UTCM

Improving the Quality of Life by Enhancing Mobility

University Transportation Center for Mobility™

DOT Grant No. DTRT06-G-0044

Algorithms for Routing Vehicles and Their Application to the Paratransit Vehicle Scheduling Problem

Final Report

Myoungkuk Park, Paul Oberlin, Sivakumar Rathinam, Luca Quadrifoglio and Swaroop Darbha

Performing Organization
University Transportation Center for Mobility™
Texas Transportation Institute
The Texas A&M University System
College Station, TX

Sponsoring Agency
Department of Transportation
Research and Innovative Technology Administration
Washington, DC

Texas Transportation Institute

UTCM Project #09-15-13
March 2012

Technical Report Documentation Page

1. Report No. UTCM 09-15-13	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Algorithms for Routing Vehicles and Their Application to the Paratransit Vehicle Scheduling Problem		5. Report Date March 2012 6. Performing Organization Code Texas Transportation Institute
7. Author(s) Myoungkuk Park, Paul Oberlin, Sivakumar Rathinam, Luca Quadrifoglio, Swaroop Darbha		8. Performing Organization Report No. UTCM 09-15-13
9. Performing Organization Name and Address University Transportation Center for Mobility™ Texas Transportation Institute The Texas A&M University System 3135 TAMU College Station, TX 77843-3135		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTRT06-G-0044
12. Sponsoring Agency Name and Address Department of Transportation Research and Innovative Technology Administration 400 7th Street, SW Washington, DC 20590		13. Type of Report and Period Covered Final Report Sep 2009 – Sep 2011 14. Sponsoring Agency Code

15. Supplementary Notes

Supported by a grant from the US Department of Transportation, University Transportation Centers Program

16. Abstract

As the demand for paratransit services increases, there is a constant pressure to maintain the quality of service provided to the customers while minimizing the cost of operation; this is especially important as the availability of public funding for paratransit services has been on the decline. Key tasks in accomplishing this objective are efficiently allocating vehicles to service trips and adjusting the schedules of vehicles dynamically in response to calls received by the service providers from the customers on the day of the service. For many paratransit services, capacity of vehicles is not a binding constraint. This is especially so in rural applications. For this reason, we will focus on dealing with routing vehicles that are not subject to any passenger capacity constraints.

In this report, we consider two important relaxations of this problem, which may be considered as problems of independent interest and significance. The first problem deals with relaxing all the constraints associated with the order in which the vehicles must visit pickup and delivery locations of the passengers as well as the time window constraints. The second relaxation additionally imposes ordering requirements. Both problems are combinatorially hard problems, and we provide formulations and algorithms for finding sub-optimal solutions along with an estimate of their quality. In the last section of this report, we consider the time window constraints for pickup and delivery of customers and provide a heuristic to find feasible solutions. We corroborate the results numerically with small, randomly generated instances of the paratransit scheduling problem.

17. Key Word Vehicle Routing Problem, Precedence Constraints Problem, Lower Bounds	18. Distribution Statement Public distribution

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 43	22. Price n/a

Algorithms for Routing Vehicles and Their Application to the Paratransit Vehicle Scheduling Problem

By

Myoungkuk Park
Department of Mechanical Engineering
Texas A&M University

Paul Oberlin
Department of Mechanical Engineering
Texas A&M University

Sivakumar Rathinam
Department of Mechanical Engineering
Texas A&M University

Luca Quadrifoglio (co-PI)
Department of Civil Engineering
Texas A&M University

Swaroop Darbha (PI)
Department of Mechanical Engineering
Texas A&M University

Prepared for:
University Transportation Center for Mobility^{TM}
Project #09-15-13

University Transportation Center for Mobility^{TM}
Texas Transportation Institute
The Texas A&M University System
College Station, TX 77843-3123

March 2012

Disclaimer

Acknowledgement

Support for this research was provided in part by a grant from the U.S. Department of Transportation, University Transportation Centers Program to the University Transportation Center for Mobility^{TM}(DTRT06-G-0044).

Contents

List of Figures

NOTE: Color figures in this report may not be legible if printed in black and white. A color PDF copy of this report may be accessed via the UTCM website at http://utcm.tamu.edu, the Texas Transportation Institute website at http://tti.tamu.edu, or the Transportation Research Board's TRID database at http://trid.trb.org.

List of Tables

Executive Summary

Paratransit service involves primary alternative modes of transportation that do not adhere to a fixed route or schedule. Unlike public transportation, the paratransit service can provide accommodating services to the customers, including handicap-accessible-system and door-to-door transportation. Paratransit services also fill the gap for transportation needs in rural areas, where it may be expensive to operate vehicles on a fixed schedule or route. The demand for paratransit services has been on the rise.

Paratransit services are availed by a broad section of society in rural, urban, and suburban settings. Customers such as airplane passengers who do not want to park their car in an airport lot for a long period call paratransit service providers such as door-to-door shuttle services and make a reservation for a ride in advance. In rural areas, transportation services may be limited due to low passenger demand. In such scenarios, paratransit services fill the void.

Generally, paratransit service providers use mini-buses to serve the transportation needs of the passengers. A majority of the calls to the transportation providers arrive at least one day in advance. The problem of routing faced by paratransit service providers is to schedule their vehicles to meet the passenger demand in an efficient manner while providing quality service and controlling operating costs. The quality of paratransit service is important; otherwise, passengers may opt for other modes of transportation. Usually, clients assess the quality of service by the timeliness of pickup and delivery of passengers.

As the demand for paratransit services increases, there is a constant pressure to maintain the quality of service provided to the customers while minimizing the cost of operation; this is especially important as the availability of public funding for paratransit services has been on the decline. Key tasks in accomplishing this objective are efficiently allocating vehicles to service trips and adjusting the schedules of vehicles dynamically in response to calls received by the service providers from the customers on the day of the service.

For many paratransit services, capacity of vehicles is not a binding constraint. This is especially so in rural applications. For this reason, we will focus on dealing with routing vehicles, that are not subject to any passenger capacity constraints.

In this report, we consider two important relaxations of this problem, which may be considered as problems of independent interest and significance. The first problem deals with relaxing all the constraints associated with the order in which the vehicles must visit pickup and delivery locations of the passengers as well as the time window constraints. The second relaxation additionally imposes ordering requirements. Both problems are combinatorially hard problems, and we provide formulations and algorithms for finding sub-optimal solutions along with an estimate of their quality. In the last section of this report, we consider the

time window constraints for pickup and delivery of customers and provide a heuristic to find feasible solutions. We corroborate the results numerically with small, randomly generated instances of the paratransit scheduling problem.

1 Introduction

Paratransit service involves primary alternative modes of transportation that do not adhere to a fixed route or schedule. Unlike public transportation, the paratransit service can provide accommodating services to the customers, including handicap-accessible-system and door-to-door transportation. Paratransit services also fill the gap for transportation needs in rural areas, where it may be expensive to operate vehicles on a fixed schedule or route. The demand for paratransit services has been on the rise.

Paratransit services are availed by a broad section of society in rural, urban, and suburban settings. Customers such as airplane passengers who do not want to park their car in an airport lot for a long period call paratransit service providers such as door-to-door shuttle services and make a reservation for a ride in advance. In rural areas, transportation services may be limited due to low passenger demand. In such scenarios, paratransit services fill the void.

Generally, paratransit service providers use mini-buses to serve the transportation needs of the passengers. A majority of the calls to the transportation providers arrive at least one day in advance. The problem of routing faced by paratransit service providers is to schedule their vehicles to meet the passenger demand in an efficient manner while providing quality service and controlling operating costs. The quality of paratransit service is important; otherwise, passengers may opt for other modes of transportation. Usually, clients access the quality of service by the timeliness of pickup and delivery of passengers.

As the demand for paratransit services increases, there is a constant pressure to maintain the quality of service provided to the customers while minimizing the cost of operation; this is especially important as the availability of public funding for paratransit services has been on the decline. Key tasks in accomplishing this objective are efficiently allocating vehicles to service trips and adjusting the schedules of vehicles dynamically in response to calls received by the service providers from the customers on the day of the service.

For many paratransit services, capacity of vehicles is not a binding constraint. This is especially so in rural applications. For this reason, we will focus on dealing with routing vehicles, that are not subject to any passenger capacity constraints.

2 Literature Review

There are several well-organized survey papers, such as [1], [2], [3], [4], and [5], that deal with different aspects of the routing problems that are encountered in the design of algorithms for the vehicle routing problem for paratransit applications. The paratransit scheduling problem falls in the category of the general pickup and delivery problem (GPDP). In [3] and [4], Parragh et al. surveyed many papers related to the GPDP and classified those into several subclasses. The GPDPs are divided according to various classification criteria, including paired or unpaired pickup and delivery locations, transportation of goods or passengers, and so on. According to the authors' classification, the paratransit vehicle problem belongs to the dial-a-ride problem (DARP) subclass. The main difference between PDP and DARP is that since the DARP considers transportation of passengers, it includes additional constraints for customer inconvenience into the problem. The DARP received more attention of researchers after Psaraftis formulated and solved the single vehicle DARP with time windows as a dynamic program in [6]. Because of the curse of dimensionality of the dynamic program approach, the algorithm was capable of solving only a relatively small number of request cases. Following Psaraftis, various approaches have been proposed, including heuristics (see [7],[8]) and meta-heuristic approaches(see [4]). For the exact solution, Cordeau et al. proposed a branch and cut algorithm for the static DARP in [9]; "static" means that all requests are known and fixed before solving the DARP. If the requests can be changed during the service, then the DARP is referred to as a dynamic DARP. They considered only one depot, so all vehicles depart and return to the depot. In [10], Desrosiers et al. proposed a forward dynamic programming algorithm for the static single vehicle DARP. Our aim is to formulate a static DARP with time window constraints and multiple depots.

The PDP and DARP are generalizations of the traveling salesman problem (TSP), which is known to be NP-hard. Since TSP is a non-trivial combinatorial problem, generalizations of this problem such as the ones considered in the report are also non-trivial. The practical implication of a problem being NP-hard is that it is not known whether an efficient algorithm for solving every instance of the problem exactly exists; the efficiency of the algorithm is measured in terms of the running time being bounded by a polynomial function of the size of the instance. As a consequence, one can handle such NP-hard problems in two ways: either forego running time guarantees and develop techniques to solve the problem exactly, or forego optimality but find a sub-optimal solution using an algorithm with a polynomial running time guarantee along with bounds on the deviation of the solution from the optimum. In the latter category, there are two types of algorithms: those that provide bounds on the deviation of the solution to be found by the algorithm – which are referred to as approximation algorithms, and those that provide bounds after the solution is computed (in stark contrast to the approximation algorithms where the bounds are provided even before the solution is computed).

In this report, we will focus on techniques for finding bounds for the optimal solutions and describe some heuristics to find feasible solutions. Bounds on optimal solutions help in determining bounds on the quality of a feasible solution; the tighter the bounds, the better the estimate of the quality of the solution. When solving an instance of an NP-hard problem, the computational efficiency (measured in terms of running time) of an algorithm depends on the quality of the bounds, as the bounds are used frequently to prune a set of solutions that are guaranteed not to be optimal.

