PATH PLANNING ALGORITHMS FOR MULTIPLE HETEROGENEOUS VEHICLES

A Thesis

by

PAUL VICTOR OBERLIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2009

Major Subject: Mechanical Engineering

PATH PLANNING ALGORITHMS FOR MULTIPLE HETEROGENEOUS VEHICLES

A Thesis

by

PAUL VICTOR OBERLIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Darbha Swaroop
Committee Members,	Shankar Bhattacharyya
	Bryan Rasmussen
Head of Department,	Dennis O'Neal

May 2009

Major Subject: Mechanical Engineering

ABSTRACT

Path Planning Algorithms for Multiple Heterogeneous Vehicles. (May 2009)

Paul Victor Oberlin, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Darbha Swaroop

Unmanned aerial vehicles (UAVs) are becoming increasingly popular for surveillance in civil and military applications. Vehicles built for this purpose vary in their sensing capabilities, speed and maneuverability. It is therefore natural to assume that a team of UAVs given the mission of visiting a set of targets would include vehicles with differing capabilities. This paper addresses the problem of assigning each vehicle a sequence of targets to visit such that the mission is completed with the least "cost" possible given that the team of vehicles is heterogeneous. In order to simplify the problem the capabilities of each vehicle are modeled as cost to travel from one target to another. In other words, if a vehicle is particularly suited to visit a certain target, the cost for that vehicle to visit that target is low compared to the other vehicles in the team. After applying this simplification, the problem can be posed as an instance of the combinatorial problem called the Heterogeneous Travelling Salesman Problem (HTSP). This paper presents a transformation of a Heterogenous, Multiple Depot, Multiple Traveling Salesman Problem (HMDMTSP) into a single, Asymmetric, Traveling Salesman Problem (ATSP). As a result, algorithms available for the single salesman problem can be used to solve the HMDMTSP. To show the effectiveness of the transformation, the well known Lin-Kernighan-Helsgaun heuristic was applied to the transformed ATSP. Computational results show that good quality solutions can be obtained for the HMDMTSP relatively fast.

Additional complications to the sequencing problem come in the form of precedence constraints which prescribe a partial order in which nodes must be visited. In

this context the sequencing problem was studied seperately using the Linear Program (LP) relaxation of a Mixed Integer Linear Program (MILP) formulation of the combinatorial problem known as the "Precedence Constrained Asymmetric Travelling Salesman Problem" (PCATSP).

ACKNOWLEDGMENTS

Special thanks to the Air Force Research Laboratory (AFRL) Air Vehicles Directorate for providing funding and motivation for portions of this thesis, Waqar Malik for his Matlab codes that were implemented in some of the algorithms used in this thesis and Sivakumar Rathinam for guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Optimal routing of multiple vehicles is a common problem enountered in many applications. Variants of this problem differ based on the constraints imposed on routing. This thesis considers variants types of routing problem:

1. The Precedence Constrained Asymmetric Travelling Salesman Problem (PCATSP)
2. The Asymmetric Travelling Salesman Problem with vehicle motion constraints and the
3. Heterogeneous Travelling Salesman Problem (HTSP).

All these problems are generalizations of the Asymmetric Travelling Salesman Problem, which can be stated as follows: Given a set of nodes with one designated as a "depot", and the distance (cost) between every pair of nodes, what is the route of least cost such that every node is visited once and the route starts and ends at the depot. The PCATSP requires an additional constraint to be satisfied: There are some nodes that must be visited before other nodes. We will assume that the ordering constraints are consistent, i.e. they satisfy the transitivity relation. The ATSP with motion constraint requires the construction of a motion plan or trajectory of least distance (cost) so that the motion constraints of the vehicle are not violated. In this thesis, the motion of a vehicle is constrained by a bound on its yaw rate; we will assume that the vehicle is traveling at a constant speed. Under this assumption, the constraint on its yaw rate is equivalent to bounding the curvature of the trajectory of

---

The journal model is *IEEE Transactions on Automatic Control.*

the vehicle. The problem of optimization now has two inter-related components - the discrete or combinatorial component requires the sequencing of nodes to be visited and the continuous component requires the determination of the optimal heading at every node. The resulting optimization problem may be viewed as a mixed integer, nonlinear program.

The HTSP arises when dissimilar vehicle must be routed. It is conceivable that the cost of traveling from a specified node to another specified node may depend on the vehicle that is deployed. The problem here requires the partition of nodes to be visited by each vehicle in conjunction with the sequencing of the nodes.

As an intermediate step in studying these problems, the One-in-a-Set TSP is considered. The One-in-a-Set TSP may be described as follows: Given a collection of node sets and the cost between nodes of different sets, what is the least cost route through at least one node in each set in such a way that the route starts and ends at the same depot. The One-in-a-Set TSP is useful because a transformation exists that allows it to be solved as a standard TSP using available tools.

The PCATSP may be formulated as an Integer Linear Program (ILP); one such formulation is due to Bhootra, Sarin and Sherali [1]. Since an ILP is difficult to solve directly, one often approaches the solution for ILPs through the computation of bounds for the optimum. Such bounds can be used in Branch and Bound procedures to find the optimal solutions or to construct feasible solutions using heuristics and evaluate their quality, i.e. the percent deviation of the feasible solution from the optimum. An easy approach to find a lower bound is to relax the integrality requirement of the ILP and solve the resulting Linear Program (LP), the solution for which efficient algorithms exist. A formulation is considered "tight" if the lower bound (optimal value of the relaxed program) is "close" to the optimal value of the ILP for all instances of the problem. While it is difficult to attest to the tightness of

a particular ILP formulation, numerical results seem to indicate that the formulation of Bhootra, Sarin and Sherali is tighter than other formulations of this problem. The relative tightness of the linear relaxation was exploited to develop heuristics to solve the ILP.

Of the heuristics considered, the most promising is the split dual algorithm. The split dual allows the PCATSP to be split into two subproblems that can be solved separately at each iteration and compared. The subproblems are the problem of forming an integer solution to the regular TSP and the problem of enforcing of the precedence constraints with the formulation of Bhootra, Sarin and Sherali. This process was iterated and terminated after a certain amount of time, or after a certain solution quality was achieved. The upper bound was found to converge more quickly than the lower bound. The convergence of the lower bound was improved using a cut generation algorithm.

The ATSP with motion constraints was considered with vehicles satisfying the Dubins' vehicle model. A Dubins' vehicle is one that travels forward at a constant speed and has a minimum turning radius. This results in asymmetry in the cost to travel between a pair of nodes. A transformation that converts the ATSP with motion constraints to a One-in-a-Set TSP was developed, which in turn was solved as a standard TSP using a transformation due to Noon and Bean. The transformation discretizes the heading angles for each node, forming a set for each node. The Noon-Bean One-in-a-set transformation is then applied forming a single ATSP which can solved using a standard ATSP solver.

The HTSP was formed using several vehicles which travel at different speeds. Some vehicles incur a penalty as a function of distance from certain points in the range. Some vehicles are also forbidden to come within some distance of certain points. A transformation is presented which allows the HTSP to be formulated as a

single ATSP. For each node in the HTSP a virtual node is created for every vehicle. This forms a set for every node. The Noon-Bean transformation is then applied to these sets to form a single ATSP which is solved using a standard ATSP solver. Computational results are of good quality.

CHAPTER II

LITERATURE REVIEW

Because the TSP often arises in real-world problems, its solution and the solution to its generalizations are a subject of active research[2, 3, 4, 5, 6]. This thesis focuses on three generalizations of the ATSP: Precedence Constrained Asymmetric Travelling Salesman Problem (PCATSP), Asymmetric Travelling Salesman Problem with vehicle motion constraints and the Heterogeneous Travelling Salesman problem(HTSP).

A. PCATSP

The PCATSP is the Asymmetric Travelling Salesman Problem with the additional constraints that certain cities must be visited before other cities. Many formulations of the PCATSP are based on a formulation of the ATSP due to Miller, Tucker and Zemlin (MTZ) in [7], 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 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 problem is formulated as:

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

Subject to

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall i \in N \tag{2.1}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall j \in N \tag{2.2}$$

$$u_j - u_i \geq 1 - (n-1)(1 - x_{ij}) \quad i, j \in \{2, \ldots, n\},\ i \neq j \tag{2.3}$$

$$1 \leq u_i \leq (n-1) \quad i, j \in \{2, \ldots, n\},\ i \neq j \tag{2.4}$$

An interesting feature of this formulation are the subtour elimination constraints 2.3 and 2.4 which introduce the variables u_i which represent the position of node i in the tour. This variable gives the formulation a notion of order which other formulations lack. Using the MTZ the constraint for the precedence relation " node i must precede node j" is $u_j \geq u_i + 1$. One way to measure the quality of a formulation is to observe gap between the optimal solution of the Integer Linear Program (ILP) and the optimal solution to the problem with the itegrality 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 IP 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[8, 9, 10]. This is important because ILPs are currently solved by 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 the next step was to strengthen the MTZ subtour elimination constraints.

