitivity of SIBSAA to social learning bias parameter (α) strategy and settings

social learning strategy, i.e., static and dynamic bias. The second column is the parameter of the corresponding strategy, i.e., $\triangle_\alpha$ for dynamic and α for static. Note that in the dynamic strategy, we select initial value as $\alpha^{k=0} = 1$. The first stage solutions and corresponding objective function value is shown in columns third and fourth. Last column presents the CPU time in terms of seconds. The exact solution is shown at the bottom row which took longer than six hours.

First observation is that the SIBSAA is relatively insensitive to the strategy employed and the parameter settings. In dynamic case, $\triangle_\alpha = 0.05$ identifies the optimal solution, and the remainder parameter settings identify a similar solution except the facility 7 is opened in place of facility 4. Further, as the $\triangle_\alpha$ increases, the swarm's best incumbent solution becomes increasingly more important leading to decreased computational time. Static strategy, similarly, identifies the exact solution when $\alpha = 0.6$ and $\alpha = 0.8$ and finds the same near-optimal solution in other settings. The CPU time with static strategy is variable and slightly higher than the dynamic strategy on the average. In comparison, the dynamic strategy converges to a solution faster than the static strategy, e.g., average of 63.5 versus 69.2 seconds in Table 1, respectively. These results show that the SIBSAA's sensitivity to the α parameter strategy and settings is nominal.

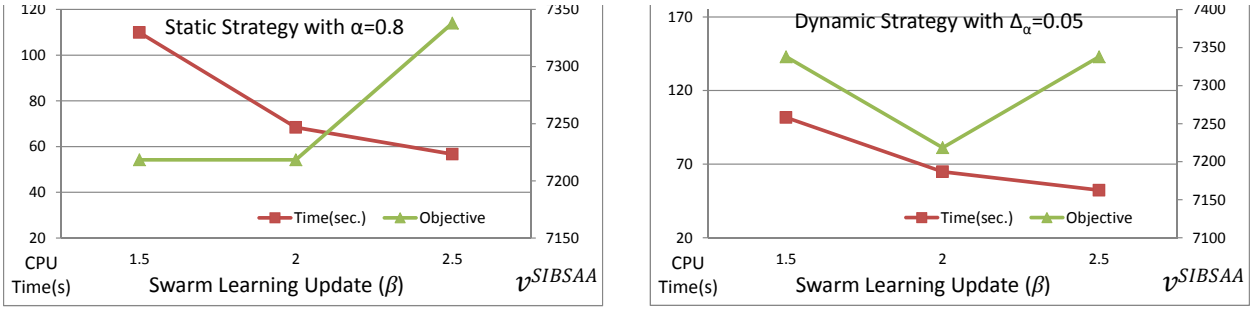
Next, we tested the SIBSAA's sensitivity with respect to the number of samples by varying M from 3 to 20, while keeping the sample size same ($N = 10$). For brevity, we



(a) (b)

Figure 1. Effect of number of samples (M) on SIBSAA's performance in the case of (a) static strategy with $\alpha = 0.8$, and, (b) dynamic strategy $\triangle_\alpha = 0.05$

illustrate only the results for dynamic strategy with $\triangle_\alpha = 0.05$ and static strategy with $\alpha = 0.8$. Remainder of the settings are taken as before.Figure 1 shows that increasing the number of samples improves quality of the solution in both static and dynamic strategies. Further, the SIBSAA algorithm is able to converge to the optimal solution even with small number of samples, e.g., $M = 5$. While the CPU time increases with increasing number of samples, this increase is linear in M and the CPU time performances of both strategies are similar.



(a) (b)

Figure 2. Effect of the swarm learning parameter update (β) on SIBSAA's performance in the case of (a) static strategy with $\alpha = 0.8$, and, (b) dynamic strategy $\triangle_\alpha = 0.05$

Lastly, we tested the sensitivity of the SIBSAA with respect to the swarm learning update (β) using both the static and dynamic strategies of the swarm learning bias (Figure 2). In the static strategy, increasing the swarm learning rate results in faster convergence but to an inferior solution. In comparison, with dynamic strategy using $\triangle_\alpha = 0.05$, the solution quality first improves and then declines as the CPU time decreases (Figure 2b). This indicates an

interaction between the swarm learning bias parameter and swarm learning update. While the dynamic strategy can converge to a solution faster than the static strategy, the swarm learning update and bias parameters need to be tuned jointly to ensure high solution quality. Lastly, we note that as the bias parameter ($\triangle_\alpha$) and the swarm learning update (β) increase, the convergence rate increases. However, at the same time, the solution quality might degrade as the best incumbent solution might not have improved sufficiently. Hence, a good tuning strategy is to use high (low) β with low (high) $\triangle_\alpha$ so in order to achieve fairly speedy convergence to a good quality solution.

4.3. ***Computational Performance of SIBSAA***

In this section, we compare the performances of the SAA and the proposed SIBSAA. In all experiments, we use the same settings of β= 2 and static strategy for learning bias with α= 0.8. In Figure 3, we present the CPU time and solution quality performance of the SAA for $N = \{10, 25, 50\}$ sample sizes and compare with that of the proposed SIBSAA with $N = 10$ in solving CRFLP with failure probabilities q=$\{$0.2,0.4,0.6,0.8$\}$. We use $M = 5$ samples in both methods. The results indicate that the sample size effect on the SAA's solution quality is varied. For instance, whereas the solution quality is non-monotone with $q = 0.2$, it is monotone decreasing (increasing) with $q = 0.4$ and $q = 0.6$ ($q = 0.8$). In none of the failure probability cases, however, the solution quality performance of SAA has converged to that of SIBSAA's. Further, in all cases, the CPU time of the SAA is growing exponentially. For instance, with $q = 0.4$ and $q = 0.6$, the CPU time of SAA has grown about 14 and 2 times that of the SIBSAA's while a significant portion of the solution gap still remaining.