One way to find lower bounds for a minimization problem is by relaxing constraints. The general pickup and delivery problem has the following constraints: a customer must be picked up at a specified location in a certain time window and dropped at the customer's intended destination within a certain specified time window. There are two ways to relax this constraint: drop this constraint altogether, or keep the constraint in a weak form,for example, if customer A needs to be picked up at location A and dropped at location B. We ignore the requirement on time windows and require that location B be visited by the same vehicle only after it has visited location A. The relaxed problems by themselves are difficult problems of independent interest and yield non-trivial lower bounds. For this problem, we will focus on these two problems in the rest of the report. The former problem is the multi-depot traveling salesmen problem (MDTSP), and the latter is the precedence constrained, asymmetric TSP (PCATSP).

Combinatorial problems such as the one considered in this paper are sensitive to the mathematical formulation of the problem. For example, the TSP admits different integer program formulations with the same optimum; however, a relaxation of the integrality requirement yields completely different values. The strength of a formulation depends on how close the optimum of the relaxed program is to the integer program. For this reason, the problem formulation is important. In the TSP, the integer programming formulations from Miller, Tucker, and Zemlin (MTZ) differ from those of Dantzig, Fulkerson, and Johnson (DFJ) in the description of the sub-tour elimination constraints. The DFJ formulation admits an exponential number of constraints and is tighter in the sense that the polytope describing the feasible solutions of the relaxed program is contained in the polytope corresponding to the relaxed program for the MTZ formulation.

The MDTSP problem may be described as follows: given a set of depots, $\mathcal{D}$, from which the vehicles start and a set of customer nodes, $\mathcal{N}$, to be visited by the collection, find a tour for each vehicle so that every customer is attended to by some vehicle in the collection and the total cost of the tours is a minimum. The cost of a tour is a proxy for the cost associated with fuel consumption and other maintenance costs. The MDTSP, being a generalization of TSP, is an NP-hard problem. However, there are transformation methods that convert MDTSP to a standard TSP and we adopt the transformation from Noon and Bean [11] in this report. By doing this transformation, one can avail of the algorithms for the asymmetric traveling salesman problem (ATSP) to address the MDTSP. With the aid of the transformation, one can use the well-known Lin-Kernighan-Helsgaun (LKH) heuristic [12], which is one of the best-known heuristics for the ATSP, to solve the transformed problem. Computational results show that solutions whose costs are within 1% of the optimum can be obtained relatively fast (40 s of computation time) for the MDTSP involving 10 heterogeneous vehicles and 160 nodes. Given the difficulty of the MDTSP and the dearth of the results for the same, these results indicate a promising approach to the problem.

The PCATSP is a generalization of the ATSP with the additional constraints that certain nodes or locations must be visited before others. Many formulations of the PCATSP are based on a formulation of the ATSP based on MTZ in [13], given as follows: let the set of nodes be N, where $|N| = n$ and the depot is defined as node number 1. Let c_{ij} be the distance (cost) to travel between node i and node j.

The binary variable x_{ij} is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if the path from node i to node j is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

The MTZ formulation also uses a decision variable, u_i, which indicates the position of node i in the tour. For example, if $u_i = 1$, it is the first node to be visited in the tour, and if $u_i = 5$, it is the fifth node to be visited in the tour. The problem is formulated as:

$$min \sum_{\substack{i,j=1 \\ i \neq j}}^{n} c_{ij} x_{ij}$$

subject to

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall i \in N \tag{1}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall j \in N \tag{2}$$

$$u_j - u_i \geq 1 - (n-1)(1 - x_{ij}) \quad i, j \in \{2, \ldots, n\},\ i \neq j \tag{3}$$

$$1 \leq u_i \leq (n-1) \quad i, j \in \{2, \ldots, n\},\ i \neq j \tag{4}$$

An interesting feature of this formulation is the sub-tour elimination constraint given by inequalities (3) and (4). These inequalities place restrictions on the variables u_i, which represent the position of node i in the tour. This variable gives the formulation of a notion of order, which other formulations lack. A precedence relation of node i must precede node j can then be expressed as $u_j \geq u_i + 1$. One way to measure the quality of a formulation is to observe the gap between the optimal solution of the integer linear program (ILP) and the optimal solution to the problem with the integrality requirement dropped. The second problem is called the linear program (LP) relaxation. If the optimal solution to the LP relaxation is close to the optimal solution of the ILP the formulation is said to be tight. In the case of the MTZ formulation, the price paid for simple constraints is that it is the weakest of all known ATSP formulations[14, 15, 16]. This is important because ILPs are currently solved using iterative algorithms that solve the LP relaxation at each iteration, constituting the majority of the computation. In general, a tight LP relaxation will decrease the number of iterations required to find the ILP optimum over a weak LP relaxation. With this in mind, some authors have tried to strengthen the formulation with the MTZ sub-tour elimination constraints.

Desrochers and Laporte strengthened the MTZ sub-tour elimination constraints in [17] by applying a lifting procedure to the sub-tour elimination constraints. The lifting procedure strengthens the original constraints by adding in other variables from the problem in such a way that the constraint remains valid for the integer program. This procedure resulted in a tighter formulation than the unmodified MTZ formulation. Next, Sherali and Driscoll applied the reformulation linearization technique (RLT) to propose a formulation based on the MTZ in [18]. The RLT linearizes a non-linear program by replacing non-linear terms with new variables and corresponding constraints that increase the size of the problem. This results in a formulation that analytically strengthens the formulation of Desrochers and Laporte in [17]; however, in general, solving the problem using this formulation takes more time.

In order to improve the convergence of iterative algorithms to solve the PCATSP, several valid inequalities, or "cuts," have been proposed. Cut generation algorithms inspect the solution of the LP relaxation for infeasibility with respect to the ILP and generate inequalities that separate this solution from the set of feasible solutions of the LP relaxation. One such cut was proposed by Ascheuer et al. in [19] that, in essence, separates LP solutions that contain a path from a successor to a predecessor. In other words, if the PCATSP has the requirement that node i must precede node j, then this cut removes solutions that contain a path from node j to node i. Another class of cuts was proposed by Balas et al. in [20] and exploit the fact that if node i must precede node j, then the path from the depot to node i should not contain node j, the path from node i to node j should not contain the depot, and the path from node j to the depot should not contain node i. Using these cuts along with ATSP cuts, Ascheuer et al. proposed a cutting plane approach in [21]. The cutting plane approach was implemented in a branch and cut algorithm to solve the PCATSP in [22].

Unsatisfied with the tightness of contemporary PCATSP formulations, Gouveia and Pires proposed a multicommodity flow formulation (MCF) for the PCATSP in [23]. This model takes the flow point of view in order to solve the ATSP, to which precedence constraints can be easily added. This formulation introduces a flow variable, F_{kij}, which represents the flow through the edge connecting node i to node j in the path from the depot to node k. This formulation was shown analytically to have a tighter LP relaxation than the MTZ-based formulations, although its large size increases solution time in general.

Recognizing the need to bridge the gap between the compact but weak MTZ formulation and the tight but cumbersome MCF formulation, Sarin, Sherali, and Bhootra proposed a new formulation (PCATSPxy formulation) in [24]. This formulation introduced a new precedence variable, y_{ij}, which represents if node i precedes node j in the tour, not necessarily immediately. Computational results showed that solution to this model with precedence constraints outperformed MTZ- and MCF- based formulations, both in number of iterations and computational time. For this reason, PCATSPxy will be used to investigate the PCATSP in this report.

In the next section, we will describe the mathematical formulations and the associated techniques used for solving the problems considered in the report.

3 Multiple Depot Routing

The MDTSP is a generalization of the multiple TSP. Multiple TSPs (MTSPs) can be classified according to the starting location of the salesmen (vehicles). If all the salesmen start at the same location (depot), then the corresponding MTSP is called a single depot MTSP; otherwise, it is referred to as a multiple depot MTSP. In Bellmore and Hong [25], Hong and Padberg [26], and Rao [27], the authors consider a single depot MTSP (SDMTSP) where each vehicle is available for service at a specific cost and the edge costs need not satisfy triangle inequality. They provide a way of transforming this SDMTSP to a standard TSP. Jonker and Volgenant [28] gave an improved transformation for a variant of the symmetric SDMTSP where each vehicle has to visit at least one target. For the single depot, heterogeneous, multiple TSP, Noon and Bean presented a transformation in [11] to a single ATSP. In [11], Noon and Bean also mentioned that their transformation can be extended to the multiple depot case. In [27], Rao gave a transformation for a two-depot TSP. For the variant of the multiple depot TSP where each vehicle need not return to its initial depot and must visit at least one target, GuoXing [29] provided a transformation to a single ATSP.

The transformation presented in section 3.2 uses the Noon-Bean transformation to convert the MDTSP to a single ATSP.

3.1 Problem Statement

Let there be n customer locations (which will be referred to as targets interchangeably) and m vehicles located at distinct depots. Let V (T) be the set of vertices that correspond to the initial locations of the vehicles (targets), with the m vertices, $\{V_1, \ldots, V_m\} = V$, representing the vehicles (i.e., the vertex V_i corresponds to the i^{th} vehicle) and $\{T_1, \ldots, T_n\} = T$ representing the targets. Let $\mathtt{V}^i = V_i \bigcup T$ be the set of all the vertices corresponding to the i^{th} vehicle. Let $E^i = \mathtt{V}^i \times \mathtt{V}^i$ denote the set of all edges (pairs of vertices) corresponding to the i^{th} vehicle, and let $C^i : E^i \rightarrow \Re_+$ denote the cost function with $C^i(a,b)$ representing the cost of traveling from vertex a to vertex b for vehicle i. We consider all the cost functions to be asymmetric, i.e., $C^i(a,b)$ may not be equal to $C^i(b,a)$ for all $a, b \in \mathtt{V}^i, i = 1, \ldots, m$. A vehicle either does not visit any target or visits a subset of targets in T. If the i^{th} vehicle does not visit any target, then its tour, $TOUR_i, = \emptyset$ and its corresponding cost, $C(TOUR_i), = 0$. If the i^{th} vehicle visits at least one target, then its tour may be represented by an ordered set, $\{V_i, T_{i_1}, \ldots, T_{i_{r_i}}, V_i\}$, where $T_{i_l},\ l = 1, \ldots,$ and r_i corresponds to r_i distinct targets being visited in that sequence by the i^{th} vehicle. There is a cost, $C(TOUR_i)$, associated with a tour for the i^{th} vehicle visiting at least one target, and it is defined as $C(TOUR_i) = C^i(V_i, T_{i_1}) + \sum_{k=1}^{r_i-1} C^i(T_{i_k}, T_{i_{k+1}}) + C^i(T_{i_{r_i}}, V_i)$. This article addresses the following heterogeneous, multiple depot, multiple vehicle routing problem or multi-depot traveling salesmen problem: find tours for the vehicles so that

- each target is visited exactly once by some vehicle, and
- the overall cost defined by $\sum_{i \in V} C(TOUR_i)$ is minimized.

3.2 Transformation Method for the MDTSP

Before presenting the details, we first present the basic ideas behind the transformation in the following discussion. MDTSP can be transformed to a single ATSP in two steps. In the first step, we transform it to an one-in-a-set ATSP, and then the resulting one-in-a-set ATSP can be converted to a single ATSP using the Noon-Bean transformation [11]. As it is vital to understand the Noon-Bean transformation, we will explain the same in the next subsection.