Desrochers and Laporte strengthened the MTZ subtour elimination constraints in [11] by applying a lifting procedure to the subtour elimination contraints. 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. Next, Sherali and Driscoll applied the Reformulation Linearization technique (RLT) to propose a formulation based on the MTZ in [12]. The RLT linearizes a non-linear program by replacing non-linear terms with new variables and corresponding contraints that increase the size of the problem. This resulted in a formulation that analytically and computationally dominated the formulation due to Desrochers and Laporte in [11], although in general taking more time to solve.

In order to improve the convergence of iterative algorithms to solve the PCATSP several valid inequalites, or "cuts" were 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 [13] which in essense separates LP solutions which 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 were proposed by Balas et al. in [14] which 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 containt node i. Using these cuts along with ATSP cuts Ascheuer et al. proposed a cutting plane approach in [15] which was implemented in a branch and cut algorithm to solve the PCATSP in [16].

Unsatisfied with the tightness of contemporary PCATSP formulations Gouveia and Pires proposed a multicommodity flow formulation (MCF) for the PCATSP in [17]. This model takes the flow point of view in order to solve the ATSP, to which

precedence constraints are easily added. This model introduced 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 model was shown analytically to have a tighter LP relaxation that the MTZ based formulations although its large size increases solution time.

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" in [1]. 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 thesis.

B. Generalized Travelling Salesman Problems

In many problems involving ordering and routing a natural is to model these problems as generalizations of the Travelling Salesman Problem. One such problem is the One in a Set TSP which is given as follows: Given a collection of node sets and one node designated as the "depot", what is the route of shortest length such that the route starts and ends at the depot and exactly one node from each node set is visited. This problem becomes more interesting in light of a transformation proposed by Noon and Bean in [6] which provides a procedure for posing the One in a set TSP as a standard ATSP. This transformation allows problems of this type to be attacked using methods developed for the ATSP. With this in mund consider the Heterogeneous Travelling Salesman Problem and the Asymmetric Travelling Salesman Problem with vehicle

motion constraints.

The HTSP is the problem of solving the Multiple Travelling Salesman Problem with vehicles that are not homogeneous. In other words, the cost to go between any two nodes is different depending on the vehicle that is traversing the path. This problem is posed as a One in a Set TSP by Noon and Bean in [6] in the case of all vehicles starting at the same point, however the case of vehicles starting at different positions is left as future work and is completed in this thesis.

The ATSP with vehicle motion constraints allows for the vehicle in the problem to have a certain range of motion, which in this case is maximum yaw-rate. In the case of constant forward speed maximum yaw-rate reduces to a minimum turning radius. Given a minimum turning radius the distance traveled by the vehicle to go between two points will rely on the departure heading angle and the arrival heading angle. This problem couples the discrete problem of deciding the sequence of points to be visited and the continuous problem of optimal heading angle. Efforts to solve this problem have focused on separating the two problems into first finding the sequence, then finding the optimal heading angles for each node [3, 2]. In this thesis the continuous problem is discretized and the ATSP with vehicle motion constraints is transformed into a One in a Set TSP.

CHAPTER III

PRECEDENCE CONSTRAINED ASYMMETRIC TRAVELLING SALESMAN PROBLEM

A. Introduction

The application of the asymmetric travelling salesman problem (ATSP) is widespread [2, 3, 4, 5, 6] and can be stated as: Given a set of nodes with one designated as a "depot", and the distance (cost) between each node, what is the route of least distance such that every node is visited once and the route starts and ends at the depot. The precedence constrained travelling salesman problem (PCATSP) has an additional set of constraints that some node i must be visited before some other node j, represented symbolically as $i < j$. A generalization of the ATSP is the case of multiple depots and multiple travelling salesman (MDMTSP) and can be restated: Given a set of nodes with some of them designated as "depots" and the distance (cost) between each node, what is the set of routes of minimum total distance such that every node is visited once, each route contains one depot. In this chapter we show that the MDMTSP can be transformed into a PCATSP. As the PCATSP is a generalization of the MDMTSP, a quick solution algorithm would not only be useful in solving standard PCATSPs but also any other problem that can be posed as an ATSP such as the Hamiltonian path problem, or the ATSP with vehicle motion constraints. In this chapter we present methods to utilize the relative tightness[8, 7] of the lifted PCATSPxy formulation proposed by Sarin, Sherali and Bhootra in [1] in an attempt focus the search for edges present in the optimal integer solution. Results of these methods are shown.

B. The Formulation: PCATSPxy

The PCATSP formulation used in this paper was proposed by Sarin, Sherali and Bhootra in [1] 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 is formulated as:

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

subject to

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall i \in N \tag{3.1}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall j \in N \tag{3.2}$$

$$y_{ij} \geq x_{ij} \quad i, j \in \{2, \ldots n\}, i \neq j \tag{3.3}$$

$$y_{ij} + y_{ji} = 1 \quad i, j \in \{2, \ldots n\}, i \neq j \tag{3.4}$$

$$y_{ij} + y_{jk} + y_{ki} + x_{ji} \leq 2 \quad i, j, k \in \{2, \ldots n\}, \quad i \neq j \neq k \tag{3.5}$$

$$y_{ij} = 1 \quad \forall (i,j) \in P \tag{3.6}$$

$$x_{ij} \in [0\ 1] \tag{3.7}$$

Constraints (3.1) and (3.2) are the assignment constraints for each depot, constraints (3.3)-(3.5) are the subtour elimination constraints and constraint (3.6) forces the precedence constraints. In [1] 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 it for use in the methods presented in this paper. A typical solution to the PCATSP is pictured below with the constraints : $4 < 5,\ 5 < 6,\ 6 < 7,\ 7 < 8$. This problem was formulated using YALMIP and solved using GLPK. Below in Figure 1 an example solution to a randomly generated instance of the PCATSP.

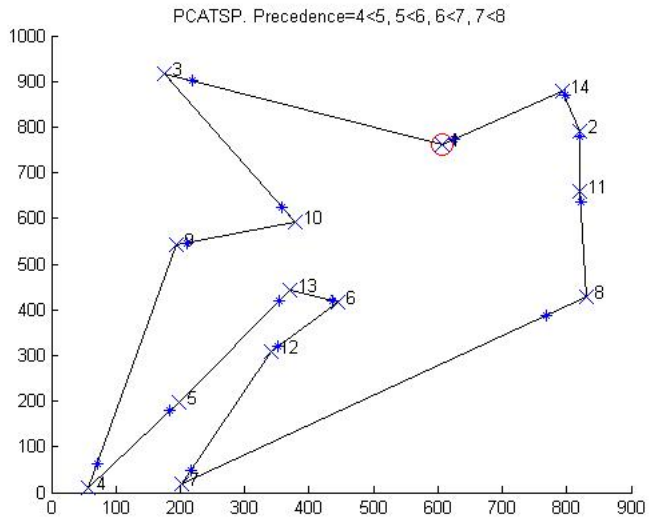


Fig. 1.: Example solution of PCATSP

C. Equivalence of MDMTSP and PCATSP

By the following transformation the MDMTSP can be posed as a PCATSP:

Let the set of nodes be N, where $|N| = n$ and the set of depots is defined as D where $|D| = m$. Let the set P be defined as the set of pairs (i, j) such that $i < j$. For every depot $d_i \in D$ create a virtual depot v_i at the same location. Let the set of virtual depots be V where $|V| = m$. Let M be a sufficiently large number. If c_{ij} is the cost to go from node i to node j, the modified cost $\tilde{c}_{ij}$ is as follows:

$$\tilde{c}_{ij} = M \quad i \in N/D, j \in D \tag{3.8}$$

$$\tilde{c}_{ij} = M \quad i \in V, j \in N \tag{3.9}$$

$$\tilde{c}_{ij} = M \quad i, j \in D, i \neq j \tag{3.10}$$

$$\tilde{c}_{ij} = M \quad i, j \in V, i \neq j \tag{3.11}$$

$$\tilde{c}_{d_i v_j} = M \quad d_i \in D, v_j \in V, i \neq j \tag{3.12}$$

$$\tilde{c}_{v_i d_j} = M \quad d_j \in D, v_i \in V, j \neq i + 1 \tag{3.13}$$

$$\tilde{c}_{d_i v_i} = 0 \quad v_i \in V, d_i \in D \tag{3.14}$$

$$\tilde{c}_{v_i d_{i+1}} = 0 \quad i \in \{1, \ldots, m-1\}, v_i \in V, d_i \in D \tag{3.15}$$

$$\tilde{c}_{v_m d_1} = 0 \tag{3.16}$$

where M is sufficiently large. In addition, the following precedence constraints are added:

$$(d_i, v_i) \rightarrow P, \quad i \in \{1, \ldots, m\} \tag{3.17}$$

$$(v_i, d_{i+1}) \rightarrow P, \quad i \in \{1, \ldots, m-1\} \tag{3.18}$$

The purpose of equation (3.8) is to effectively remove edges that start in the set

of non-depot nodes and end in the set of depots by making their cost large enough to prohibit their membership the optimal solution. Likewise equation (3.9) effectively removes all edges starting in the set of virtual depots and ending in the set of non-depot nodes. Equations (3.10)-(3.16) use the same method to effectively create zero cost edges that connect each depot to its corresponding virtual depot, and each virtual depot to one adjacent depot. The precedence constraints (3.17) and (3.18) ensure that the portion of the optimal tour between each depot and its corresponding virtual depot will not contain any other depot or virtual depot.