We further analyzed the performance of the proposed SIBSAA with respect to that of the exact method and the SAA method with different sample sizes (N) and number of samples (M). Tables 2 and 3 illustrate these benchmark results for q=$\{$0.1,0.3,0.5,0.7$\}$. The third column, F*, indicates the solution converged by each method, e.g., facilities opened. For instance, with $q = 0.3$, the SAA's solution is to open facilities F*=$\{$2,4,5,8$\}$ whereas the SIBSAA and exact solution open facilities F*=$\{$1,2,10$\}$. Fourth column present the objective function value (Obj.) for the SAA, SIBSAA and exact method, e.g., v^{SAA}, v^{SIBSAA} andv^*. Fifth column, $\overline{v}^M$, is the average of the sample optimal objective functions for the SAA method. The sixth and seventh columns display two optimality gap measures. The first gap (GAP_1) is only applicable for the SAA since it is based on the assumption that $\overline{v}^M$ is a statistical lower bound on the v^*. It is defined as,

$$\text{GAP}_1 = \frac{v^{SAA} - \overline{v}^M}{\overline{v}^M} \times 100\%.$$

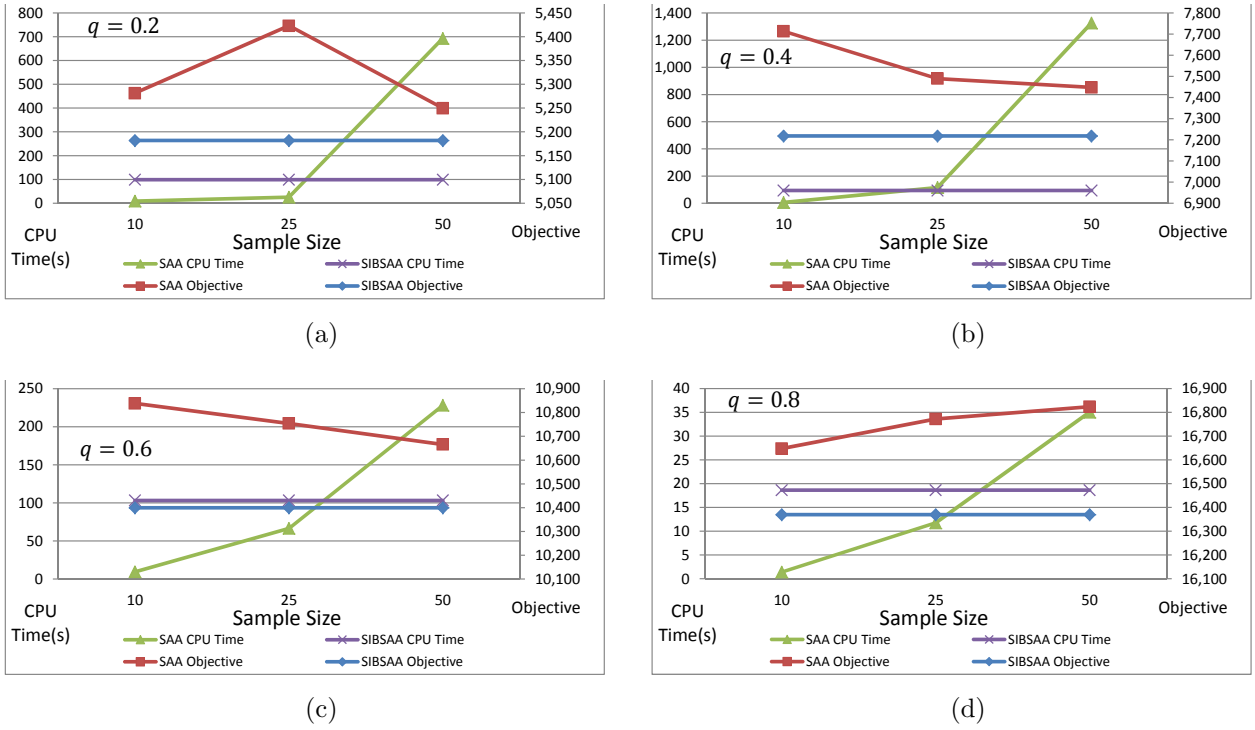


Figure 3. Effect of sample size on the solution quality and CPU time performance of SAA in comparison with SIBSAA for CRFLP with facility failure probabilities (a) q=0.2, (b) q=0.4, (c)q=0.6, and (d) q=0.8.

The second gap (GAP_2) is applicable to both the SAA and the SIBSAA, and uses the optimal solution value v^*. It is defined as,

$$\text{GAP}_2 = \begin{cases} \frac{v^{SAA} - v^*}{v^*} \times 100\% & \text{for SAA,} \\ \frac{v^{SIBSAA} - v^*}{v^*} \times 100\% & \text{for SIBSAA.} \end{cases}$$

Tables 2 and 3 show that, as the sample size increases, the SAA's objective function is not always monotonously decreasing while the CPU time is increasing exponentially. These observations are in accordance with those in Figure 3. Note that the GAP_1 is monotonously decreasing since the effect of sampling variance on v^{SAA} is compensated by $\overline{v}^M$. The negative GAP_1 values are due to the fact that $\overline{v}^M$ is a "statistical" lower bound. In Table 2, with

$q = 0.1$, we observe that SAA method finds the optimal solution in all M and N cases except $(M, N) = (5, 10)$ whereas the SIBSAA's solution is always optimal. With $q = 0.3$, the SIBSAA converged to the optimum solution in 125 seconds and the SAA's average GAP_2 and CPU time are 2.6% and $2,062$ seconds.

		q=0.1						q=0.3					
Method	**M-N**	**F***	**Obj.**	$\overline{v}^M$	**CPU(s)**	**GAP_1(%)**	**GAP_2(%)**	**F***	**Obj.**	$\overline{v}^M$	**CPU(s)**	**GAP_1(%)**	**GAP_2(%)**
SAA	5-10	1,2,4	4,401	4,194	4	4.9	1.8	2,4,5,8	6,576	5,695	12	15.5	7.0
	5-25	1,2,10	4,322	4,251	31	1.7	0.0	1,2,4,7	6,321	5,771	87	9.5	2.8
	5-50	1,2,10	4,322	4,345	181	-0.5	0.0	1,2,7,10	6,249	5,866	736	6.5	1.7
	10-10	1,2,10	4,322	4,122	11	4.9	0.0	2,4,5,10	6,521	5,624	23	16.0	6.1
	10-25	1,2,10	4,322	4,263	63	1.4	0.0	1,2,4,8	6,219	5,844	167	6.4	1.2
	10-50	1,2,10	4,322	4,336	340	-0.3	0.0	1,2,4,10	6,164	5,845	1,866	5.5	0.3
	15-10	1,2,10	4,322	4,129	18	4.7	0.0	2,4,5,10	6,521	5,628	27	15.9	6.1
	15-25	1,2,10	4,322	4,257	95	1.5	0.0	1,2,4,8	6,219	5,848	249	6.4	1.2
	15-50	1,2,10	4,322	4,319	487	0.1	0.0	1,2,4,10	6,164	5,740	2,949	7.4	0.3
	20-10	1,2,10	4,322	4,105	23	5.3	0.0	1,2,4,7	6,321	5,678	34	11.3	2.8
	20-25	1,2,10	4,322	4,216	134	2.5	0.0	1,2,4,8	6,219	5,845	309	6.4	1.2
	20-50	1,2,10	4,322	4,297	628	0.6	0.0	1,2,8,10	6,146	5,784	18,288	6.3	0.0
SIBSAA	5-10	1,2,10	4,322	-	48	-	0.0	1,2,8,10	6,146	-	125	-	0.0
Exact	-	1,2,10	4,322		659	-	-	1,2,8,10	6,146	-	>21,600	-	-

Table 2. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probabilities $q = 0.1$ and $q = 0.3$.