Transforming a one-in-a set TSP to a single ATSP

For illustrative purposes, consider a scenario where one must go to an automatic teller machine (ATM), a grocery store, a gas station, and a bookstore before getting back home (depot). Suppose there is a set of locations corresponding to an ATM, a different set of locations for a grocery store, and so on. Suppose there are a lot of one-way toll roads in the city where these are located. The problem of a one-in-a-set ATSP is that of picking the route with the shortest cost (toll) so that one can visit exactly one ATM, grocery store, gas station, and bookstore before getting home. A schematic of the problem set up with a feasible solution is shown in Figure 1.

What we seek is the following: we want to find a modified topology (costs of the roads as well as connections between ATMs, bookstores, etc. [i.e., between all the vertices]) so that one may obtain the optimal solution to the one-in-a-set ATSP from the optimal solution to the single ATSP for the modified graph.

In this pursuit, let us consider the set of ATM locations to choose from in the illustrative example. Since only one ATM needs to be visited, the optimal solution to the one-in-a-set ATSP will not change by removing the roads connected between the ATMs and replacing them with a toll-free, one-way ring road. In the one-in-a-set ATSP, one introduces a zero-cost, directed cycle among the vertices in each group, as shown in Figure 2. The introduction of the direct cycle will not alter the solution to the one-in-a-set ATSP but is important in the transformation of the one-in-a-set ATSP to the single ATSP, as it maintains connectivity between all the vertices in a group. One can easily see a tour-like solution to the single ATSP with exactly the same cost as the corresponding feasible solution to the one-in-a-set ATSP in Figure 2. To make it a tour, one can exit a group as soon as all the targets in the group are visited and move on to the vertex in the next group it would have visited, as shown in Figure 3. However, such a construction requires modification of the cost of the edges for the single ATSP to ensure that the cost of the single ATSP and the cost of the corresponding

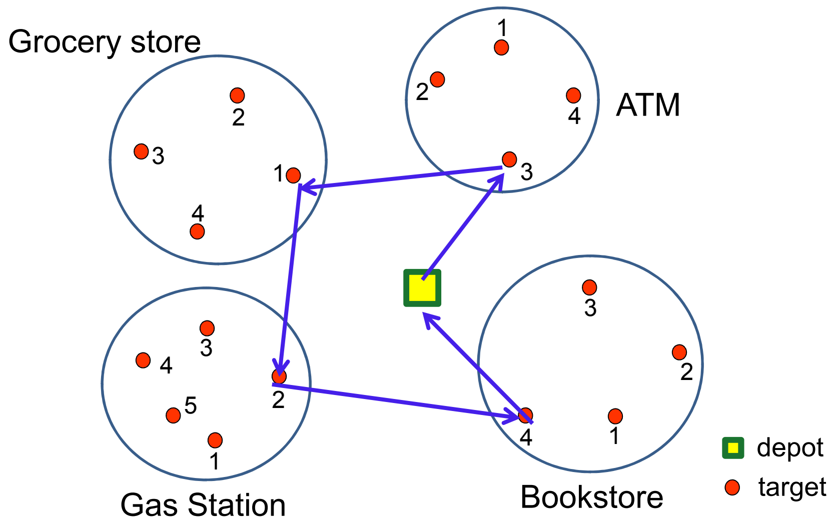


Figure 1: Illustration of a feasible solution for the one-in-a-set ATSP.

feasible solution to the one-in-a-set ATSP are the same. To see the difficulty, let us say we visited the third ATM and then went to the first grocery store. When one visits the third ATM (the ATM visited for the first time in the group of ATMs), one takes toll-free road connecting all other ATMs and exits the group after visiting the second ATM. (We note that one cannot get back to the third ATM before exiting the group, as the directed ATSP requires entering a vertex exactly once and leaving the vertex exactly once). However, the cost of the edge connecting the second ATM to the first grocery store is not the same as the cost of the edge connecting the third ATM to the first grocery store as in the feasible solution to the one-in-a-set ATSP. For this reason, the cost of the outgoing edges in the single ATSP are adjusted so that the cost of leaving from the second ATM to the first grocery store for the single ATSP is the same as the cost of traveling from the third ATM to the first grocery store in the one-in-a-set ATSP. In general, once the direction and sequence of vertices in the zero-cost cycle is fixed in each group, one can always identify a successor vertex (say v) to any given vertex (say u) on the cycle. One can set the cost of the directed edge in the transformed graph from u to a vertex w in another group to be the same as the cost of the edge from v to w in the original graph. By doing so, one can always construct a feasible solution to

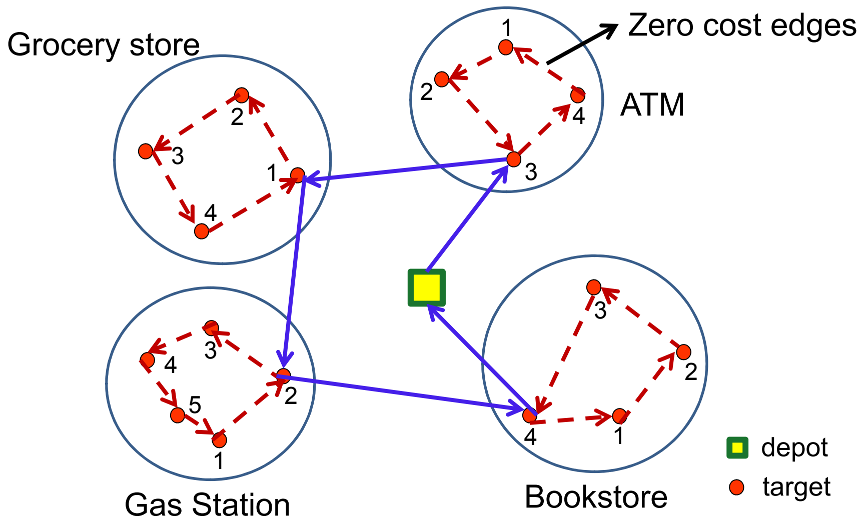


Figure 2: Construction of a transformation from one-in-a-set ATSP to a single ATSP: addition of toll-free ring roads (zero cost edges) connecting the targets within each group.

the single ATSP corresponding to a feasible solution to the one-in-a-set ATSP without any change in cost.

A question, however, remains: Is it possible to recover the solution to the one-in-a-set ATSP by solving the single ATSP? Generally, the answer is no; however, with one more modification, it is possible to recover the solution. The problem lies in the fact that it may not be cheaper sometimes to visit all the vertices in a group at once, but rather to break from the loop, visit another nearest vertex from another group, and get back to the loop at some other vertex. If there are n target groups and one depot, there will be more than $n+1$ edges connecting vertices from different groups in this case. To disallow such optimal solutions to the single ATSP and allow us to recover the solution to the one-in-a-set ATSP, we will need the following observation: Suppose the cost of the edges connecting the vertices from different groups were to be increased by the same positive constant (say M) for the single ATSP. Then, every feasible solution to the single ATSP corresponding to a feasible solution of one-in-a-set ATSP would cost exactly $(n+1)M$ more. If M were to be an upper

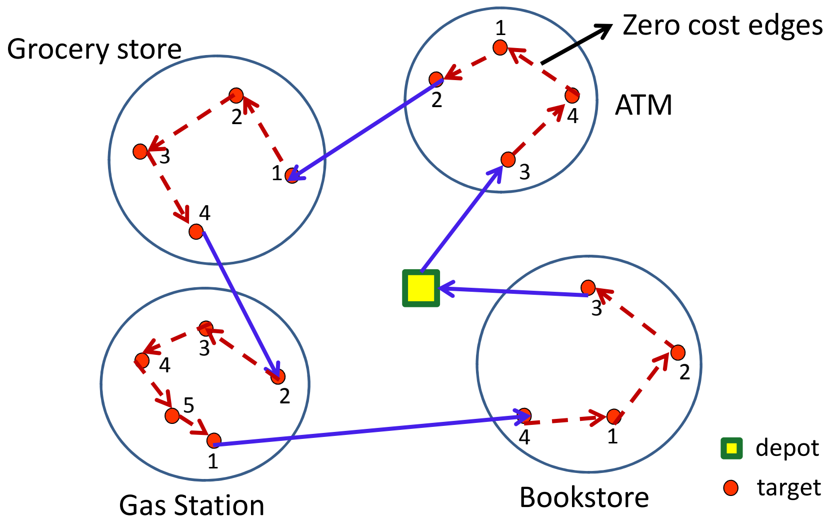


Figure 3: Construction of a transformation from one-in-a-set ATSP to a single ATSP: the corresponding solution to the ATSP.

bound for the one-in-a-set ATSP, then clearly one could not have more than $(n+1)$ edges in the optimal solution to the single ATSP connecting vertices in different groups; if it were the case, then the optimal solution would cost more than $(n+2)M$. However, the cost of the feasible single ATSP solution corresponding to the optimal one-in-a-set ATSP would be less than $(n+2)M$. Clearly, with this modification, every optimal solution to the single ATSP would have exactly $n+1$ edges connecting vertices from different groups.

Suppose one finds an optimal solution to the single ATSP; in that case, one can now recover a feasible solution to the one-in-a-set ATSP by identifying the sequence of vertices first visited in each group. The cost of the optimal solution (say C^*) to the single ATSP is exactly $(n+1)M$ more than the cost of the feasible solution to the one-in-a-set ATSP. This feasible solution to the one-in-a-set ATSP must also be optimal; otherwise, one can construct a feasible solution to the single ATSP with a cost smaller than C^* using the construction we have described.

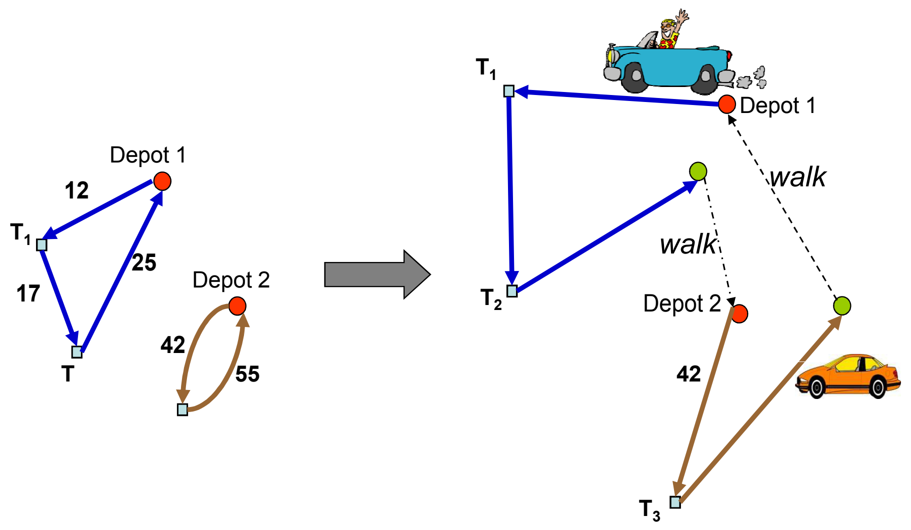


Figure 4: Transforming a feasible solution to MDTSP to a tour for a cab driver problem.