D. Equivalence of PCATSP and Hamiltonian Path

A well known transformation can be applied to the Hamiltonian path problem in order to pose it as an ATSP. The Hamiltonian path problem can be stated as: Given a set of nodes with one designated as the "depot" and distance (cost) between each, what is the minimum cost path starting at the depot such that every node is visited once. The transformation is performed as follows: First add a virtual node v to the set of nodes with cost:

$$c_{vd} = 0 \tag{3.19}$$

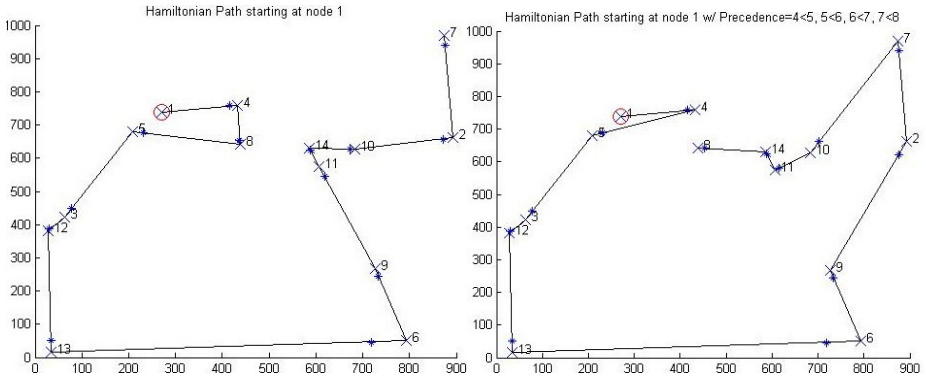
$$c_{dv} = 0 \tag{3.20}$$

$$c_{iv} = 0 \quad i \in N, i \neq d \tag{3.21}$$

$$c_{vi} = M \quad i \in N, i \neq d \tag{3.22}$$

where M is sufficiently large, d is the depot and N is the set of nodes. Solution of the ATSP over this modified cost will return the Hamiltonian path. The precedence constrained Hamiltonian path problem adds the additional set of constraints that

some node i must be visited before some other node j, represented symbolically as i¡ j. This problem can be transformed into the PCATSP with the transformation described above. The solution to this problem will give the Hamiltonian path such that the precedence constraints are satisfied. An example solution of the Hamiltonian path problem with and without precedence constraints is pictured below. Both of these problems were formulated using YALMIP and solved using GLPK. Below in Figure 2 example solutions to a randomly generated instances of the Hamiltonian Path problem with and without precedence constraints are shown.



(a) Without precedence constraints (b) With precedence constraints

Fig. 2.: Example solutions of Hamiltonian path problem

CHAPTER IV

SUB-OPTIMAL SOLUTION TECHNIQUES FOR THE PCATSP

A. Introduction

While the determination of an optimal solution is the goal of any optimization problem, techiniques to solve these problems to optimality are often time consuming and experience "tailing off" in which after a certain time, very little improvement in the objective function is realized after each iteration of the solver. For this reason it is often advantageous to use an algorithm that generates a feasible solution very quickly with a guarantee on the quality of the solution. Several such 'sub-optimal' solution techinques are presented.

B. 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, and evaluate the solution to the new problem for feasibility for the original problem. Solutions to the new problem that are infeasible for the original problem are then removed from the set of feasible solutions by adding a constraint. This process is then 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 are removed leaving a standard ATSP problem. The standard ATSP is then solved and the solution is evaluted for feasibility. If the ATSP solution is PCATSP infeasible then two types of cut are generated. These cuts are know as the W-cut and Predecessor-Succesor cuts and are described in the following subsections.

1. W-cut

The W-cut is proposed in [13] 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 below.

Consider as above, a precedence relation that node i must precede node j. Now consider a solution to the ATSP containing the subset of edges shown in Figure 3.

Fig. 3.: Edge subset in an infeasible solution

Next form the set W with the nodes that lie in the path from j to i as indicated in the Figure 4 below.

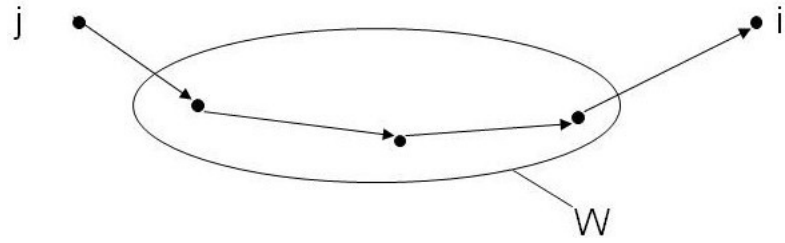


Fig. 4.: "W" set in an infeasible solution

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 may be visualized in Figure 5.

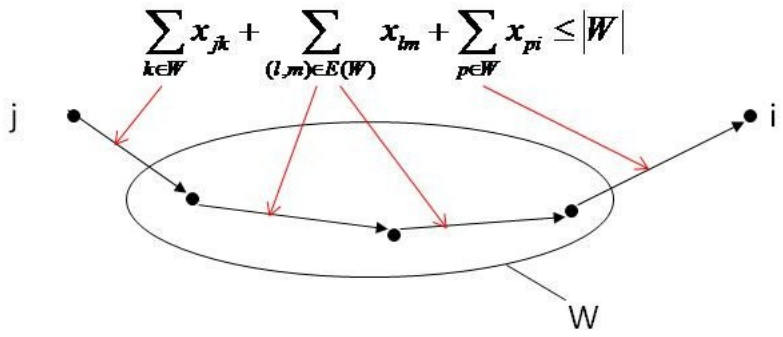


Fig. 5.: Edges dictating the "W" cut

The addition of this inequality to the ATSP will result in the separation of the previous solution.

2. 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 capactiy to carry one unit of flow from i to j. In this context given we can contruct the Predecessor-Successor Cuts. First we can define a set π_i, which is the set of nodes which 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 $(i, j) \in P$. By assumption the set P is consistent (satisfies the transitivity relation), meaning if $(i, j) \in P$ and $(j, k) \in P$ then $(i, k) \in P$. 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 one 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 one 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 one 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 one. This algorithm is illustrated below.

First we have a feasible solution to the relaxation to the ATSP (shown in Figure 6 which will be evaluated for feasibility given the constraint that node i must precede node j.

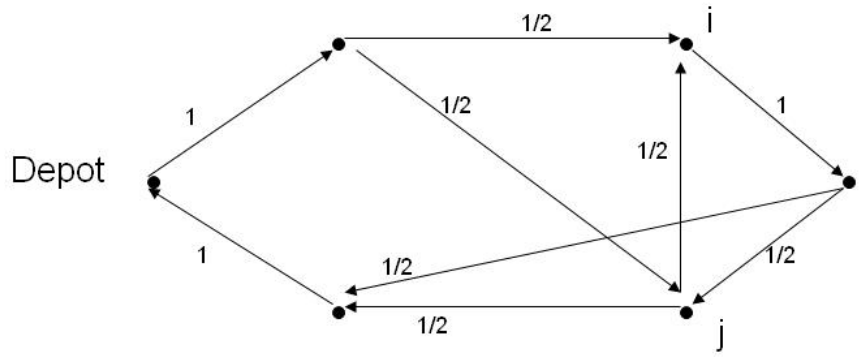


Fig. 6.: LP feasible solution

Next remove the successors of node i, or in this case, just node j (as shown in Figure 7).

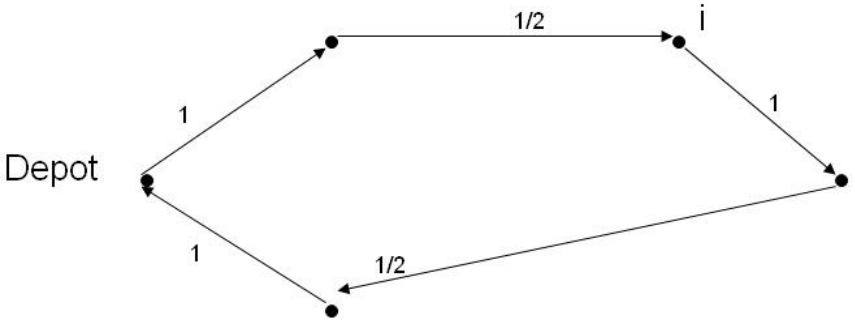


Fig. 7.: Remove members of σ_i

Now from the remaining graph, it can be seen that one unit of flow cannot be passed from the depot to node i along the path that is highlighted in Figure 8.