Table 3 results for $q = 0.5$ and $q = 0.7$ show that SAA is not able to find the optimum solution. With $q = 0.5$, the SIBSAA converged to the optimum solution in 82 seconds and the SAA's average GAP_2 and CPU time are 3.5% and 1,259 seconds. With $q = 0.7$, the SIBSAA converged to the optimum solution in 43 seconds and the SAA's average $GAP_2$and CPU time are 3.1% and 120 seconds. These results show that SIBSAA is able to find better quality solutions in much less time than SAA. Table 4 presents the results for failure probability $q = 0.9$. The SIBSAA converge to a solution with GAP_2=0.1% in 6 seconds, whereas the SAA's average $GAP_2$and CPU time are 1.2% and 7 seconds.

Figure 4 illustrates the effect of facility failure probability on the solution quality (GAP_2) and CPU time performance of SIBSAA with $N = 10$ in comparison with those of SAA with $N = 10$, 25 and 50 scenarios. The SIBSAA is always outperforming the SAA in terms of solution quality; the largest improvements are observed when the failure probability is neither too low nor too high (Figure 4a). In Figure 4b, a semi-log plot, the CPU time performance of SIBSAA is similar to that of SAA with $N = 25$ and significantly better than $N = 50$. in comparison with SAA, the SIBSAA improves the solution quality with

		q=0.5						q=0.7					
Method	**M-N**	**F***	**Obj.**	$\overline{v}^M$	**CPU(s)**	**GAP$_1$(%)**	**GAP$_2$(%)**	**F***	**Obj.**	$\overline{v}^M$	**CPU(s)**	**GAP$_1$(%)**	**GAP$_2$(%)**
SAA	5-10	1,2,3,4,5,7	9,206	7,710	4	19.4	6.6	1,2,4,7,10	13,307	9,594	3	38.7	3.1
	5-25	2,4,7,8,10	9,003	7,953	81	13.2	4.3	2,4,5,8,10	13,564	11,775	21	15.2	5.1
	5-50	1,2,7,8,10	8,746	8,397	1,953	4.2	1.3	1,2,4,6,7,10	13,282	12,461	114	6.6	2.9
	10-10	1,2,3,4,5,7	9,206	7,181	9	28.2	6.6	1,2,4,7,10	13,307	10,283	5	29.4	3.1
	10-25	1,2,3,7,10	8,997	7,742	119	16.2	4.2	1,2,3,8,9,10	13,441	11,776	46	14.1	4.2
	10-50	1,2,7,8,10	8,746	8,448	3,265	3.5	1.3	1,2,4,57,8,10	13,169	12,326	218	6.8	2.1
	15-10	2,5,7,8,10	9,100	7,322	13	24.3	5.4	1,2,4,7,10	13,307	10,349	8	28.6	3.1
	15-25	1,2,4,5,8	8,861	7,970	196	11.2	2.6	1,2,4,5,6,8	13,415	11,655	74	15.1	4.0
	15-50	1,2,7,8,10	8,746	8,312	4,163	5.2	1.3	1,2,3,7,8,10	13,141	12,116	365	8.5	1.8
	20-10	1,2,4,8,9	9,088	7,327	17	24.0	5.3	1,2,4,7,10	13,307	10,163	10	30.9	3.1
	20-25	1,2,4,7,8	8,811	7,875	256	11.9	2.1	1,2,4,7,10	13,307	11,670	97	14.0	3.1
	20-50	1,2,4,7,10	8,710	8,323	5,036	4.6	0.9	1,2,3,4,5,8,10	13,020	12,057	482	8.0	0.9
SIBSAA	5-10	1,2,4,8,10	8,634	-	82	-	0.0	1,2,4,7,8,10	12,903	-	43	-	0.0
Exact	-	1,2,4,8,10	8,634	-	>21,600	-	-	1,2,4,7,8,10	12,903	-	>21,600	-	-

Table 3. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probabilities $q = 0.5$ and $q = 0.7$.

		q=0.9					
Method	**M-N**	**F***	**Obj.**	$\overline{v}^M$	**CPU(s)**	**GAP$_1$(%)**	**GAP$_2$(%)**
SAA	5-10	2,3,8,10	20,891	17,012	0	22.8	2.2
	5-25	2,3,4,8	20,921	18,821	1	11.2	2.4
	5-50	2,4	20,589	19,506	2	5.6	0.7
	5-100	2,4,7	20,741	19,985	5	3.8	1.5
	10-10	2,3,8,10	20,891	17,201	1	21.5	2.2
	10-25	2,4,	20,589	19,294	2	6.7	0.7
	10-50	2,8	20,586	19,771	4	4.1	0.7
	10-100	2,8	20,586	20,132	10	2.3	0.7
	15-10	1,4,8	20,864	17,431	1	19.7	2.1
	15-25	2,4	20,589	18,999	3	8.4	0.7
	15-50	2,8	20,586	19,763	9	4.2	0.7
	15-100	2,10	20,561	20,065	18	2.5	0.6
	20-10	2,3,4	20,837	17,422	2	19.6	1.9
	20-25	2,8	20,586	18,924	4	8.8	0.7
	20-50	1,2,4,8	20,567	19,634	12	4.8	0.6
	20-100	1,2,8	20,482	20,025	34	2.3	0.2
SIBSAA	5-10	1,2,10	20,450	-	6	-	0.1
Exact	-	1,2	20,439	-	239	-	-

Table 4. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probability $q = 0.9$.

similar computational effort. Moreover, the computational effort necessary to attain the same solution performance is much less with SIBSAA than SAA.

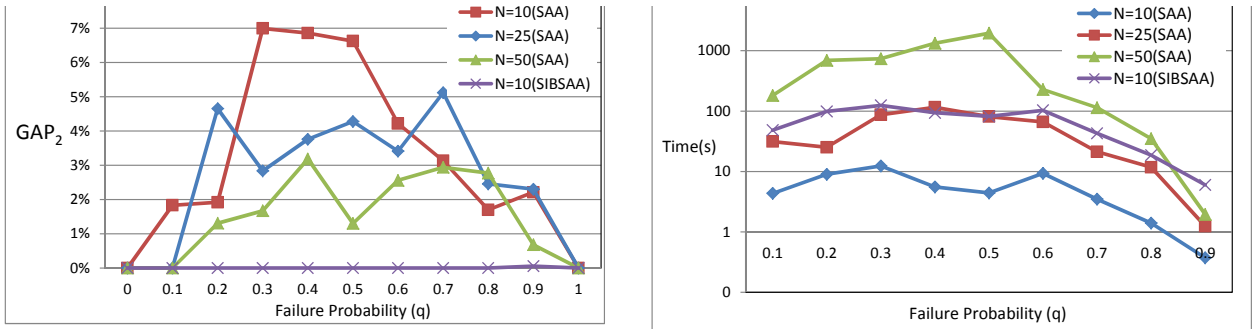


Figure 4. Solution quality (a) and CPU time (b) comparison of SIBSAA and SAA with different sample sizes (M) and failure probabilities (q).