3.3 Transforming the MDTSP to a Single ATSP

From the results in the previous subsection, it is sufficient to show that one can transform the MDTSP to a one-in-a-set ATSP to achieve the objective. For this purpose, consider a cab driver who takes the first vehicle from the depot, drives through the assigned set of targets, returns to the depot, and parks the vehicle before walking to the depot where the second vehicle needs to be taken out and driven through the targets assigned to it. This procedure continues until the last vehicle is parked. After the last vehicle is parked, the driver walks back to the first depot. Instead of viewing the problem as a routing problem of multiple heterogeneous vehicles, one may view it as a problem of routing the cab driver. We replicate the depot to get a corresponding terminal. The cab driver may be thought of as taking the vehicle from the depot, visiting all the targets that the vehicle is assigned to visit, and parking at the corresponding terminal before walking to the next depot, as shown in Figure 4. In order to ensure that the cost does not change for both problems, one can set the cost of the cab driver walking from a terminal to the next depot to be zero. Please refer to Figure 4, which illustrates this idea for two vehicles and three targets.

Even though the cab driver problem concerns a single driver, it is still not a single ATSP because the cost of driving from one target to another depends on the vehicle used. In order to overcome this difficulty, one can replicate targets such that there is exactly one replica corresponding to each vehicle for each target (refer to Figure 5). In the example illustrated in the Figure, as there are two vehicles, there are two replicas created for each target. Let a target group represent the set of all the replicas corresponding to a given target. One can construct a graph (say $\mathcal{G}_1$) where there are target groups, the depots, and their corresponding terminals. Suppose one were to consider the replicated targets, depot, and terminal corresponding to the i^{th} vehicle; it seems natural to assign the cost of corresponding edges to be those of the i^{th} vehicle. As before, one can set the cost of the cab driver walking from a terminal to the next depot to zero. In this graph, $\mathcal{G}_1$, there are no other edges. One can now pose the cab driver problem as the following one-in-a set ATSP on $\mathcal{G}_1$: find an optimal tour on $\mathcal{G}_1$ such that exactly one vertex is visited from each of the target groups. A feasible solution to the one-in-a-set ATSP is shown in the left subfigure of Figure 5.

With the Noon-Bean transformation described in the previous subsection, it is therefore possible to convert a feasible solution to the MDTSP in Figure 4 to a feasible solution to the single ATSP, as illustrated in Figure 5.

The basic idea associated with the generalization of the MDTSP also involves another application of the Noon-Bean Transformation. Suppose the cost of traveling from an i^{th} target to a j^{th} target for the k^{th} vehicle also depends on the approach angles at the i^{th} and j^{th} targets. In this case, one can discretize the approach angles and associate an i^{th} group of vertices at the i^{th} target where the vehicle must pick exactly one vertex from the group (i.e., one approach angle from the group). The remaining part of this section summarizes these basic ideas discussed above mathematically.

Let the new set of target vertices replicated for vehicle i be denoted by $T_1^i, \dots, T_n^i$ (refer to the problem formulation in section 3.1 for notations). Also, let V_i^d be the replica of the depot vertex (referred to as the terminal vertex) corresponding to vertex V_i. For each $k \in \{1, \dots, n\}$, T_k^i is the replicated-target of T_k for vehicle i. Vehicle i is allowed to *only* visit replicated-targets in $\{T_1^i, \dots, T_n^i\}$. The cost of traveling from target T_j^i to target T_k^i for vehicle i is $C^i(T_j, T_k)$ for all $j, k \in \{1, \dots, n\}$. If a target, T_k^i, is visited by vehicle i, then it is required that none of the targets in the set $\{T_k^j : j \in \{1, \dots, m\} \setminus \{i\}\}$ be visited by any of its corresponding vehicles.

There are essentially $n + 2$ vertices denoted by $V_i, V_i^d, T_1^i, \dots, T_n^i$ corresponding to vehicle i. Since there are m vehicles, the new transformed graph has $m(n + 2)$ vertices. Let M be an upper bound on the optimal cost of the MDTSP. A simple upper bound is $2(n + m) \max_{i=1..m} \max_{a,b \in V_i \bigcup T} C^i(a, b)$. The edges in the new transformed graph and their corresponding costs are specified as follows:

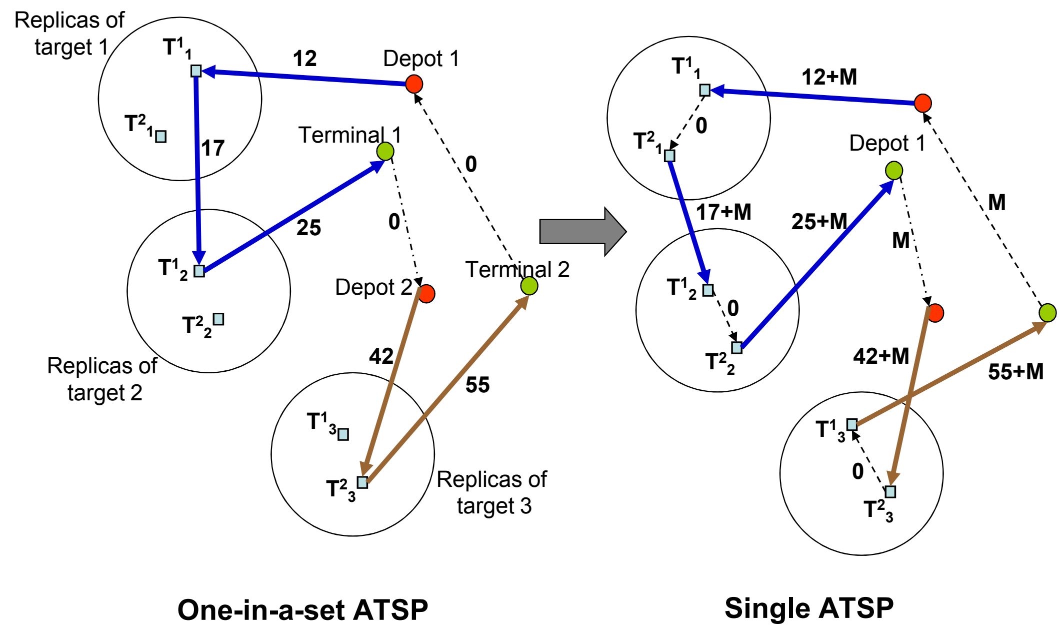


Figure 5: Transforming a feasible solution for the one-in-a set ATSP tour to a feasible solution to the single ATSP.

$$\widehat{C}(V_i, T^i_k) = C^i(V_i, T_k) + M, \text{ for all } i \in \{1, \dots, m\}, k \in \{1, \dots, n\} \tag{5}$$

$$\widehat{C}(V_i, V^d_i) = M, \text{ for all } i \in \{1, \dots, m\} \tag{6}$$

$$\widehat{C}(V^d_i, V_{i+1}) = 0, \text{ for all } i \in \{1, \dots, m-1\} \tag{7}$$

$$\widehat{C}(V^d_m, V_1) = 0$$

$$\widehat{C}(T^i_j, T^{i+1}_k) = C^{i+1}(T_j, T_k) + M, \text{ for all } i \in \{1, \dots, m-1\}, j, k \in \{1, \dots, n\}, j \neq k \tag{8}$$

$$\widehat{C}(T^m_j, T^1_k) = C^1(T_j, T_k) + M, \text{ for all } j, k \in \{1, \dots, n\}, j \neq k \tag{9}$$

$$\widehat{C}(T^i_j, T^{i+1}_j) = 0, \text{ for all } i \in \{1, \dots, m-1\}, j \in \{1, \dots, n\} \tag{10}$$

$$\widehat{C}(T^m_j, T^1_j) = 0, \text{ for all } j \in \{1, \dots, n\} \tag{11}$$

$$\widehat{C}(T^i_j, V^d_{i+1}) = C^{i+1}(T_j, V_{i+1}) + M, \text{ for all } i \in \{1, \dots, m-1\}, j \in \{1, \dots, n\} \tag{12}$$

$$\widehat{C}(T^m_j, V^d_1) = C^1(T_j, V_1) + M, \text{ for all } j \in \{1, \dots, n\} \tag{13}$$

An edge does not exist in the transformed graph if it is not assigned a cost in Equations (5) to (13).

3.4 Computational Results

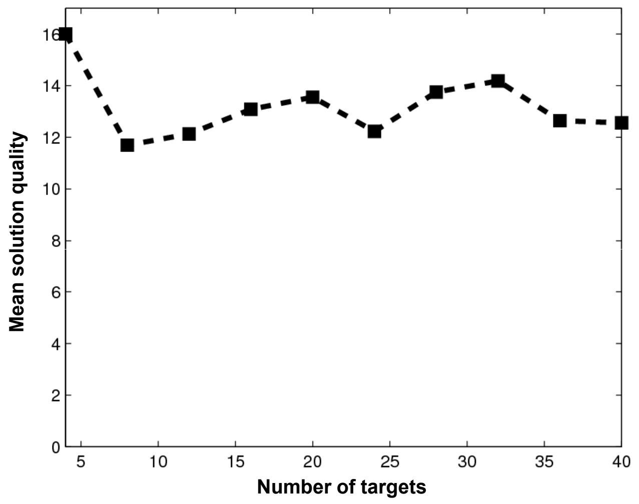
In the first set of simulations for the MDTSP, the number of vehicles were fixed to be equal to 6 and the number of targets were allowed to vary from 4 to 40. Each of the vehicles was different from the others, and the costs were assigned in an arbitrary manner. Each vehicle was assigned a subset of targets (generated randomly) that the vehicle could not visit. In essence, heterogeneity among vehicles was introduced in two ways: the Dubins' distance between any two targets depended on the minimum turning radius of a vehicle, and the initial vehicle-target assignment determined whether a vehicle could visit a target or not.

For a given number of vehicles (m) and targets (n), 100 instances were randomly generated. An upper bound on the solution quality of an instance I is defined as

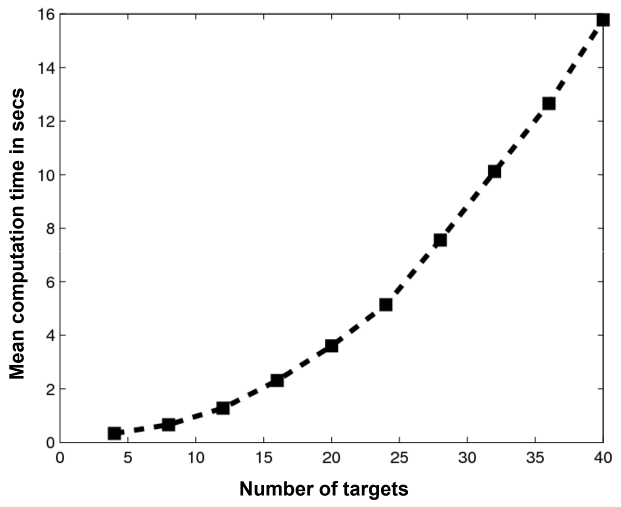
$$100\left(\frac{C^I_{LKH} - C^I_{LB}}{C^I_{LB} - (n+m)M}\right) \tag{14}$$

where C^I_{LKH} is the cost of the solution obtained by applying the LKH heuristic on the transformed graph, and C^I_{LB} is the lower bound for the single ATSP on the transformed graph. The LKH program by Helsgaun, available at http://www.akira.ruc.dk/keld/research/LKH/, was used to solve the ATSP. The LKH program was run without changing any of its default settings. Also, all the simulations were run on a Pentium 4 CPU with 3GHz processing power and 1.24 GB RAM.
The results regarding the average upper bounds on the quality of solutions and their computation times for the first set of simulations are shown in Figure 6(a) and Figure 6(b), respectively. For the range of targets considered in the simulations, the results show that the upper bound on the quality of the solutions was 16% on average. As expected, the mean computation times increased with the number of targets.