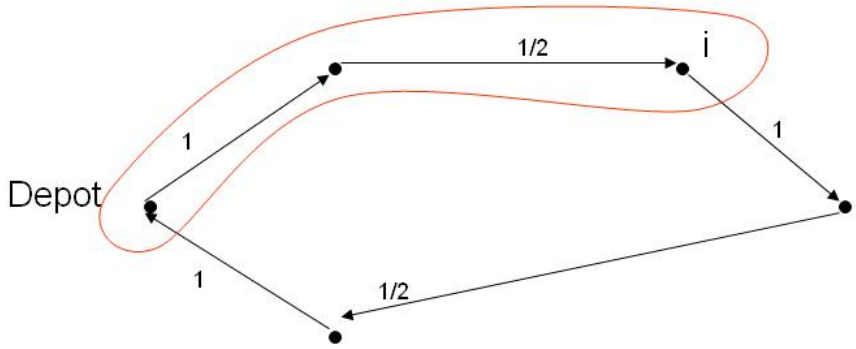


Fig. 8.: Path along which 1 unit of flow cannot pass

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 one. These two sets are labeled below in figure 9.

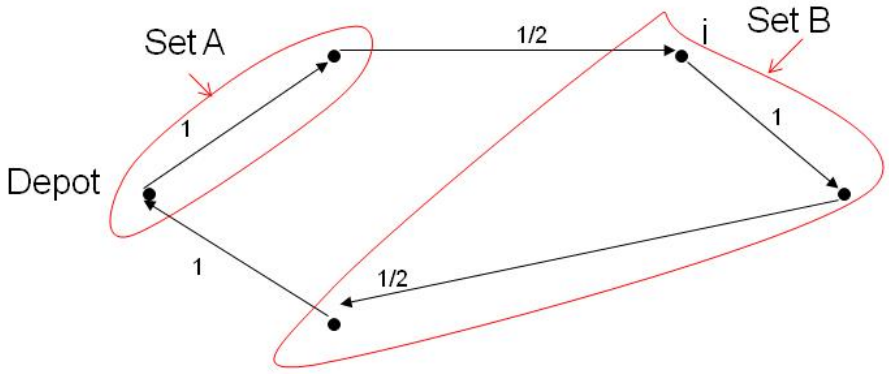


Fig. 9.: Set A and Set B

In this example then a cut is generated and added to the ATSP relaxation requiring the sum of the flow capacitites over the edges that start in set A and end in set B must be at least 1. Also note that sets A and B are not necessarily unique.

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 one. This algorithm is similar to the procedure presented graphically in figures 6 - 9.

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 one. This algorithm is similar to the procedure presented graphically in figures 6 - 9.

C. Split Dual

Consider the following formulation for the PCATSP:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{4.1}$$

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 4.1. Clearly one may express 4.1 as:

$$\min \sum_{i,j} c_{ij}x_{ij} \tag{4.2}$$

Subject to:

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

$$x = z \tag{4.3}$$

Let μ_{ij} be the penalty for the violation of constraint $x_{ij} = z_{ij}$. Then constraint (4.3) can be relaxed, giving the 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 re-written as:

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

Since the first term is the ATSP over $c - \mu$, 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 I below shows the results of the split-dual algorithm implemented in C++. The ATSP was solved using the LKH algorithm written by Keld Helsgaun [5]. The LP relaxations and integer PCATSPxy were solved using GLPK and the solution to this problem is under the "IP OPT" column and the time in which this solution was computer is under the "IP time" column. At each iteration the upper bound is generated by applying a simple heuristic to the optimal tour in order to satisfy the precedence constraints. The algorithm terminated after 200 iterations, or if the duality gap fell under 1%. Cases in which the iteration limit was exceeded when solving the integer PCATSPxy are denoted with a "DNS" or did-not-solve. The time listed for these cases is the time elapsed until the iteration limit was exceeded. The first column "n" is the number of targets considered, $|P|$ is number of precedence constraints, "SP LB", "SP UB" and "D Gap" denote the split dual lower bound, upper bound and duality gap respectively. While the duality gap on average is greater than 10%, the gap between the primal feasible upper bound, and the IP optimal solution is less than 5% on average.

Table I.: Split Dual Results

n	$\|P\|$	SP LB	SP UB	D Gap	IP OPT	SD Time	IP Time	(SD-IP)/IP
15	4	3448.98	3480	0.01	3480	13.48	8.94	0.00
15	4	3641.64	4448	0.22	DNS	80.62	251.00	DNS
15	4	4457.65	4531	0.02	4526	46.18	114.80	0.00
15	4	4042.67	4477	0.11	4280	59.35	6.65	0.05
15	4	3993.73	4929	0.23	DNS	70.33	250.00	DNS
15	4	4207.27	4255	0.01	4255	51.91	19.36	0.00
15	4	4500.93	5343	0.19	5067	68.38	124.00	0.05
15	4	4554.10	5233	0.15	5160	42.28	103.36	0.01
15	4	4011.53	4220	0.05	4046	76.35	8.00	0.04
15	4	4062.03	4500	0.11	4085	97.09	8.68	0.10
			AVG	0.11		60.60	89.48	0.04
16	4	3405.21	4186	0.23	DNS	121.26	353.00	DNS
16	4	4448.75	4659	0.05	4601	73.90	86.00	0.01
16	4	4308.65	5167	0.20	4765	113.68	18.44	0.08
16	4	4314.75	4864	0.13	4696	105.47	67.08	0.04
16	4	4222.07	5286	0.25	DNS	133.10	533.00	DNS
			AVG	0.17		109.48	211.50	0.04

As can be seen in Table I the lower bound formed by the LP relaxation converged slowly relative to the upper bound, which causes 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 II below. The algorithm was stopped after 25 iterations or if the duality gap fell below 5%. The first column "n" is the number of targets considered and $|P|$ is number of precedence constraints. The next three 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, "UB gap" is the difference between the split dual upper bound and the IP optimal solution divided by the IP solution, and "Time" is the time in seconds to run the split dual algorithm with no cuts applied to the lower bound. The final three 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, "Cut UB gap" is the difference between the split dual upper bound and the IP optimal solution divided by the IP solution, 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 (healthy) 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 5 randomly generated instances of the problem with the labeled number of targets and precedence constraints.

Table II.: Split Dual with Cut Generation

n	$\|P\|$	D-gap	UB gap	Time	Cut D-gap	Cut UB gap	Cut Time
10	2	0.056	0.016	2.7	0.039	0.009	2.7
10	3	0.111	0.028	3.5	0.050	0.012	4.6
10	4	0.128	0.021	4.2	0.043	0.006	5.4
10	5	0.075	0.003	3.7	0.040	0.009	5.2
15	2	0.230	0.056	36.6	0.098	0.019	54.4
15	3	0.114	0.020	34.6	0.089	0.021	51.9
15	4	0.137	0.025	30.9	0.096	0.020	44.5
15	5	0.167	0.026	31.9	0.106	0.027	45.9
20	2	0.373	N/A	291.2	0.228	N/A	350.1
20	3	0.200	N/A	300.6	0.138	N/A	362.7
20	4	0.287	N/A	272.8	0.251	N/A	338.1

From the Table II it can be seen that adding cut generation is successful in closing the duality gap, albeit with a penalty in solution time. It is also interesting to note that the upper bound generated by the split dual algorithm with or without cuts were high quality, however in situations where the IP optimum is unavailable, adding cut generation can drastically reduce the duality gap.