5. Conclusion

We developed a hybrid method, SIBSAA, which integrates the swarm intelligence concept of the PSO within the SAA methodology to efficiently solve the capacitated reliable facility location problems. This integration considers each sample solution as a particle in the swarm and employs the SAA algorithm iteratively. In each iteration, the social learning is injected into the solution process by introducing penalty terms in the objective that guides the solution of each sample to the swarm's balanced solution. The two key parameters of SIBSAA are swarm's learning bias and swarm's learning parameter. The learning bias parameter adjusts the importance given to swarm's best found solution and to the most recent average sample solution in calculating the swarm's balanced solution. The learning parameter modulates the rate at which the sample solutions converge to the swarm's balanced solution. Given that the swarm's best found solution improves over time, we propose two strategies for the bias parameter: static versus dynamic strategy.

We first conducted experiments for sensitivity analysis of the algorithm with respect to the parameters as well as number of samples. The results show that the SIBSAA's solution quality performance is relatively insensitive to the choice of strategy for the swarm learning bias parameter, i.e., both the static and dynamic strategies are able to converge to the optimum solution. However, the dynamic strategy is slightly more computationally efficient than the static strategy. Further, increasing the number of samples improves the solution quality at a cost of linearly increasing computational effort. Lastly the SIBSAA is able to converge to the optimal solution even with small number of samples. In addition to the sensitivity experiments, we compared the performance of SIBSAA with SAA's. These results show that the SIBSAA is able to improve the solution quality noticeably with reasonable

computational effort compared to SAA. Further, increasing SAA's sample size to match the solution quality performance of SIBSAA requires significant computational effort which is not affordable for many practical instances of CRFLP.

There are three possible avenues of future research on SIBSAA. First opportunity is to investigate the integration of alternative swarm intelligence mechanisms in order to improve the convergence rate and solution quality. Another extension is to investigate the conditions under which the dynamic strategy for swarm's learning bias is more advantageous over the static strategy. Lastly, the adaptation of the ideas of SIBSAA to multi-stage stochastic programs, e.g., reliable facility location over multiple periods, is another fruitful future research avenue.

REFERENCES:

Ahmed, S., Shapiro, A., 2002. The sample average approximation method for stochastic programs with integer recourse. Optimization Online, http://www.optimization-online.org/.

Arostegui, Jr. M.- A., Kadipasaoglu, S.-N., Khumawala, B.-M., 2006. An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of Production Economics, 103 (2), 742–754.

Brennan, P., 2011. Lessons learned from the Japan earthquake. Disaster Recovery Journal, 24(3), http:// www.drj.com/2011-articles/summer-2011-volume-24-issue-3/lessons-learned-from-the-japan-earthquake.html. Last accessed on Sunday, February 05, 2012.

Chouinard, M., D'Amours, S., Aït-Kadi, D., 2008. A stochastic programming approach for designing supply loops. International Journal of Production Economics. 113, 657–677.

Gade, D. Capacitated facilities location problems with unreliable facilities. Master's Thesis, University of Arkansas, 2007.

Higle, J.-L., Sen, S., 1991. Stochastic decomposition: An algorithm for two-stage linear programs with re-course. Mathematics of Operations Research, 16, 650- 669.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. The Proceedings of the International Conference on Systems, Manand Cybernetics, 5, IEEE Press, 4104–4108.

Kennedy, K., Eberhart, R., 1995. Particle swarm optimization. In The Proceedings of the 1995 IEEE International Conference on Neural Network, 1942–1948.

Kleywegt, A.-J., Shapiro, A., Homem-De-Mello, T., 2001. The sample average approximation method for stochastic discrete optimization. SIAM Journal of Optimization, 12, 479–502.

Kratica, J., Tosic, D., Filipovic, V., Ljubic, I., 2001. Solving the simple plant location problems by genetic algorithm. RAIRO Operations Research, 35, 127–142.

Linderoth, J., Shapiro, A., Wright, S., 2006. The empirical behavior of sampling methods for stochastic programming. Annals of Operations Research, 142(1), 215–241.

Masihtehrani, B., 2011. Stochastic analysis of disruption in supply chain networks. PhD Dissertation, Pennsylvania State University, USA.

Peng, P., Snyder, L.W., Lim, A., Liu, Z., 2011, Reliable logistics networks design with facility disruptions, Transportation Research Part B, 45, 1190–1211

Rockafeller, R.-T., Wets, R.-J.-B., 1991. Scenarios and policy aggregation in optimization under uncertainty. Mathematics and Operations Research, 16 (1), 119-147.

Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A., 2005. A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research, 167, 96–115.

Schütz, P., Tomasgard, A., Ahmed, S., 2009. Supply chain design under uncertainty using sample average approximation and dual decomposition. European Journal of Operational Research 199 (2), 409–419.

Shapiro, A., Homem-de-Mello, T., 1998. A simulation-based approach to two-stage stochastic programming with recourse. Mathematical Programming, 81, 301–325.

Shen, Z.-J.-M., Zhan, R.-L., Zhang, J., 2011. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. Informs Journal on Computing, 23(3), 470-482.

Shi, Y., Eberhart, R.-C., 1998. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation, 69–73.

Snyder, L.-V., Daskin, M.-S., 2005. Reliability models for facility location: The expected failure cost case. Transportation Science, 39, 400-416.

Snyder, L.-V., Scaparra, M.-P., Daskin, M.-S., Church, R.-L., 2006. Planning for disruptions in supply chain networks. Forthcoming in Tutorials in Operations Research, Informs, Baltimore, MD, USA.

Snyder L.-V., Ülker, N., 2005. A model for locating capacitated, unreliable facilities. IERC Conference, Atlanta, GA, USA.

The Guardian, 2011. Toyota profit slides on Japan earthquake disruption. http://www.guardian.co.uk/business/2011/may/11/toyota-profit-hit-by-japan-earthquake. Last accessed on Sunday, February 05, 2012.

Van Slyke, R., Wets, R.-J.-B., 1969. L-shaped linear programs with applications to optimal control and stochastic linear programs. SIAM Journal on Applied Mathematics, 17, 638-663.

Verweij, B., Ahmed, S., Kleywegt, A., Nemhauser, G., Shapiro, A., 2002. The sample average approximation method applied to stochastic routing problems: A computational study. Technical report, Georgia Institute of Technology, Atlanta, GA, USA.

Wang, K.-J., Wang, S.-M., Chen, J.-C., 2008. A resource portfolio planning model using sampling-based stochastic programming and genetic algorithm. European Journal of Operational Research, 184, 327–340.