Similarly, in the second set of simulations for the MDTSP, the number of targets was fixed at 40 and the number of vehicles varied from 2 to 10. The minimum turning radius (r) of all the vehicles in the simulations was chosen to vary uniformly from 100 to 200 meters. The results regarding the mean solution quality and the computation times for the second set of simulations are shown in Figure 7(a) and Figure 7(b) respectively. Considering that a heterogeneous, asymmetric, multiple depot, multiple vehicles routing problem is a difficult problem to solve, these results indicate that the approach given in the paper is promising. However, the limitation of the approach proposed in this article is that the transformed graph contains a large number of vertices (i.e., $(n+2)m$ vertices) where nm is the number of targets (vehicles). If there are a large number of vehicles present in an instance, finding good tours in the transformed graph could still be a computationally challenging problem.

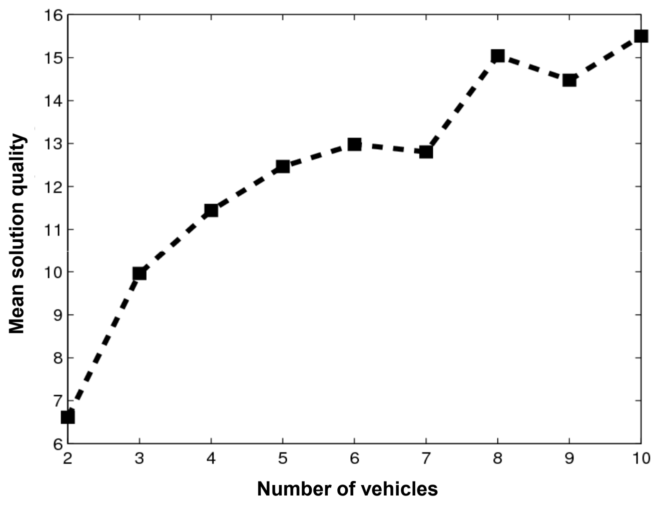


(a)

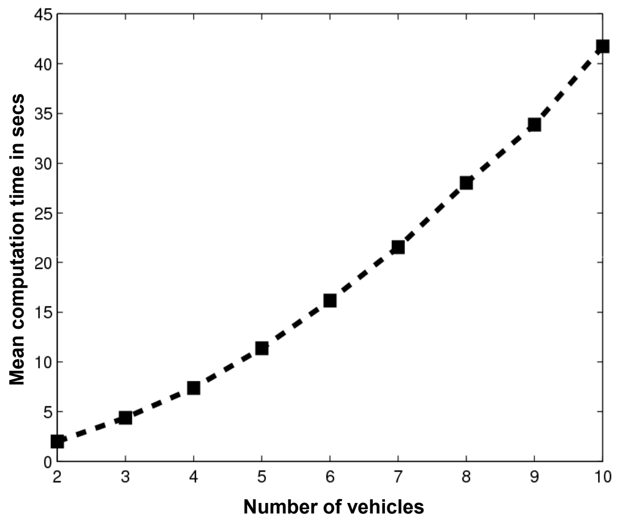


(b)

Figure 6: Computational results for MDTSP: number of vehicles fixed.



(a)



(b)

Figure 7: Computational results for MDTSP: number of targets fixed.

4 Precedence Constrained ATSP

We begin with a mathematical formulation of the precedence constrained ATSP in the following subsection. This mathematical formulation is helpful in finding feasible solutions, if not optimal solutions, and enables us to bound the suboptimality of the feasible solutions thus found.

4.1 PCATSPxy Formulation

The PCATSP formulation used in this report was proposed by Sarin, Sherali, and Bhootra in [24] and presented as follows: let the set of nodes be N, where $|N| = n$ and the depot is defined as node number 1. Let the set P be defined as the set of pairs (i, j) such that $i < j$. The binary variable x is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if node i immediately precedes node j} \\ 0 & \text{otherwise} \end{cases}$$

The continuous variable y is defined as:

$$y_{ij} = \begin{cases} 1 & \text{if node i precedes node j (not necessarily immediately)} \\ 0 & \text{otherwise} \end{cases}$$

The problem may be formulated as:

$$min \sum_{\substack{i,j=1 \\ i\neq j}}^{n} c_{ij} x_{ij}$$

subject to

$$\sum_{\substack{j=1 \\ i\neq j}}^{n} x_{ij} = 1 \quad \forall i \in N \tag{15}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \quad \forall j \in N \tag{16}$$

$$y_{ij} \geq \ x_{ij} \quad i, j \in \{2, \ldots n\}, i \neq j \tag{17}$$

$$y_{ij} + y_{ji} = 1 \quad i, j \in \{2, \ldots n\}, i \neq j \tag{18}$$

$$y_{ij} + y_{jk} + y_{ki} + x_{ji} \leq 2 \quad i, j, k \in \{2, \ldots n\}, \quad i \neq j \neq k \tag{19}$$

$$y_{ij} = 1 \quad \forall (i, j) \in P \tag{20}$$

$$x_{ij} \in [0\ 1] \tag{21}$$

Constraints (15) and (16) are the assignment constraints for each depot, constraints (17)-(19) are the sub-tour elimination constraints, and constraint (20) forces the precedence constraints. In [24], the authors give computational results, which show the relative tightness of the LP relaxation of the PCATSPxy formulation in respect to the LP relaxations of other IP formulations of the PCATSP. For this reason, we have chosen the PCATSPxy for use in the methods presented in this report.

4.2 PCATSP: Numerical Results

Solving the PCATSP to optimality can take a large amount of time. For this reason, we are interested in fast algorithms that can iteratively produce increasingly better solutions. If the problem may be solved using an iterative algorithm, it is possible to terminate the algorithm before it has reached optimality to obtain a feasible solution. If a lower bound to the optimal solution is also available, the quality of the solution can be quantified by its deviation from the lower bound. If this algorithm is convergent, it allows the user to specify whether a quick solution is preferred at the cost of quality or a certain quality of solution is required irrespective of the computation time. With this motivation, we present our algorithm for the PCATSP in the following sections.

Cut generation

One strategy to solve a difficult problem is to remove constraints that make the problem difficult, forming a new problem that is easy to solve. The easy problem may then be solved and checked for feasibility for the original problem. Solutions to the easy problem that are infeasible for the original problem are then removed from the set of feasible solutions to the easy problem by adding a constraint. This process can then be repeated iteratively until a feasible solution to the original problem is found. These inequalites are called cuts because they cut off infeasible solutions from the set of feasible solutions. For the PCATSP, the precedence constraints may be removed, yielding a standard ATSP problem. The standard ATSP may then be solved, and the solution may be evaluated for feasibility. If the ATSP solution is infeasible for the PCATSP, then two types of cuts are introduced. These cuts are known as the W-cut and predecessor-successor cuts and are described in the following subsections.

W-cut

The W-cut is proposed in [19] and separates solutions that violate precedence constraints. If there is a precedence relation that i must precede j, then the W-cut separates solutions in which there is a path from j to i. An illustration of the formation of this cut is shown in Figures 8 and 9.

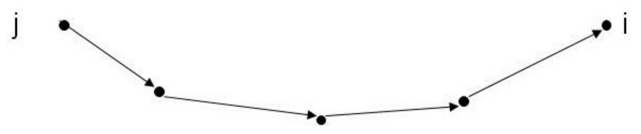


Figure 8: Edge subset in an infeasible solution.

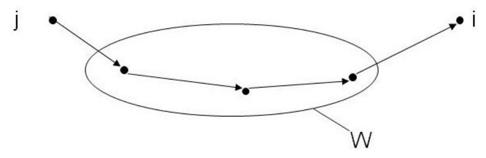


Figure 9: W set in an infeasible solution.

Consider, as above, a precedence relation that node i must precede node j. Now consider a solution to the ATSP containing the following subset of edges. Next, form the set W with the nodes that lie in the path from j to i, as indicated in Figure 9. Now that W is formed, an inequality that is valid for the PCATSP can be formed by adding the number of edges that have one end in W and the number of edges with both ends in W. This number should be less than the number of nodes that belong to W. This inequality may be expressed as follows:

$$\sum_{k \in W} x_{jk} + \sum_{(l,m) \in E(W)} x_{lm} + \sum_{p \in W} x_{pi} \leq |W|$$

This inequality is visualized in Figure 10. The addition of this inequality to the ATSP will

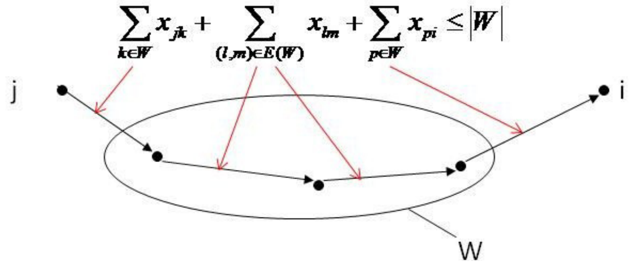

$\sum_{k \in W} x_{jk} + \sum_{(l,m) \in E(W)} x_{lm} + \sum_{p \in W} x_{pi} \leq |W|$
j
i
W

Figure 10: Edges dictating the W Cut.

result in the separation of the previous solution.

Predecessor-Successor Cuts

The predecessor-successor cuts rely on the perspective of edge variables as a capacity of flow. In other words, if the edge variable representing the edge from some node i to some other node j is represented as x_{ij}, then a solution in which the variable $x_{ij} = 1$ could be interpreted as the edge from i to j having the capacity to carry one unit of flow from i to j. In this context, we can construct the predecessor-successor Cuts. First, we can define a set π_i, which is the set of nodes that must precede node i and the set σ_i, which is the set of nodes that must succeed node i. In order to form these sets, let P be the set of precedence constraints. In other words, if node i must precede node j then $(i, j) \in P$. Without any loss of generality, we assume that the set P is consistent (satisfies the transitivity relation), meaning if $(i, j) \in P$ and $(j, k) \in P$, then $(i, k) \in P$; otherwise, there cannot be a solution to the problem considered here. Let V denote the set of nodes and let the depot be node 1. Then π and σ are formed as follows:

$$\pi_j := \{i \in V/\{1\} : (i, j) \in P\}$$

$$\sigma_i := \{j \in V/\{1\} : (i, j) \in P\}$$

After these sets are formed, we can find three cuts by using the following logic:

1. For some node i such that $(i, j) \in P$, then every feasible solution to the PCATSP should contain a path from node 1 to node i with a flow capacity of 1 that does not include any nodes $j \in \sigma_i$.

2. For some nodes (i, j) such that $(i, j) \in P$, then every feasible solution to the PCATSP should contain a path from node i to node j with a flow capacity of 1 that does not include any nodes $k \in \pi_i \cup \sigma_j \cup \{1\}$.