D. α-nearness

As the relaxation of the formulation used is relatively tight[1] our strategy is to leverage the LP relaxation to obtain an estimate of how likely an edge is to be in the optimal solution. This approach is borrowed from the notion of "α-nearness" used by Keld Helsguan in [5] in which edges are ranked in order of the cost of the minimum one-tree to which they belong. Helsguan reported that 75%-85% of the edges in the optimal tour are present in the optimal one-tree[5]. Implementing this into the Lin-Kernighan heuristic proved very successful and found tours whose cost set the record for several unsolved instances in the TSPLIB. This algorithm is successful because one-trees capture the underlying structure of the problem, one-trees are easy to compute, and the edge ranking not only reduces the number of edges considered, but first checks edges which are most likely to be in the solution. Similar to the 1-tree α-rank in [5], we "alpha rank" the edge x_{ij} by the optimal objective function value of the PCATSPxy LP relaxation with the additional constraint that the edge x_{ij} is present in the solution. In order to force edge x_{ij} to be present in the solution we add in a virtual node v and modify the edge costs. Let M be sufficiently large. Then we set:

$$c_{iv} = 0 \tag{4.4}$$

$$c_{vj} = 0 \tag{4.5}$$

$$c_{kv} = M \quad \forall k \in N/\{i\} \tag{4.6}$$

$$c_{vk} = M \quad \forall k \in N/\{j\} \tag{4.7}$$

After the cost is modified, the relaxation is solved and c_{ij} is added to the objective function. Equations (4.4)-(4.7) ensure that the edges x_{iv} and x_{vj} will be present in the optimal solution, making the remaining problem equivalent to the relaxed

Hamiltonian path problem from j to i. Since these two edges have zero cost, the cost of the edge from i to j must be added to the cost of the Hamiltonian path to reflect the cost of the corresponding tour. This method is preferred to constraining the edge to be in the solution because adding the constraint may render the previous solution infeasible, forcing the LP solver to calculate a new feasible starting point, which is computationally costly. Table III shows the results of the procedure described previously. The column 'n' is the number of nodes in the problem, and $|P|$ is the number of precedence constraints. For comparison with the alpha-rank, the nearest neighbor rank for each edge was calculated, along with the optimal one-tree found after adding the dual variable associated with the degree constraint for each edge to its cost. While the alpha-rank consistently outperformed the nearest neighbor rank, the alpha rank failed to provide a low rank for edges present in problems which the IP took a long time to solve, where it is most needed. All linear and integer programs were formulated in YALMIP and solved in GLPK. All times are in seconds.

Table III.: α-nearness

n	$\|P\|$	avg rank	max rank	avg nn-rank	max nn-rank	avg d-one-tree	max d-1tree	alpha-time	IP TIME
14	4	2.79	12	3.43	10	2.93	7	52.01	2.60
14	4	2.00	5	3.21	8	3.21	9	59.68	3.00
14	4	1.93	6	3.57	12	2.57	6	53.77	3.84
14	4	3.14	10	4.14	12	3.14	7	58.59	40.22
14	4	1.00	1	3.21	10	1.43	5	54.52	1.71
14	4	2.50	12	3.57	13	3.57	13	55.51	143.66
14	4	2.28	4	3.78	13	2.64	6	55.51	22.12
14	4	2.78	10	3.64	9	1.86	7	61.21	13.63
14	4	1.92	7	4.07	13	2.21	8	58.47	7.68
	AVG	2.30	7.3	3.669	11.1	2.656	7.5	56.80	24.97
16	4	3.00	7	2.68	7	2.06	5	164.60	4.91
16	4	2.94	11	4.06	13	2.88	11	182.25	257.71
16	4	2.69	5	3.63	10	2.06	8	162.64	8.99
16	4	3.13	11	3.88	11	2.69	10	157.93	32.97
16	4	1.88	5	2.94	8	2.13	6	152.33	8.25
16	4	1.25	2	2.94	15	2.38	8	180.93	3.84
16	4	1.88	8	2.88	9	2.13	8	165.15	5.44
16	4	1.31	2	3.13	9	2.38	11	170.74	3.58
16	4	2.81	8	3.69	9	2.44	11	185.97	194.39
	AVG	2.30	6.8	3.339	10.6	2.34	8.6	170.24	52.99

CHAPTER V

GENERALIZED TSPS POSED AS THE ONE IN A SET TSP

A. One in a Set TSP

The One in a Set TSP can be described as follows: Given a collection of node sets and the cost between nodes of different sets, what is the least cost path such that only one element of each set is visited and the path starts and ends at the same node. This problem can be restated in simpler terms. Consider a person at his house that must go and return, by car, to a hardware store, a grocery store and a dry-cleaner. The person knows of several hardware stores, grocery stores and dry-cleaners in the area and wishes to expend the least amount of fuel running his errands. In this case he must visit only one hardware store out of the set of hardware stores, one grocery store out of the set of grocery stores, etc... This problem is widely applicable and is useful in solving other classes of traveling salesman problems that can be posed as One in a Set TSPs as will be shown in this chapter. A general solution to a One in a Set TSP problem is shown below in Figure 10.

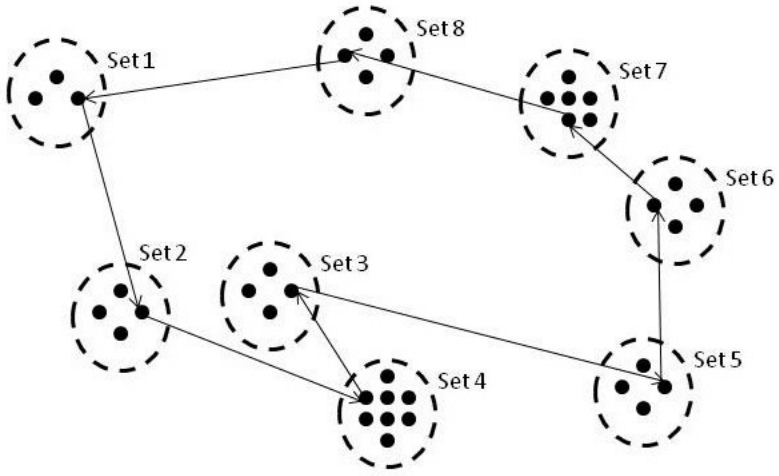


Fig. 10.: Example solution of a general One in a Set TSP

B. Noon-Bean Transformation

The method presented in [6] by Noon and Bean is a transformation that converts a one in a set TSP into a standard TSP. This is useful because it allows the one in a set TSP to be attacked using the numerous tools and methods developed to solve the standard TSP. The transformation first modifies the cost such that nodes belonging to the same set will be visited in the TSP solution in a specific, consecutive order according to which node is present in the solution to the one in a set TSP. Knowing this order allows the solution to the TSP to be transformed back into the solution to the one in a set TSP. This transformation is explained in detail below:

Start with a one in a set TSP with the set of node sets denoted as '$\mathbb{W}$' with cardinality $|\mathbb{W}| = W$. Let the node sets which are the elements of $\mathbb{W}$ be denoted as 'S_i' with cardinality $|S_i| = s_i$. Let the jth node in the set S_i be denoted as '$n_{i,j}$'. Let the cost to travel from node k in set S_i to node l in set S_j be denoted as $c_{n_{i,k}n_{j,l}}$. The cost is then modified:

First intraset edges are added:

$$c_{n_{i,k}n_{i,k+1}} = 0, \quad k = 1, \ldots, s_i - 1$$

$$c_{n_{i,s_i}n_{i,1}} = 0$$

This creates a cycle of zero cost edges between adjacent nodes in each set. This is done to ensure that if the tour enters the set S_i at node $n_{i,k}$, it will visit each node in the set in order , and leave S_i at node $n_{i,k-1}$.

Next the outgoing interset edges are 'shifted':

$$c_{n_{i,k-1}n_{j,l}} = c_{n_{i,k}n_{j,l}} \quad \forall n_{i,k} \in S_i, \forall n_{j,l} \in S_j, \ i \neq$$

This shifts the starting node of all outgoing interset edges 'back' one. If the tour enters set S_i at node $n_{i,k}$ the tour will leave at node $n_{i,k-1}$. This shift is such that the cost of the edge entering the set, plus the cost of the zero cost edges connecting the nodes in the set, plus the cost of the edge leaving the set is the same as the cost of a tour entering and leaving the set at the same point in the original one in a set TSP.

Finally add a sufficiently large M to all interset edges. If M is at least an upper bound for the one-in-a-set TSP it is considered sufficiently large. Clearly, M can be computed easily if the graph is complete on the set of nodes, as is the case we deal with here.

This ensures that only W interset edges will be chosen in the optimal tour as a tour with more than W interset edges will have unused zero cost intraset edges which can be used to create a tour of less cost.

After the TSP is solved over the transformed graph, the solution is transformed back by deleting the zero cost edges in the tour and shifting the outgoing intraset edge starting nodes 'forward' as follows:

First define $e*_{n_{i,k}n_{j,l}} = \begin{cases} 1 & \text{if edge from } n_{i,k} \text{ to } n_{j,l} \text{ is present in the TSP solution} \\ 0 & \text{otherwise} \end{cases}$

Then the set of edges '$e_{n_{i,k}n_{j,l}}$' are the solution to the one in a set TSP and are defined as:

$$e_{n_{i,k}n_{j,l}} = e *_{n_{i,k-1}n_{j,l}} \quad \forall n_{i,k} \in S_i,\ n_{j,l} \in S_j\ \forall S_i, S_j \in \mathbb{W},\ S_i \neq S_j$$

$$e_{n_{i,1}n_{j,l}} = e_{n_{i,s_i}n_{j,l}}$$

.