Zhan, R.-L., 2007. Models and Algorithms for Reliable Facility Location Problems and System Reliability Optimization, PhD Dissertation, University of Florida.

C. Results Dissemination

1. Conference Activity

Conference Presentations:

- Aydin, N, Murat, A., "A Hybrid Approach Based on Sample Average Approximation Algorithm and Progressive Hedging Algorithm for solving Stochastic Programs," INFORMS Annual Meeting, Charlotte, November 13-17, 2011.
- Azadian, F., Murat, A. and R.B. Chinnam, "Air Cargo Pickup and Delivery Problem with Alternative Access Airports," INFORMS Annual Meeting, Charlotte, November 13-17, 2011.
- Azadian, F., Murat, A. and R.B. Chinnam, "Air Cargo Pickup and Delivery Problem with Alternative Access Airports," INFORMS Annual Meeting, Charlotte, November 13-17, 2011.
- Guner, A., Chinnam, R.B., Murat, A., "Dynamic Routing in Stochastic Time Dependent Networks with Arc Interactions," Complex Adaptive Systems Conference, Chicago, Illinois, October 31-November 2, 2011.
- Guner, A., Murat, A., Chinnam, R.B., "Dynamic Routing in Stochastic Time-Dependent Networks for Milk-Run Tours with Time Windows under ITS," 4th METRANS National Urban Freight Conference, Long Beach, CA, October 12-14, 2011.
- Azadian, F., Murat, A., Chinnam, R.B., "Improving Air-cargo Transportation through Alternative Access Airport Routing," 4th METRANS National Urban Freight Conference, Long Beach, CA, October 12-14, 2011.
- Azadian, F., Murat, A., Chinnam, R.B., "Novel Framework to Exploit Freight Forwarders' Opportunities in Air-Road Multimodal Transportation Under Alternative Access Airport Policy," Complex Adaptive Systems Conference, Chicago, Illinois, October 31-November 2, 2011.
- Azadian, F., Murat, A. and R.B. Chinnam, "Dynamic freight routing on air-road intermodal network using real-time congestion information," IFORS Annual Meeting, Melbourne, Australia, July 10-15, 2011.
- Guner, Murat, A. and R. B. Chinnam, "Dynamic routing in stochastic time-dependent networks for milk-run tours with time windows," EURO Conference, Lisbon, Portugal, July 11-14, 2010.
- Guner, A., Murat, A. and R. B. Chinnam, "Dynamic Routing in Stochastic Time-Dependent Networks for Milk-Run Tours with Time Windows," ITS Michigan Annual Meeting, Dearborn, MI, May 19- 20, 2010.
- Guner, A., Chinnam, R.B., and Murat, A., "Dynamic Vehicle Routing Under Congestion Using Real-time ITS Information," ALIO-INFORMS Joint International Meeting, Buenos Aires, Argentina, June 6-9, 2010.

Conferences Planning to Attend:

- Azadian, F., Murat, A., and Chinnam, R.B., "Single Truck Pickup and Delivery Problem with Time and Destination Dependent Cost," INFORMS Annual Meeting , Phoenix, Arizona, (October 14-17, 2012).
- Aydin, N. and Murat, A., "Effect of Flexibility Decisions in Coping with Facility Disruptions in Supply Chain Networks," INFORMS Annual Meeting , Phoenix, Arizona, (October 14-17, 2012).
- Aydin, N. and Murat, A., "Hybrid Heuristic Method for Solving Multi-stage Stochastic Programs," INFORMS Annual Meeting , Phoenix, Arizona, (October 14-17, 2012).

2. Journal Publications

- Azadian*, F., Murat†, A., and Chinnam, R. B., "Dynamic Routing of Time-Sensitive Air Cargo Using Real-Time Information", Transportation Research Part E: Logistics and Transportation Review, Vol. 48, Issue 1, 2012, pp. 355-372. (Impact factor ISI-5 Year: 2.516).
- Guner*, A., Murat†, A., and Chinnam, R.B., "Dynamic routing in stochastic time-dependent networks for milk-run tours with time windows under ITS," Transportation Research-Part C: New Technologies, (Submitted July 2010, Under Revision). (Impact factor ISI-5:2.516).
- Azadian*, F., Murat†, A., and R. B. Chinnam, "A Pickup and Delivery Problem for Air-cargo Shipment with Alternative Access Airport", Transportation Science, (Submitted March 2012). (Impact factor ISI-5:2.76).
- Aydin*, N, Murat†, A., "A Swarm Intelligence Based Sample Average Approximation Algorithm for Capacitated Reliable Facility Location Problem", (Submitted February 2012), Int. Journal of Production Economics. (Impact factor ISI-5: 2.326).

3. Honors and Awards

1. Best Paper – Bronze Award, ITS Michigan Annual Meeting, Intelligent Transportation Society, 2011.

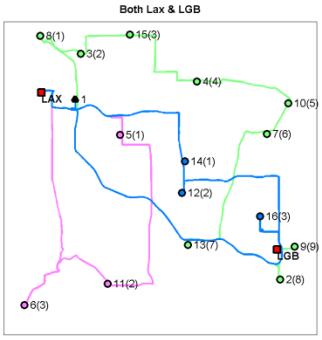
   Guner, A., Chinnam, R.B., and Murat, A., " Dynamic Routing Policies in Stochastic Time-Dependent Networks Under ITS," ITS Michigan Annual Meeting, Dearborn, MI, June 1-2, 2011.

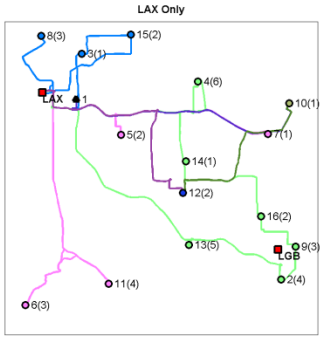
2. Best Paper – Bronze Award, ITS Michigan Annual Meeting, Intelligent Transportation Society, 2010.

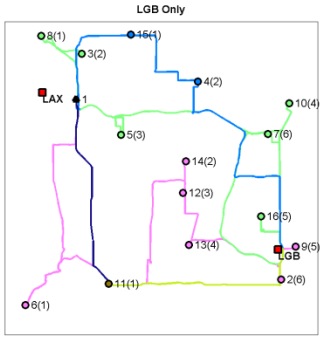
   Guner, A., Murat, A., and Chinnam, R.B., "Dynamic Routing in Stochastic Time-Dependent Networks for Milk-Run Tours with Time Windows," ITS Michigan Annual Meeting, Dearborn, MI, May 19- 20, 2010.