3. For some node j such that $(i, j) \in P$, then every feasible solution to the PCATSP should contain a path from node j to node 1 with a flow capacity of 1 that does not include any nodes $i \in \pi_j$.

Since the edge variables are considered in the context of flow, the cuts to be added can be generated using max-flow, min-cut algorithms. In practice, the cuts are generated in the following way:

1. For property (1), considering node i such that $(i, j) \in P$, let $V^* := V \backslash \sigma_i$. Form two sets, A and B, such that $V^* \backslash A = B$, $1 \in A$, $i \in B$, and $\sum_{l \in A, m \in B} x_{lm} < 1$. Then the cut is $\sum_{l \in A, m \in B} x_{lm} \geq 1$. In other words, A is the set of nodes connected to node 1 such that the sum of the edges that start at a node in A and end in B is less than 1. This algorithm is in Figure 11.
   First, we have a feasible solution to the relaxation of the ATSP, which will be evaluated for feasibility given the constraint that node i must precede node j. Next remove the successors of node i, or in this case, just node j. Now, from Figure 12, it can be seen that one unit of flow cannot be passed from the depot to node i along the path that is highlighted. This means a cut has been identified, and we must construct the two sets, A and B. Set A is constructed such that it contains the depot, and the sum of the edges starting in A and ending in B are less than 1. These two sets are labeled below in Figure 13. In this example, then, a cut is generated and added to the ATSP relaxation requiring the sum of the flow capacities over the edges that start in set A and end in set B to be at least 1. Note that sets A and B are not necessarily unique.

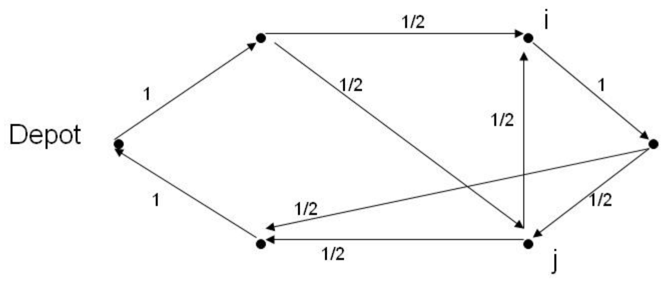


Figure 11: LP feasible solution.

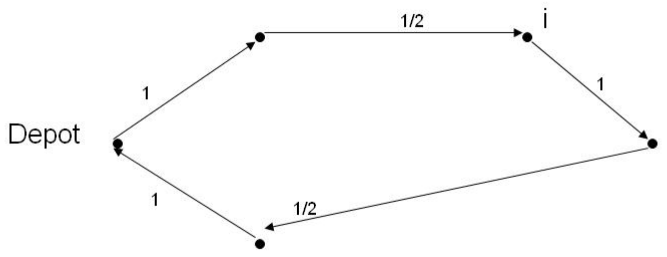


Figure 12: Remove members of σ_i.

2. For property (2), considering nodes (i, j) such that $(i, j) \in P$, let $V^* := V \backslash \pi_i \cup \sigma_j \cup \{1\}$. Form two sets, A and B, such that $V^* \backslash A = B$, $i \in A$, $j \in B$, and $\sum_{l \in A, m \in B} x_{lm} < 1$. Then the cut is $\sum_{l \in A, m \in B} x_{lm} \geq 1$. In other words, A is the set of nodes connected to node i such that the sum of the edges that start at a node in A and end in B is less than 1. This algorithm is similar to the procedure presented graphically in Figures 11-13.

3. For property (3), considering node j such that $(i, j) \in P$, let $V^* := V \backslash \pi_j$. Form two sets A and B such that $V^* \backslash A = B$, $j \in A$, $1 \in B$, and $\sum_{l \in A, m \in B} x_{lm} < 1$. Then the cut is $\sum_{l \in A, m \in B} x_{lm} \geq 1$. In other words, A is the set of nodes connected to node j such that the sum of the edges that start at a node in A and end in B is less than 1. This algorithm is similar to the procedure presented graphically in Figures 11-13.

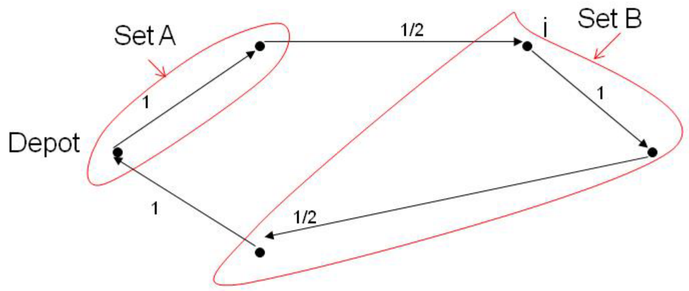


Figure 13: Set A and set B.

4.3 Split Dual

Consider the following formulation for the PCATSP:

$$\min \sum_{i,j} c_{ij} x_{ij}, \tag{22}$$

subject to

$$x \in \text{ATSP}, \quad x \in \text{relaxed PCATSPxy}$$

By $x \in$ ATSP, we mean that x satisfies the DFJ constraints and $x \in$ relaxed PCATSPxy, we mean that z satisfies the constraints of the PCATSP formulation due to Sarin, Sherali and Bhootra with the integrality constraint removed. Since the two subproblems

- $\min \sum_{i,j} c_{ij} x_{ij}$ subject to $x \in$ ATSP, and
- $\min \sum_{i,j} c_{ij} x_{ij}$ subject to $x \in$ relaxed PCATSPxy,

have been dealt with in the literature and have distinct features, the first one describing the sub-tour elimination constraints tightly and the second one describing the precedence constraints elegantly, we want to combine the best of the features by considering (22). Clearly one may express (22) as:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{23}$$

subject to

$$x \in \text{ATSP}, \quad z \in \text{relaxed PCATSPxy}$$

$$x = z \tag{24}$$

Let μ_{ij} be the penalty for the violation of constraint $x_{ij} = z_{ij}$. If constraint (24) were penalized, one gets the following split dual:

$$\min \left(\sum_{i,j} c_{ij} x_{ij} + \sum_{i,j} \mu_{ij} \left(z_{ij} - x_{ij} \right) \right)$$

subject to

$$x \in \text{ATSP} \quad z \in \text{relaxed PCATSP}$$

The split dual objective function can be rewritten as:

$$\min \sum_{i,j} (c_{ij} - \mu_{ij})\, x_{ij} + \min \sum_{i,j} \mu_{ij} z_{ij}$$

Since the first term is the ATSP with the cost associated with edge (i, j) being $c_{ij} - \mu_{ij}$, it can be solved using a standard TSP solver. The second term is the LP relaxation of the PCATSPxy, so it can be solved using a standard LP solver. This process is iterated and the dual variables μ can be updated using Polyak's rule. Table 1 shows the results of the split-dual algorithm implemented in C++ for the case of four precedence constraints. The ATSP was solved using the LKH algorithm written by Helsgaun in [12]. The LP relaxations and integer PCATSPxy were solved using GNU linear programming kit(GLPK). The time taken to compute the integer program (IP) including whether the solver iteration limit was exceeded is indicated under the IP Time column. At each iteration, an upper bound was generated by applying a simple heuristic to the optimal tour in order to satisfy the precedence constraints. We observed that the algorithm terminated after 200 iterations, or if the duality gap fell under 1% for the instances that we tried. In the table, n represents the number of targets considered, and SP LB, SP UB, and D Gap denote the split dual lower bound, upper bound, and duality gap, respectively. The last column denotes the split dual solution cost minus the IP solution cost divided by the IP solution cost. If the IP solution exceeded the solver iteration limit, then this value was DNS, or did not solve.

While the duality gap on average was greater than 10%, the gap between the primal feasible upper bound and the IP optimal solution was less than 5% on average. As can be seen in Table 1, the lower bound formed by the LP relaxation converged slowly relative to the upper bound, which caused a large duality gap. For this reason, after each iteration, W-cuts and predecessor-successor cuts were added to the LP relaxation to speed up convergence. The result of adding these cuts can be seen in Table 2. The algorithm was stopped after 25 iterations or if the duality gap fell below 5%. In Table 2, n is the number of targets considered and $|P|$ is the number of precedence constraints. The next two columns refer to the result of the split dual algorithm with no cuts applied to the lower bound. The column with the title D-gap is the duality gap resulting from the split dual algorithm with no cuts applied to the lower bound, and Time is the time in seconds to run the split dual algorithm with no cuts applied to the lower bound. The final two columns refer to the result of the split dual algorithm with cuts applied to the lower bound. The column with the title Cut D-gap is the duality gap resulting from the split dual algorithm with cuts applied, and Cut Time is the time in seconds to run the split dual algorithm with cuts applied to the lower bound. Fields with a DNS denote a problem whose IP solution exceeded the (very adequate) time or iteration limit set in GLPK; therefore, there is no available comparison to the actual IP optimal solution. All values below are averaged from five randomly generated instances of the problem with the labeled number of targets and precedence constraints. From Table 2, it can be seen that adding cut generation is successful in closing the duality gap, albeit with a penalty in solution time. In situations where the IP optimum is unavailable, adding cut generation can drastically reduce the duality gap.

Table 1: Results using split dual.

SP LB	SP UB	D-gap	SD Time	IP Time	(SD-IP)/IP
n=15					
3448.98	3480	0.01	13.48	8.94	0.00
3641.64	4448	0.22	80.62	251.00	DNS
4457.65	4531	0.02	46.18	114.80	0.00
4042.67	4477	0.11	59.35	6.65	0.05
3993.73	4929	0.23	70.33	250.00	DNS
4207.27	4255	0.01	51.91	19.36	0.00
4500.93	5343	0.19	68.38	124.00	0.05
4554.10	5233	0.15	42.28	103.36	0.01
4011.53	4220	0.05	76.35	8.00	0.04
4062.03	4500	0.11	97.09	8.68	0.10
	AVG	0.11	60.60	89.48	0.04
n=16					
3405.21	4186	0.23	121.26	353.00	DNS
4448.75	4659	0.05	73.90	86.00	0.01
4308.65	5167	0.20	113.68	18.44	0.08
4314.75	4864	0.13	105.47	67.08	0.04
4222.07	5286	0.25	133.10	533.00	DNS
	AVG	0.17	109.48	211.50	0.04

An illustration of the outputs of this algorithm are shown for a 10-node problem with the partial ordering $4 < 5 < 6 < 7$. Figure 14 shows the LP solution, which gives the lower bound when the algorithm is terminated. The dotted lines represent fractional edge variables. The asterisks represent arrow heads, as all the edges are directed. Figure 15 shows the feasible solution generated at the iteration when the algorithm is terminated. As described previously, the asterisks represent arrowheads that point in the direction of the edge.