This procedure can be seen visually in the following illustrations:

First start with the One in a Set TSP shown in Figure 11.

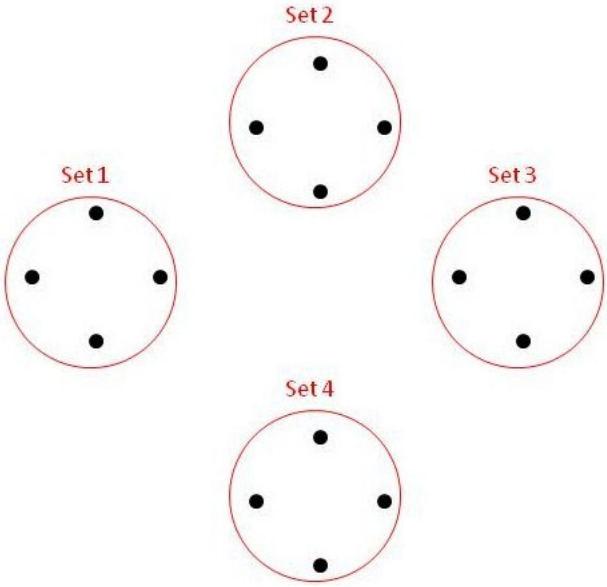


Fig. 11.: General One in a Set TSP

Next connect adjacent nodes in each set by zero cost edges as shown in Figure 12.

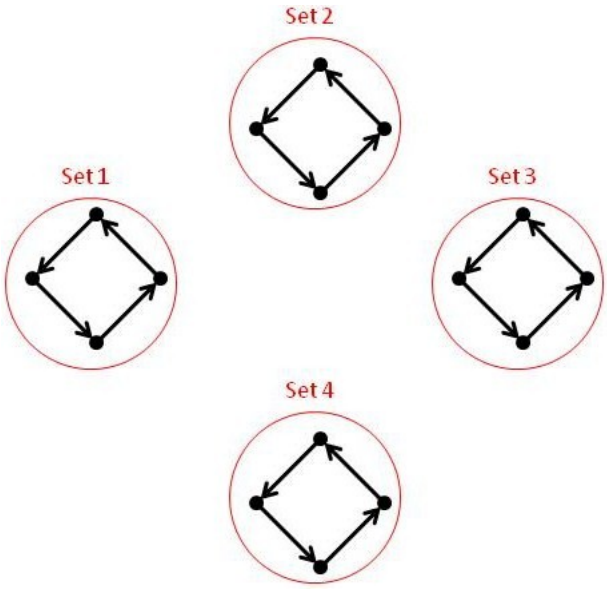


Fig. 12.: First step in transformation: Zero cost intraset edges

Next shift the starting node of each outgoing interset edge 'back' by one. This procedure is applied to every outgoing interset edge, but to avoid clutter only four edges are displayed in figure 13. This step also includes adding a sufficiently large M to each interset edge cost.

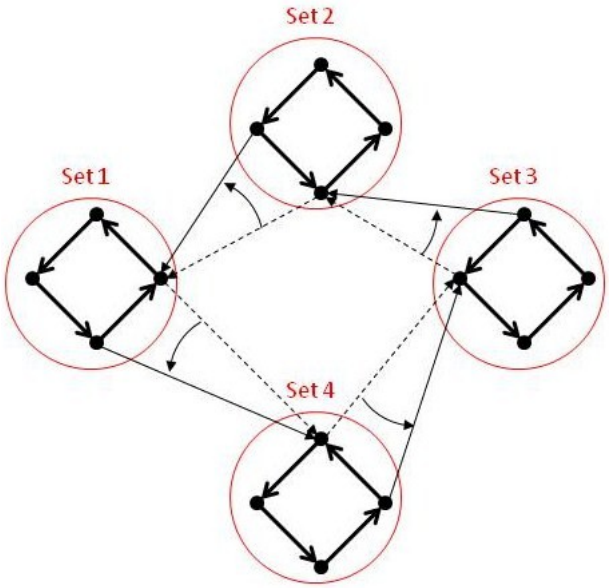


Fig. 13.: Second step in transformation: Shift edge starting node

Next the TSP over the modified graph is solved as in Figure 14.

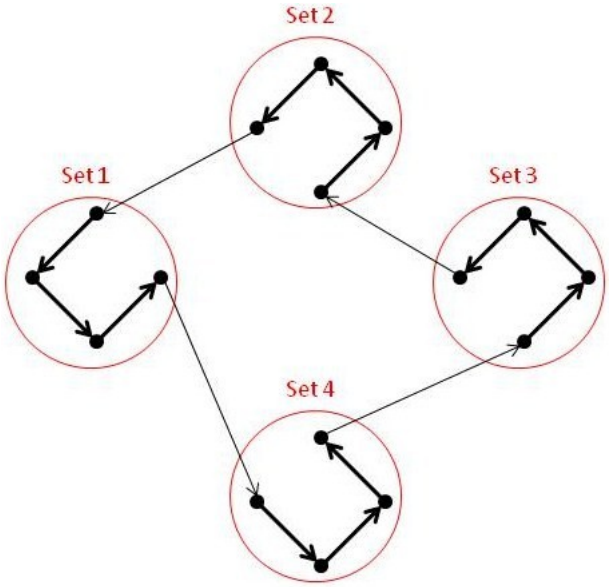


Fig. 14.: Solution to transformed TSP

The final step is to delete the zero cost intraset edges and shift the outgoing

interset edge starting points 'forward' by one. This gives the solution to the original one in a set TSP and is shown in Figure 15.

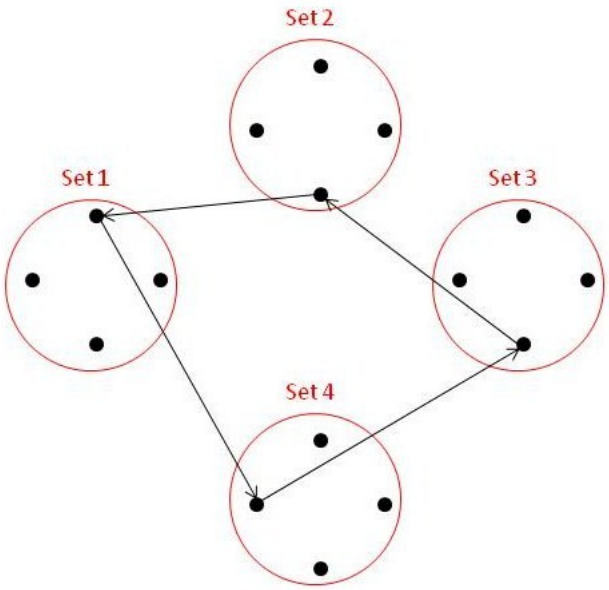


Fig. 15.: Solution to One in a Set TSP

C. ATSP with Motion Constraints

The motion constrained ATSP can be stated as follows: Given a vehicle with motion constraints and set of nodes with one designated as a "depot" and the position of each node, what is the minimum distance path for the vehicle to follow such that every node is visited once, and the path that starts and ends at the depot. As an example, Dubins' vehicle constraints will be used. A Dubins' vehicle moves forward at a constant speed and has a minimum turning radius. Given the position of two points and the heading angle at which the vehicle must visit the points, one can calculate the path of least length satisfying the motion constraint; we refer to length of such a path as Dubins' distance between the given points for the specified initial and final headings. Therefore the solution to this problem requires not only the optimum order

in which to visit the nodes, but also the optimum heading angle for the vehicle to visit each node. This coupling makes the problem very difficult. If the heading angle that the vehicle can visit each node is discretized, this problem can be posed as a One in a Set TSP, therefore transformed into a regular ATSP using the Noon-Bean transformation [6]. The one in a set TSP is solved over the set of angle discretizations for each node. In other words, for each node, a set of angle discretizations is generated. The problem is then to find the tour of minimum cost that visits one element of each these sets. This transformation is described in detail below.

The motion constraints are transformed as follows:

Given a number of heading angle discretizations m to be taken out of 360^o, each node is split into m virtual nodes, each representing one of the heading angles generated by the discretization. In other words, each node in the metric TSP becomes a set of m nodes that represent equally spaced heading angles. The problem is then to choose one node in each set to complete the tour. This transformation is shown in Figure 1 using a 4 node ATSP with 4 angle discretizations.