APPENDIX:

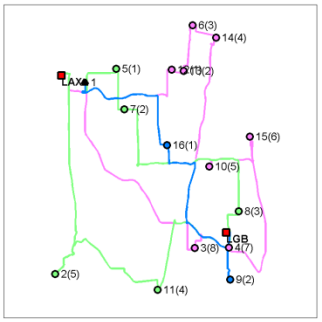
Figure A.1. Routes for ten problem instances with depot located at LAX (single airport and alternative access airport policy)

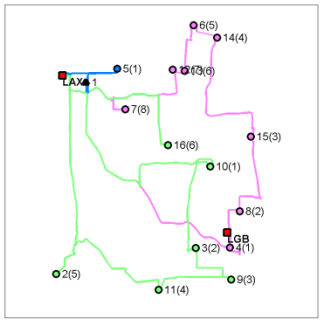


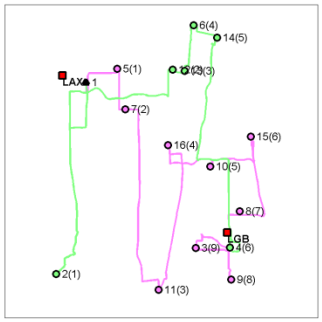




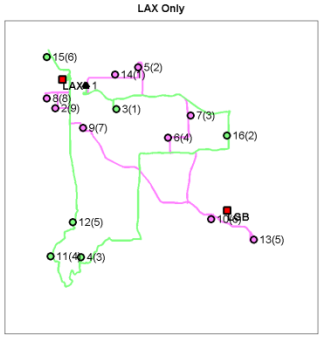
Problem 5

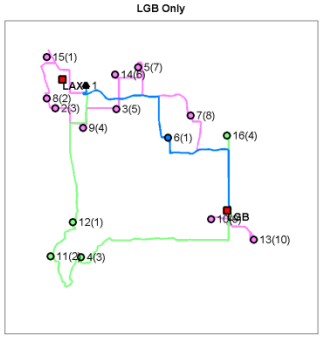




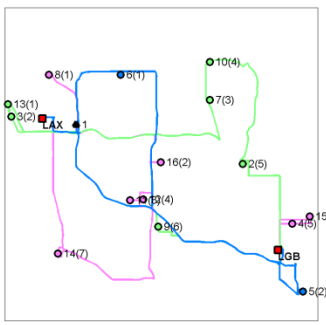


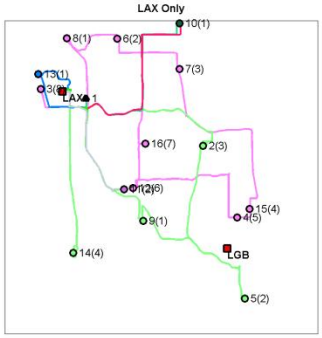
Problem 6

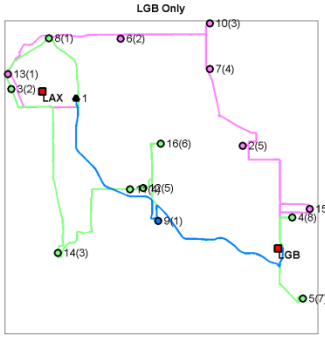




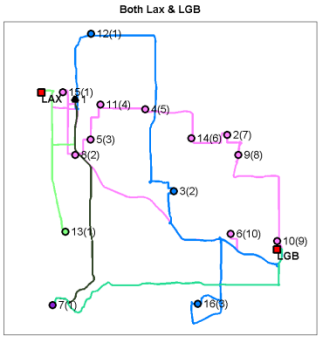
Problem 7

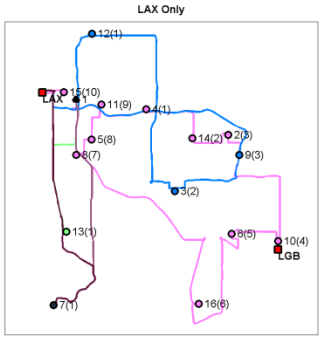


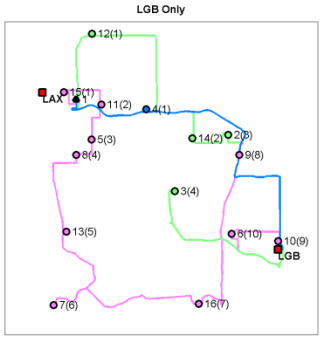




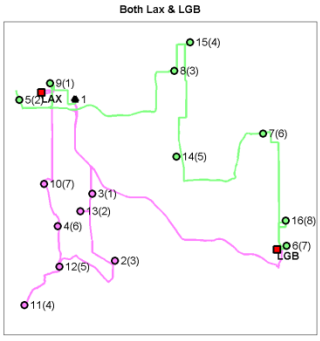
Problem 8

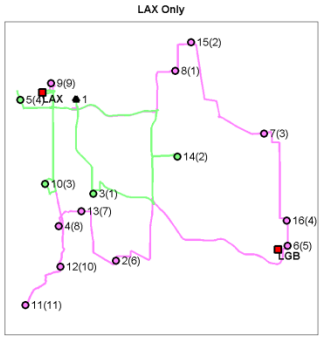


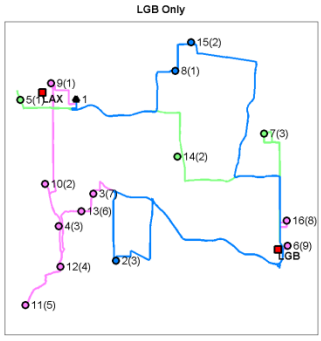




Problem 9

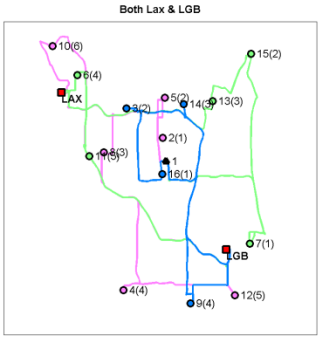


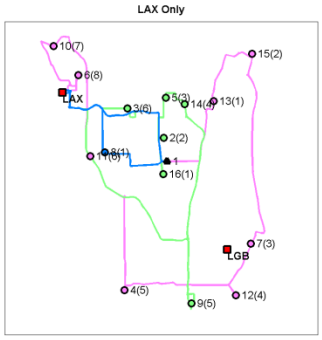


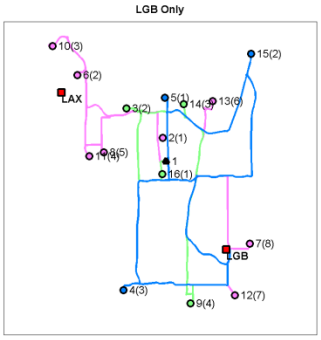


Problem 10

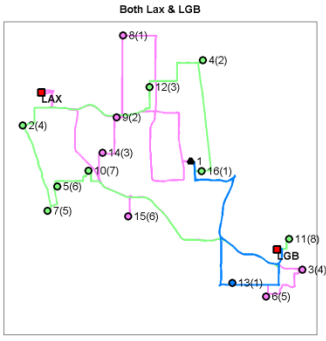
Figure A.2. Routes for ten problem instances with depot located at Compton (single airport and alternative access airport policy)