5 Vehicle Routing with Time Window Constraints

5.1 Single Vehicle Case without Time Windows

First, we formulate this problem as a TSP with precedence constraints. There are various formulations for the precedence constrained TSP. In [30], Sarin, Sherali, and Bhootra proposed a formulation with computational results that show the relative tightness of the LP relaxation of the PCATSPxy formulation in respect to the LP relaxations of other IP formulations of the PCATSP. The formulation is as follows: Let the set of nodes be N, where $|N| = n$ and

Table 2: Split dual with cut generation.

n	$\|P\|$	D-gap	Time	Cut D-gap	Cut Time
10	2	0.056	2.7	0.039	2.7
10	3	0.111	3.5	0.050	4.6
10	4	0.128	4.2	0.043	5.4
10	5	0.075	3.7	0.040	5.2
15	2	0.230	36.6	0.098	54.4
15	3	0.114	34.6	0.089	51.9
15	4	0.137	30.9	0.096	44.5
15	5	0.167	31.9	0.106	45.9
20	2	0.373	291.2	0.228	350.1
20	3	0.200	300.6	0.138	362.7
20	4	0.287	272.8	0.251	338.1

the depot is defined as node number n. The binary variable $x_{i,j}$ is defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if node } i \text{ immediately precedes node } j \\ 0 & \text{otherwise} \end{cases}$$

The continuous variable y is defined as:

$$y_{i,j} = \begin{cases} 1 & \text{if node } i \text{ precedes node } j \text{ (not necessarily immediate)} \\ 0 & \text{otherwise} \end{cases}$$

Let the distance between two nodes, i and j, be $c_{i,j}$; then, the problem is formulated as follows:

$$\min \sum_{\substack{i,j=1 \\ i \neq j}}^{n} c_{i,j} x_{i,j} \tag{25}$$

subject to

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{i,j} = 1,\ \forall i \in N \tag{26}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{i,j} = 1,\ \forall j \in N \tag{27}$$

$$y_{i,j} \geq x_{i,j},\ i,j \in \{1,2,\ldots,n-1\}, i \neq j \tag{28}$$

$$y_{i,j} + y_{j,i} = 1,\ i,j \in \{1,2,\ldots,n-1\}, i \neq j \tag{29}$$

$$y_{i,j} + y_{j,k} + y_{k,i} + x_{j,i} \leq 2, i,j,k \in \{1,\ldots,n-1\}, i \neq j \neq k \tag{30}$$

$$y_{i,j} = 1,\ \forall j \in \{1,2,\ldots,n-1\}, \forall i \in P_j \tag{31}$$

$$x_{i,j} \in [0,1], \forall i,j \in N \tag{32}$$

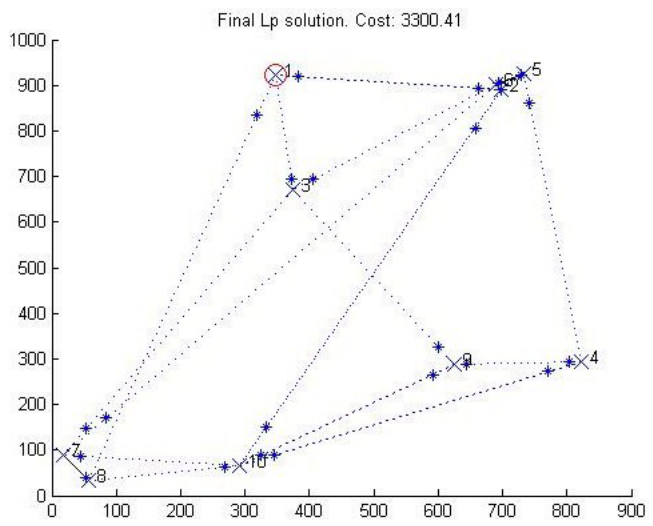


Figure 14: Lower bound generated by split dual.

Constraints (26) and (27) assure that each node has exactly one in-degree and out-degree. Constraints (28) to (30) restrict the generation of sub-tour in the tour. In constraint (31), P_j is a subset of nodes in $\{1, 2, \ldots, n-1\}$ that are required to precede city j in the Hamiltonian path that commences at the base city n. Constraint (32) is a binary constraint. The validity of this formulation is provided in [30].

Now, we reformulate this formulation for the single vehicle paratransit service problem. Each service request is composed of locations and time windows: where and when a customer wants to be picked up and dropped off, respectively. Assume that a service provider receives n service requests; then there are $2n$ nodes for a tour. Node i represents the pickup node for request i, and node $n+i$ represents the drop-off node for request i. Let set N_p be a set of the pickup nodes and set N_d be a set of the drop-off nodes; then set N will be a union set of two sets, $N = N_p \cup N_d$. The cardinality of the set N is $2n$. Let the depot be at node $2n+1$. The constraints (33) to (38) can be rewritten as the following:

$$\sum_{\substack{j=1 \\ i \neq j}}^{2n+1} x_{i,j} = 1,\ \forall i \in N \cup \{2n+1\} \tag{33}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{2n+1} x_{i,j} = 1,\ \forall j \in N \cup \{2n+1\} \tag{34}$$

$$y_{i,j} \geq x_{i,j},\ i,j \in N, i \neq j \tag{35}$$

$$y_{i,j} + y_{j,i} = 1,\ i,j \in N, i \neq j \tag{36}$$

$$y_{i,j} + y_{j,k} + y_{k,i} + x_{j,i} \leq 2, i,j,k \in N, i \neq j \neq k \tag{37}$$

$$y_{i,n+i} = 1,\ \forall i \in N_p \tag{38}$$

Constraint (38) denotes that node i, which is a pickup node of i^{th} request, precedes to node $n+i$, which is a drop-off node of the same request.

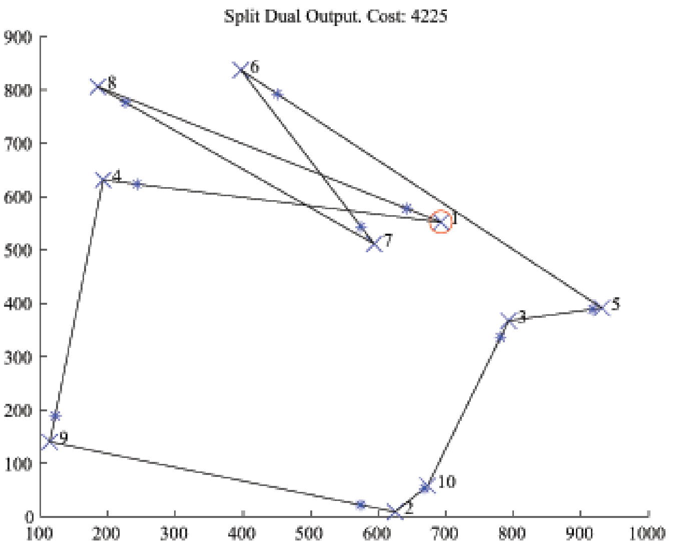


Figure 15: Feasible solution generated by split dual.

5.2 Extension to the Case of Multiple Vehicles

Hereafter, we consider that m vehicles depart their tours from different depots and come back to the depots. For the given delivery requests, one wants to find m tours corresponding to each vehicle and accomplishing all the requests.

To find the solution to the paratransit scheduling problem for multiple vehicles, we utilize the PCATSPxy formulation from the previous section. Basically, we reformulate the multiple vehicle case to a single vehicle case. The first vehicle starts its tour from its depot and returns to a depot of the second vehicle after it finishes its delivery tasks. The second vehicle starts its own depot and returns to a depot of the third vehicle, and so on. Let us index the depots starting at $2n+1$ to $2n+m$, and for a return node for the $(2n+m)^{\text{th}}$ vehicle, consider an additional node indexed as $2n+m+1$. Let set D be a set of all those depots including the additional one, $D = \{2n+1, \ldots, 2n+m+1\}$; then the first vehicle departs from node $2n+1$ and returns to node $2n+2$, the second tour starts from node $2n+2$ and ends up at node $2n+3$, and the m^{th} tour starts from node $2n+m$ and returns to node $2n+m+1$. Since the actual vehicles return to their own depot, the cost matrix needs modifications. Let $d_{i,j}$ be a new cost matrix as follows:

$$d_{i,j} = \begin{cases} c_{i,j},\ \forall i,j \in N, i \neq j, j \neq i-n \\ c_{i,j},\ \forall i \in D \setminus \{2n+m+1\}, \forall j \in N_p \\ c_{i,j-1},\ \forall i \in N_d, \forall j \in D \setminus \{2n+1\} \\ 0,\ \forall i,j \in D \\ \infty \text{ or very large number, otherwise} \end{cases} \tag{39}$$

With this modified cost matrix, the PCATSPxy can be reformulated as the following:

$$\text{minimize} \sum_{\substack{i,j=1 \\ i \neq j}}^{2n+m+1} d_{i,j} x_{i,j} \tag{40}$$

subject to

$$\sum_{\substack{j=1 \\ i \neq j}}^{2n+m+1} x_{i,j} = 1,\ \forall i \in N \cup D \tag{41}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{2n+m+1} x_{i,j} = 1,\ \forall j \in N \cup D \tag{42}$$

$$y_{i,j} \geq x_{i,j},\ i,j \in N \cup D \setminus \{2n+1\}, i \neq j \tag{43}$$

$$y_{i,j} + y_{j,i} = 1,\ i,j \in N \cup D \setminus \{2n+1\}, i \neq j \tag{44}$$

$$y_{i,j} + y_{j,k} + y_{k,i} + x_{j,i} \leq 2, i,j,k \in N, i \neq j \neq k \tag{45}$$

$$y_{i,n+i} = 1,\ \forall i \in N_p \tag{46}$$

$$y_{i,j} = y_{i+n,j},\ \forall i \in N_p, \forall j \in D \setminus \{2n+1\} \tag{47}$$

$$y_{i,i+1} = 1,\ \forall i \in D \setminus \{2n+1\} \tag{48}$$

$$x_{i,j} \in [0,1], \forall i,j \in N \cup D \tag{49}$$

Constraint (46) is a constraint that if a customer is picked up by a vehicle, then the vehicle has to visit a drop-off node of the customer. Constraint (48) restricts that the tour of i^{th} vehicle starts its tour after the $(i-1)^{\text{th}}$ vehicle finishes its tour. This formulation provides m tours for the paratransit service without consideration of time window.

As far as we consider time window constraints, we assume that time windows of the requests are reasonable, which means, at least, the time difference between a pickup and a drop-off time for a request has to be larger than the traveling time from the pickup node directly to the drop-off node. We also assume that the average speed of the homogeneous vehicles is constant, S, and then the traveling time from node i to node j is $d_{i,j}/S$. Service time is relatively smaller than traveling time, so we do not consider the service time for each visit. Let t_i be the arrival time at node i, $i \in N$. Let $[e_i, l_i]$ denote the service time window for each pickup or delivery node $i, i \in \{1, 2, \ldots, 2n\}$. Let E_k denote the earliest start time for vehicle k and L_k denote the latest return time for vehicle k, $k \in \{1, 2, \ldots, m\}$, then constraints for the time windows can be written as follows:

$$t_i + \tfrac{d_{i,j}}{S} \leq t_j + K(1 - x_{i,j}), \forall i,j \in N, i \neq j \tag{50}$$

$$E_{i-2n} + \tfrac{d_{i,j}}{S} \leq t_j + K(1 - x_{i,j}), \forall i \in D, \forall j \in N_p \tag{51}$$

$$t_i + \tfrac{d_{i,j}}{S} \leq L_{j-1-2n} + K(1 - x_{i,j}), \forall i \in N_d, \forall j \in D \tag{52}$$

$$e_i \leq t_i \leq l_i, \forall i \in N \tag{53}$$

where K is a very large number. Constraint (50) ensures that the service at node i cannot be started until the vehicle finishes serving the immediate precedent node and travels to the current node. Constraint (51) forces the first service time for each vehicle assuming that vehicle i starts at time E_i. Constraint (52) ensures that no vehicle violates the time window in the return depot, and constraint (53) ensures that the service time is within the time window.