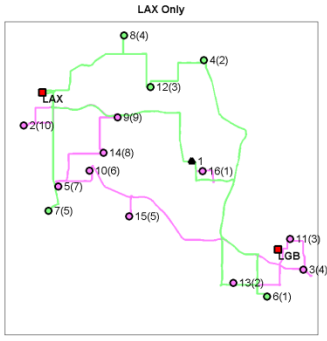


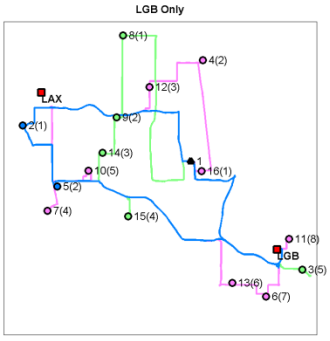




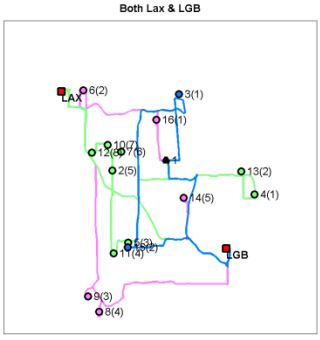
Problem 5

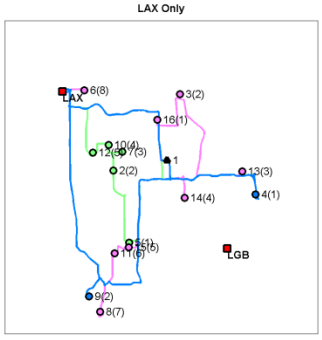


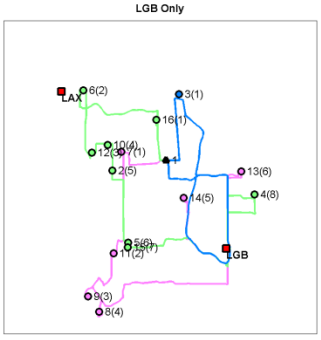




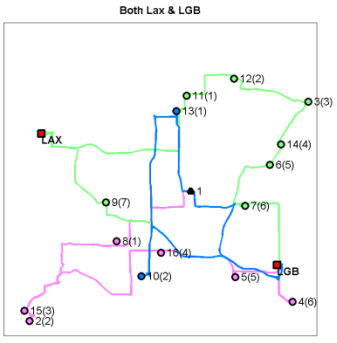
Problem 6

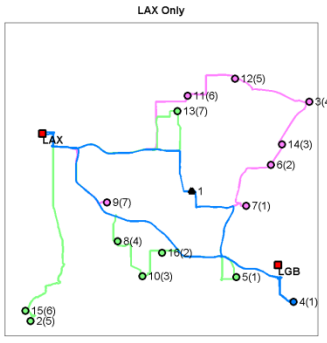


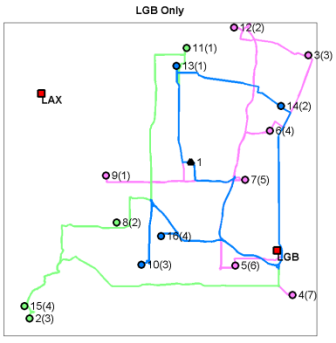




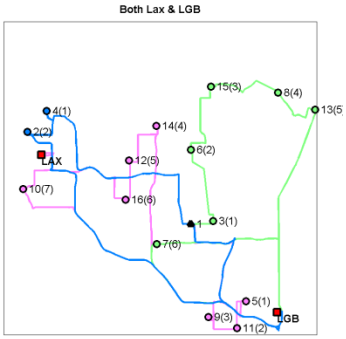
Problem 7

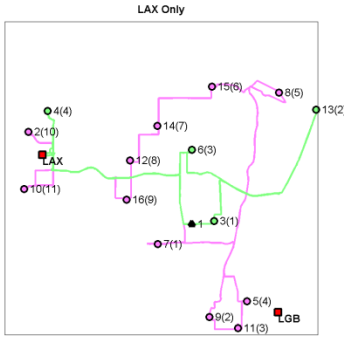


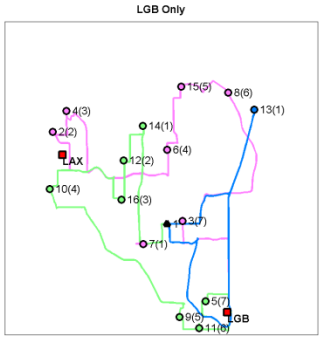




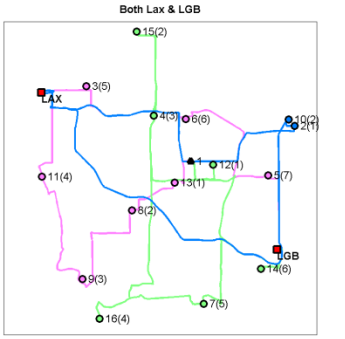
Problem 8

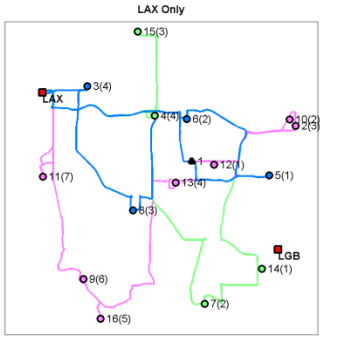


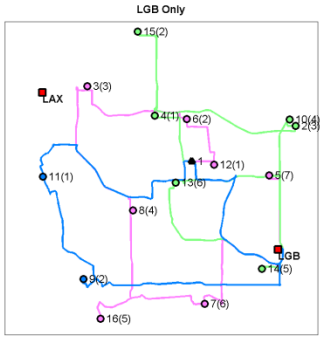




Problem 9

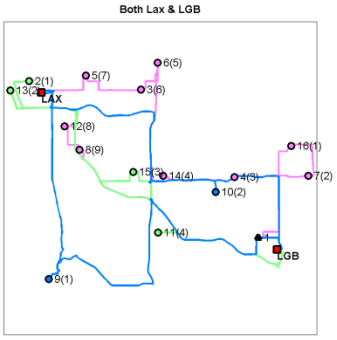


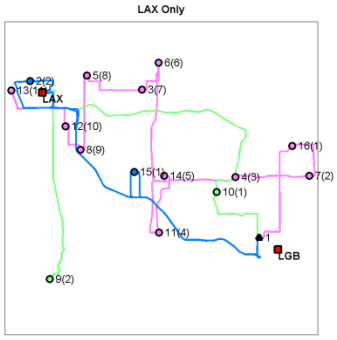


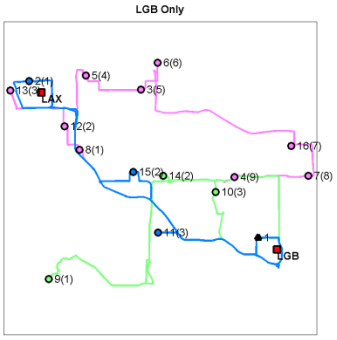


Problem 10

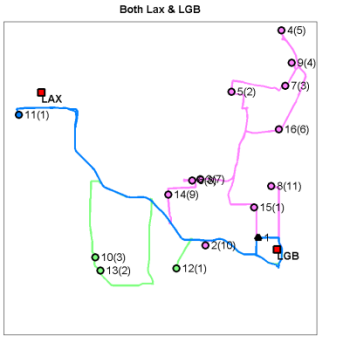
Figure A.3. Routes for ten problem instances with depot located at LGB (single airport and alternative access airport policy)