5.3 Numerical Results for Small Instances of Paratransit Problem

Now, we present a series of cases regarding the Euclidean graph. Table 3 shows the structure of request data, including coordinates and time of pickup and drop-off requests. p_{xi} and p_{yi} represent the coordinate of the pickup node of the i^{th} customer and so they are node i. Customer j's drop-off coordinates, d_{xi} and d_{yi}, will be the $(i+n)^{\text{th}}$ node for the formulation. We generate the hard time windows from p_{ti} and d_{ti} as $[e_i, l_i] = [p_{ti}, p_{ti} + \delta t], \forall i \in N_p$ and $[e_i, l_i] = [d_{ti} - \delta t, d_{ti}], \forall i \in N_d$ where δt is a time window constant we have chosen. In fact, since the time window constraints tighten the size of the feasible set, the computation time could be less than solving the same problem without the constraints. However, if the time window constant is too tight, then the feasible set can be empty. For example, if there are two customers who want to be picked up between 8:00 a.m. and 8:30 a.m., traveling time from one customer to the other takes more than 30 minutes, and only one vehicle is available to service them, then there will be no feasible schedule to satisfy both customers. So we assume that in this numerical example the time windows are wide enough to guarantee that there exists at least one feasible solution. For a numerical example, we consider there are 10 service requests and four homogeneous vehicles, $|N| = 24$, as shown in Figure 16.

Each pickup node (numbered blue circles in the figure) has a corresponding drop-off node (red circles). Each vehicle (black squares in the figure) starts from its own depot. Figure 17 represents a case without consideration of the time window constraints. In this case, the precedence constraints are only considered. Since the objective function is minimizing total traveling distance, the problem usually uses only one vehicle, in this case vehicle #1, unless some pairs of requests are located far from the others. Figure 18 shows the optimal tours for multiple vehicles with hard time window constraints. In this case, only three out of four vehicles are used to perform the paratransit service.

Table 3: Request data structure.

CUST.	PICKUP			DROP-OFF		
NO.	X	Y	TIME	X	Y	TIME
1	p_{x_1}	p_{y_1}	p_{t_1}	d_{x_1}	d_{y_1}	d_{t_1}
2	p_{x_2}	p_{y_2}	p_{t_2}	d_{x_2}	d_{y_2}	d_{t_2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	p_{x_n}	p_{y_n}	p_{t_n}	d_{x_n}	d_{y_n}	d_{t_n}

We ran several simulations with different instances with randomly generated data sets, as shown in Tables 5 and 6. Tables 5 and 6 provide information about the vehicles and

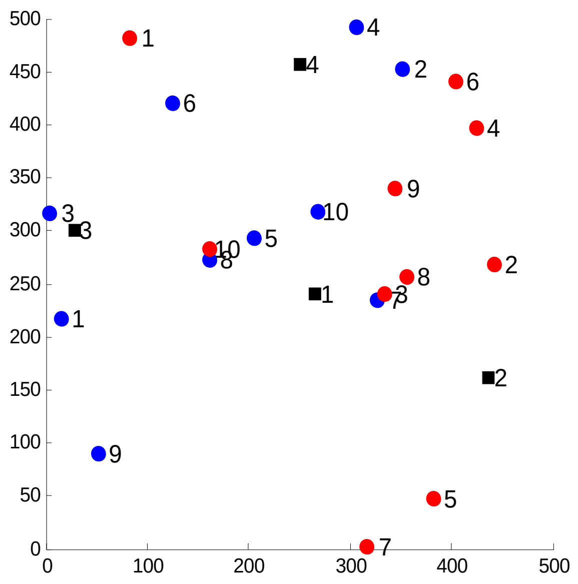


Figure 16: Example of request and vehicle nodes.

the requests, respectively. For n requests and k vehicles, we chose n requests from Table 5 and k vehicles from Table 6. We solved these mixed integer program (MIP) using CPLEX 12 and C++. The simulation results are shown in Table 4. Five different instances, $n = 5, 10, 15, 18, 20$, were tried and four were solved. For more than 19 requests, $n \geq 19$, the solver was stopped by out-of-memory error. Otherwise, we calculate the optimal tours for the paratransit services.

Table 4: Simulation results.

n	k	Total Cost	Com. Time (sec.)	Veh. Used	δt	spd.
5	3	254	0.06	2	2	50
10	3	413	8.01	2	2	50
15	3	607	1370.88	3	2	50
18	3	704	6699.2	3	2	50
20	3	N/A	N/A	N/A	2	50

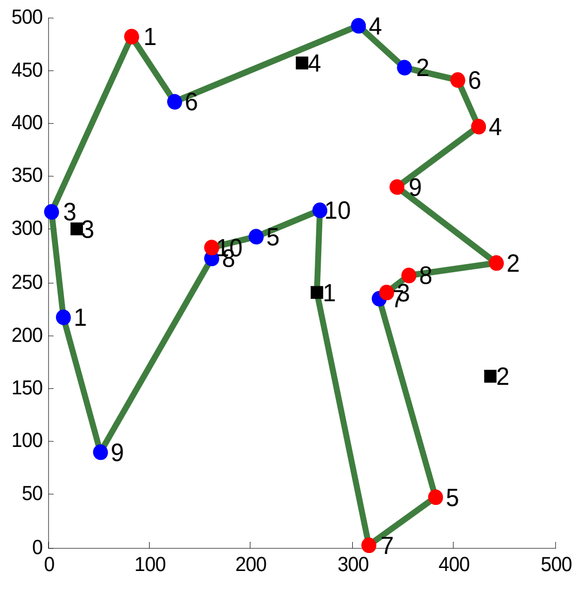


Figure 17: Tours without time window constraints

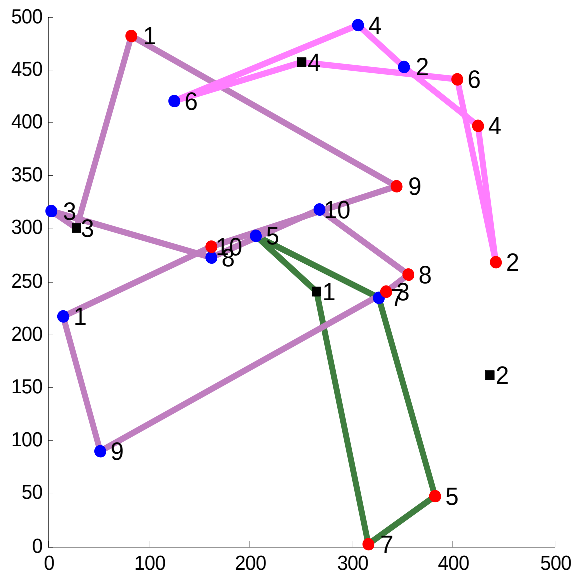


Figure 18: Tours with time window constraints

Table 5: Example request data set.

Req. Num.	Pickup			Drop-off		
	p_{x_i}	p_{y_i}	time	d_{x_i}	d_{y_i}	time
1	39	28	10	41	23	60
2	39	49	6	7	7	50
3	21	42	14	44	3	139
4	2	48	3	4	36	102
5	29	32	5	17	15	61
6	8	19	12	30	34	25
7	37	24	3	29	33	104
8	27	46	8	34	27	93
9	12	0	12	33	36	22
10	46	7	11	22	25	27
11	38	24	3	7	24	138
12	45	27	7	38	25	74
13	3	3	6	12	47	10
14	9	33	8	33	19	16
15	37	45	9	43	5	28
16	35	5	3	4	12	126
17	39	22	13	49	34	73
18	25	14	10	1	42	83
19	21	50	4	42	49	72
20	31	31	9	42	10	18

Table 6: Vehicle data.

Vehicle ID	x	y	E_i	L_i
1	49	19	0	140
2	28	46	0	140
3	43	34	0	140
4	20	19	0	140
5	23	32	0	140

References

[1] G. Berbeglia, J.F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: A classification scheme and survey. *Top*, 15(1):1–31, 2007.

[2] J.F. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1(2):89–101, 2003.

[3] S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems: Part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[4] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[5] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[6] H.N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3):351–357, 1983.

[7] J. Desrosiers, Y. Dumas, F. Soumis, S. Taillefer, and D. Villeneuve. An algorithm for mini-clustering in handicapped transport. *Cahiers du GERAD*, 1991.

[8] T.R. Sexton and L.D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science*, 19(4):411, 1985.

[9] J.F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, pages 573–586, 2006.

[10] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6(3):301–326, 1986.

[11] C.E. Noon and J.C. Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR-OTTAWA-*, 31:39–39, 1993.

[12] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[13] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[14] M. Padberg and T.Y. Sung. An analytical comparison of different formulations of the traveling salesman problem. *Mathematical Programming*, 52:315–357, 1992.

[15] AJ Orman and H.P. Williams. A survey of different integer programming formulations of the travelling salesman problem. *Optimisation, Econometric and Financial Analysis*, pages 91–104, 2007.

[16] A. Langevin, F. Soumis, and J. Desrosiers. Classification of traveling salesman problem formulations. *Operations Research Letters*, 9:127–132, 1990.

[17] M. Desrochers and G. Laporte. Improvements to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[18] H.D. Sherali and P. Driscoll. On tightening the relaxations of miller-tucker-zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 50(4):656–669, 1991.

[19] N. Ascheuer, L. Escudero, M. Groetschel, and M. Stoer. On identifying in polynomial time violated subtour elimination and precedence forcing constraints for the sequential ordering problem. *Integer Programming and Combinatorial Optimization*, pages 19–28, 1990.

[20] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68(3):241–265, 1995.

[21] N Ascheuer, L.F. Escudero, M. Grotschel, and M. Stoer. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). *SIAM Journal on Optimization*, 3:25–42, 1993.

[22] N. Ascheuer, M. Junger, and G. Reinelt. A branch and cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Computational Optimization and Applications*, 17(1):61–84, 2000.

[23] L. Gouveia and J.M. Pires. The asymmetric travelling salesman problem: On generalizations of disaggregated miller-tucker-zemlin constraints. *Discrete Applied Mathematics*, 112:129–145, 2001.

[24] S. C. Sarin, H. D. Sherali, and A. Bhootra. A precedence-constrained asymmetric traveling salesman model for disassembly optimization. *Iie Transactions*, 38(3):223–237, 2006.

[25] M. Bellmore and S. Hong. A note on the symmetric multiple travelling salesman problem with fixed charges. *Operations Research*, 25:871–874, 1977.

[26] S. Hong and M.W. Padberg. A note on the symmetric multiple traveling salesman problem with fixed charges. *Operations Research*, 25(5):871–874, 1977.

[27] MR Rao. A note on the multiple traveling salesmen problem. *Operations Research*, pages 628–632, 1980.

[28] R. Jonker and T. Volgenant. An improved transformation of the symmetric multiple traveling salesman problem. *Operations Research*, 36(1):163–167, 1988.

[29] Y. GuoXing. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. *European Journal of Operational Research*, 81(3):557–560, 1995.

[30] S. C. Sarin, H. D. Sherali, and A. Bhootra. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters*, 33(1):62–70, 2005.