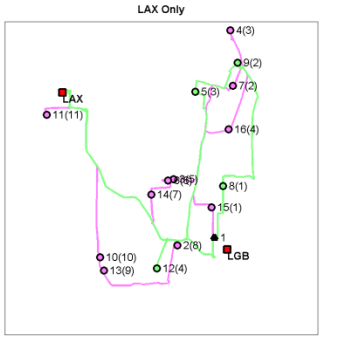


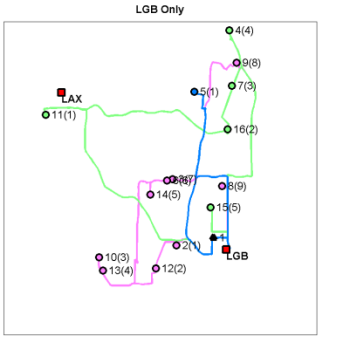




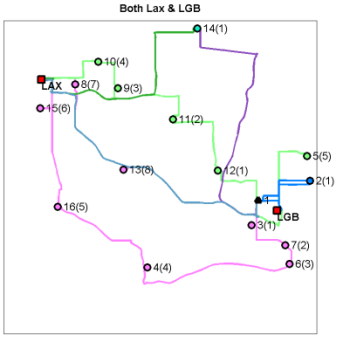
Problem 5

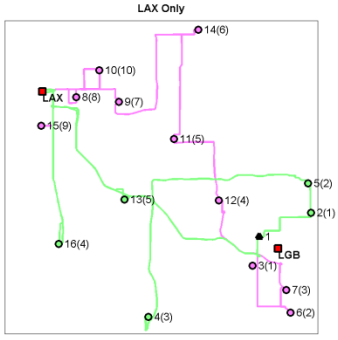


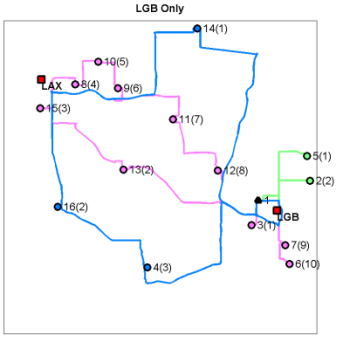




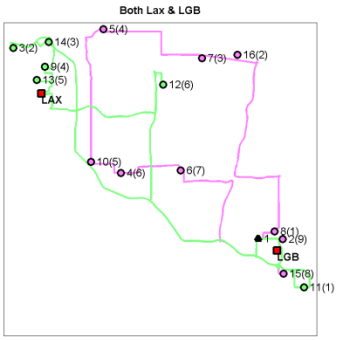
Problem 6

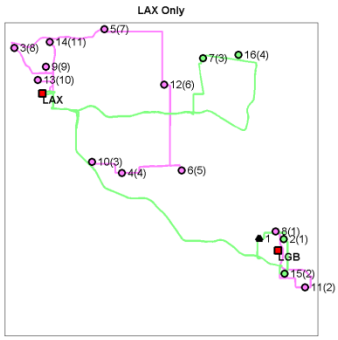


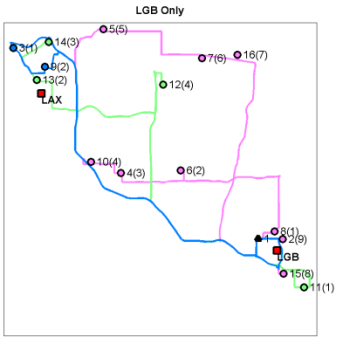




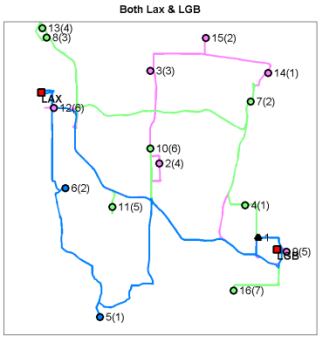
Problem 7

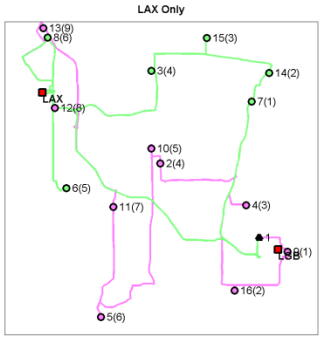


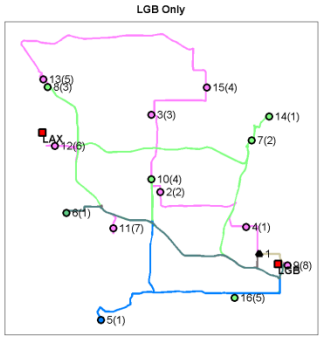




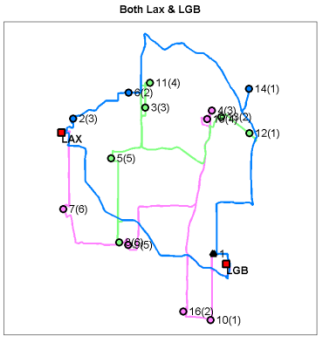
Problem 8

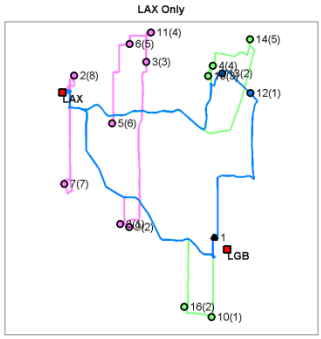


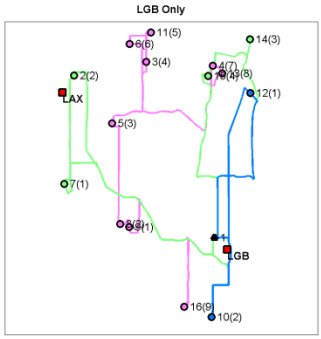




Problem 9







Problem 10