Then each set m representing the angle discretizations for each node is transformed into a standard ATSP using the transformation in [6]. This is done by connecting the new nodes representing the angle discretizations with zero cost edges as follows:

Let the heading angles be calculated by $a_i = \frac{i}{m}360^0$ where $i = 1, 2, \ldots, m$. For each node k in the TSP let k_i be the node that represents the heading angle a_i for node k. Let c_{kn} be the cost to travel from node k to node n. Then the intra set costs are:

$$c_{k_i k_{i+1}} = 0, \quad i = 1, 2, \ldots, m-1 \tag{5.1}$$

$$c_{k_m k_1} = 0 \tag{5.2}$$

Taking the nodes representing heading angles for node 3 in the metric ATSP shown above, the intra-set costs can be visualized by Figure 2:

The Dubins' distance is then calculated between different nodes in the TSP by calculating the Dubins' distance between each node using every combination of heading angle discretization for each pair of nodes as follows: Let $d_{k_i n_j}$ be the Dubins' distance between node k with heading angle a_i and node n with heading angle a_j. Then the inter-set costs are:

$$c_{k_i n_j} = d_{k_{i-1} n_j} + M, \quad i = 2, \ldots, m, \ j = 1, 2, \ldots, m \tag{5.3}$$

$$c_{k_1 n_j} = d_{k_m n_j} + M, \quad j = 1, 2, \ldots, m \tag{5.4}$$

The heading angle of the starting node is shifted back by one to force the arrival angle to be the same as the departure angle as the tour will traverse the nodes in each set in counter-clockwise order because of the intra-set edges whose cost are zero (as described in the previous section). A sufficiently large M is added to the inter-set edges to ensure that at most one inter-set edge per node in the original problem is chosen to be in the tour. If there are n nodes and m angle discretizations, after applying (5.1) - (5.4) the ATSP is solved over the $n \times m$ heading angle nodes. This process can be visualized in the figures below.

First each node is split into m virtual nodes, one for each heading angle discretization. In this case we will use 4 heading angle discretizations. This is pictured in Figure 16.

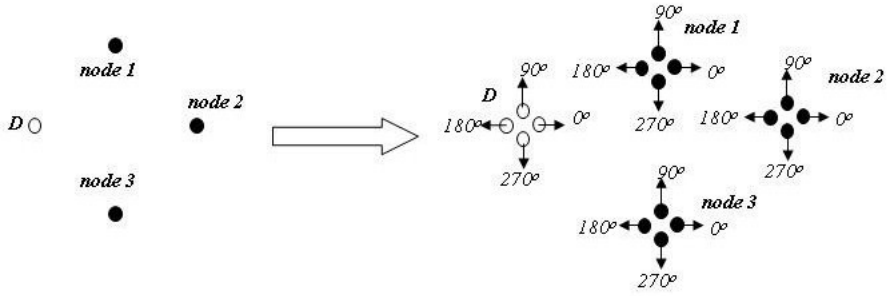


Fig. 16.: Convert Dubins' TSP into One in a Set TSP

The discretizations for each individual node form the sets over which we will solve the One in a Set TSP. The next step is to add zero cost edges between adjacent intraset nodes as in Figure 17.

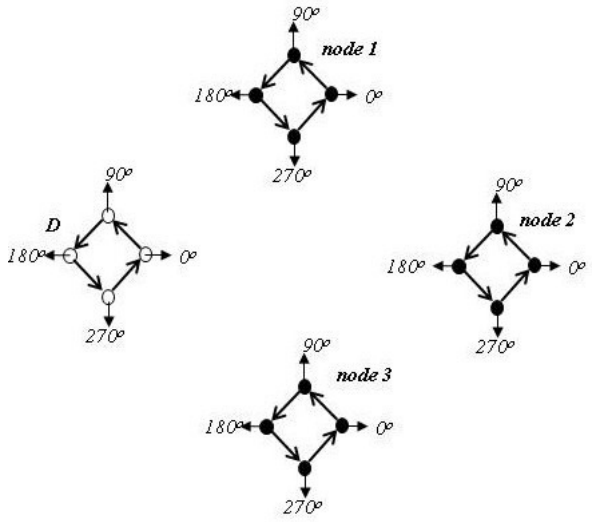


Fig. 17.: Add zero cost edges

The outgoing interset edges are then shifted as described in the previous section and M is added the their cost. The ATSP is then solved over the modified graph and transformed back into a solution to the motion constrained TSP as is shown in Figure 18.

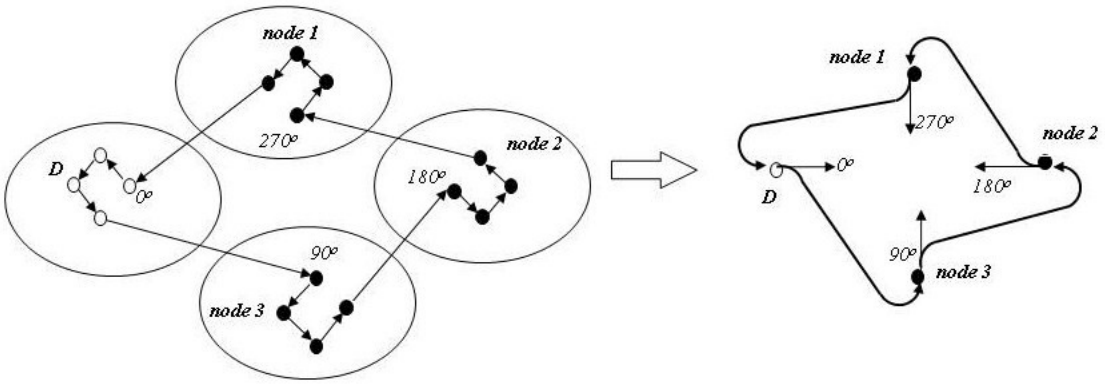


Fig. 18.: Convert One in a Set TSP solution into Dubins' TSP solution

The result of this tour is the path [D, node 3, node 2, node 1] with heading angles of 0^o, 90^o, 180^o and 270^o respectively.

Using available TSP solvers allow moderately large problems to be solved. Below in Figure 19 is a problem with 50 nodes and 16 angle discretizations. The TSP solver used is the LKH solver written by Keld Helsgaun [5].

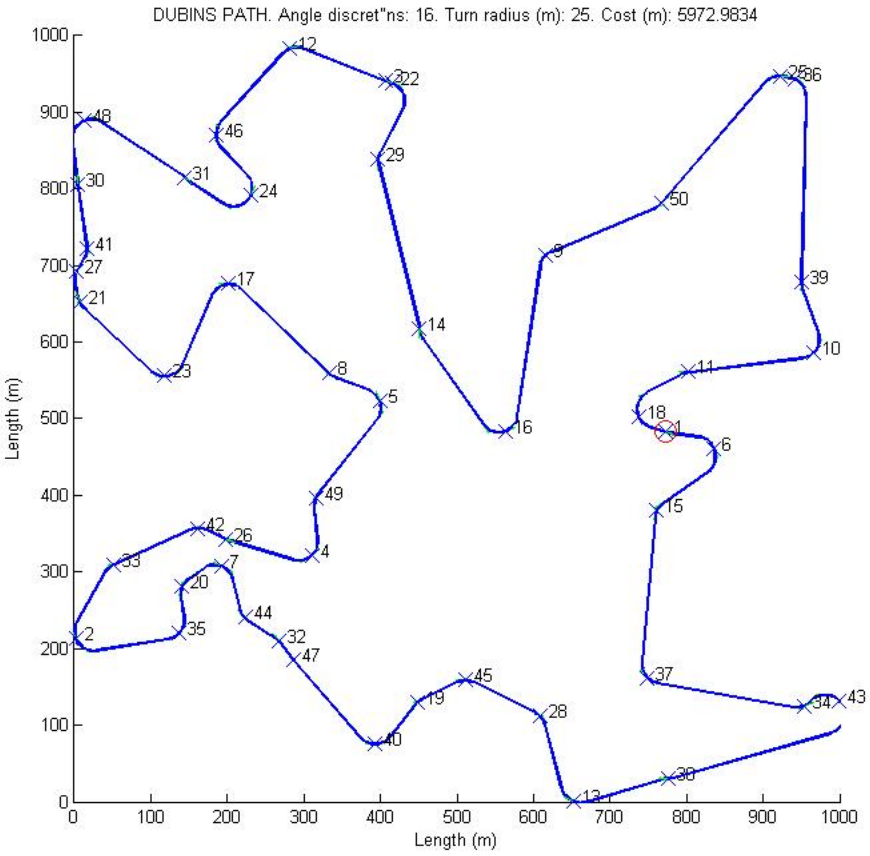


Fig. 19.: Example solution of motion constrained TSP

D. Heterogeneous TSP

The heterogeneous TSP (HTSP) can be described as follows:

Given a set of nodes $\mathbb{N}$, a set of starting positions $\mathbb{V}$ for a set of heterogeneous vehicles and the cost for each vehicle to travel between nodes, what is the path for each vehicle such that every node is visited once, each vehicle returns to its starting point, and the total cost of all vehicle paths is minimum. This problem can be restated in a practical example. A geologist wants to study the geology of a desert and knows the

specific locations in the desert where he would like to take samples. Some of these locations are accessible only by paved road, some are accessible using unpaved offroad trails and some are accessible by both. Given that the geologist has a truck able to travel on both paved roads and offroad trails, and a vehicle that can only travel on paved roads, what is the sequence of locations he should visit in each vehicle such that the least amount of fuel is used. This problem can be transformed into a One in a Set TSP using the Noon-Bean transformation and is described in [6]. The process is described below:

Let the cardinality of $\mathbb{N}$ be $|\mathbb{N}| = N$, the cardinality of $\mathbb{V}$ be $|\mathbb{V}| = V$. Let the cost of vehicle i to travel between node j and node k be denoted as c_{ijk}. Let the cost to travel between node i and node j in the transformed graph be denoted as $c*_{ij}$

1. Replace each node $n \in \mathbb{N}$ with V virtual nodes, so there are $N \times V$ nodes in the new graph, effectively creating single TSP problem for each vehicle. Let the virtual node for vehicle i that represents node j in the HTSP be denoted as $n_{i,j}$. The set of virtual nodes corresponding to each node in the HTSP form a set which will be denoted as S_j, or $n_{1j} \cup n_{2j} \cup \cdots \cup n_{Vj} = S_j$

2. For each vehicle $v \in \mathbb{V}$ create a virtual node d. Let the union of all these virtual nodes form the set D. Connect the vehicle starting positions and and the virtual nodes V as follows:

$$c*_{v_i d_i} = M, \quad \forall v_i \in \mathbb{V}$$

$$c*_{d_i v_{i+1}} = 0, \quad i = 1, \ldots, V-1$$

$$c*_{d_V v_1} = 0$$

This creates a set of edges of cost M between each vehicle starting position and its corresponding virtual node and a zero cost edges from each virtual node to

its adjacent vehicle starting position.

The costs to travel to and from the vehicle starting positions are defined as follows:

$$c*_{v_i n_{i+1,j}} = c_{iv_ij} + M, \quad \forall v_i \in \mathbb{V},\ j \in 1, \ldots, N$$

$$c*_{v_V n_{1,j}} = c_{v_V j} + M$$

$$c*_{n_{i,j} d_i} = c_{ijv_i} + M, \quad i \in 1, \ldots, V\ j \in 1, \ldots, N$$

3. Perform the Noon-Bean transformation on the sets S_i:

$$c*_{n_{i,j} n_{i+1,j}} = 0, \quad i = 1, \ldots, V-1,\ \forall j \in \mathbb{N}$$

$$c*_{n_{V,j} n_{1,j}} = 0, \quad \forall j \in \mathbb{N}$$

$$c*_{n_{i,j} n_{i+1,k}} = c_{ijk} + M, \quad \forall j,k \in \mathbb{N},\ i = 1, \ldots, V-1$$

$$c*_{n_{V,j} n_{1,k}} = c_{Vjk} + M, \quad \forall j,k \in \mathbb{N}$$

After the TSP is solved over the transformed graph, the solution is transformed back by deleting the edges connecting the vehicle starting positions and the corresponding virtual nodes in the transformed tour and 'shifting' the intraset edges as follows:

First define $e*_{n_{i,k} n_{j,l}} = \begin{cases} 1 & \text{if edge from } n_{i,k} \text{ to } n_{j,l} \text{ is present in the transformed tour} \\ 0 & \text{otherwise} \end{cases}$

Then the new set of edges 'e' such that:

$$e_{n_{i,k} n_{i,l}} = e*_{n_{i,k} n_{i+1,l}}, \quad \forall k,l \in \mathbb{N},\ i = 1, \ldots, V-1$$

$$e_{n_{V,k} n_{V,l}} = e_{n_{V,j} n_{1,j}}, \quad \forall k,l \in \mathbb{N}$$

$$e_{v_i n_{i,j}} = e*_{v_i n_{i+1,j}}, \quad \forall j \in \mathbb{N},\ i = 1, \ldots, V-1$$

$$e_{v_V n_{V,j}} = e*_{v_V n_{1,j}}, \quad \forall j \in \mathbb{N}$$

$$e_{n_{i,j} v_i} = e*_{n_{i,j} d_i}, \quad \forall j \in \mathbb{N},\ \forall i \in \mathbb{V}$$

is the solution to the HTSP.

This procedure can be seen visually in the following illustrations.

First each node in the HTSP is split into V virtual nodes, or one for each vehicle. This can be thought of as creating a separate plane for each vehicle. This can be visualized in Figure 20.

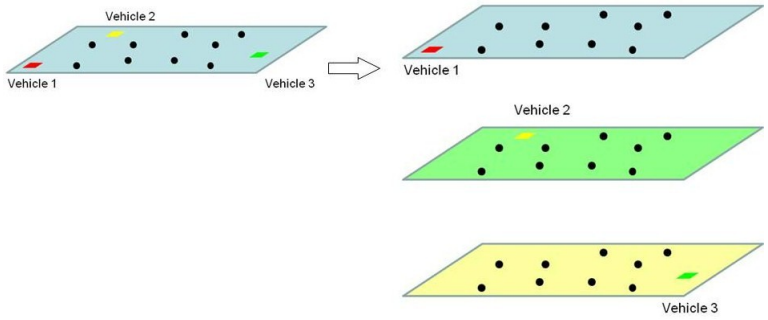


Fig. 20.: Convert HTSP to one in a set TSP

The virtual nodes that represent a node in the HTSP form the sets over which the One in a Set TSP is solved. Next zero cost edges are added between adjacent intraset nodes. This can be visualized as connecting each node with a zero cost edge to the node in the plane 'below' it. This is shown in Figure 21. In order to avoid clutter, edges that connect nodes in the 'plane' of vehicle three to nodes in the 'plane' of vehicle one are shown to disappear from the bottom of the figure and reappear at the top of the figure where they meet the proper nodes in the plane of vehicle 1.

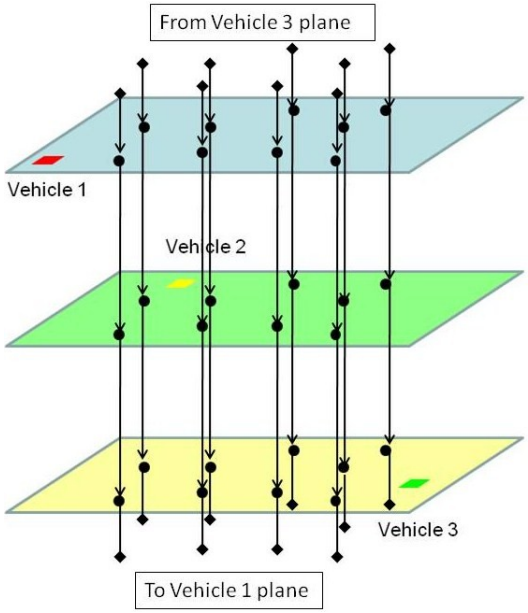


Fig. 21.: Connect intraset nodes with zero cost edges

Next, vehicle starting positions are connected with their respective virtual nodes by an edge of cost M, and virtual nodes are connected with adjacent vehicle starting positions with zero cost edges as shown in Figure 22.

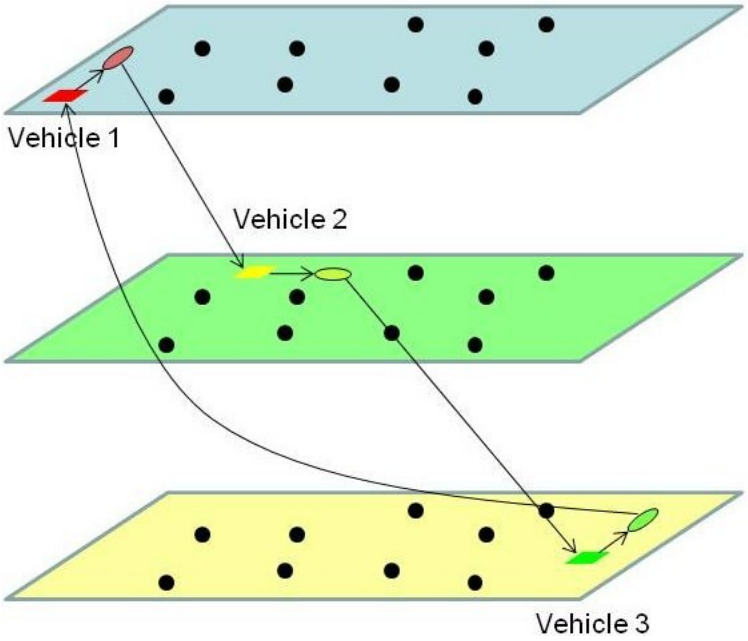


Fig. 22.: Connect vehicle starting postions and virtual nodes

Now interset edges have their end points shifted 'forward'. This can be visualized as moving the end point of all interset edges 'down' into the plane below as is shown in Figure 23. Then M is added to the cost of all interset edges. This process is done to every interset edge, but (again) for the sake of clutter only three such edges are shown.

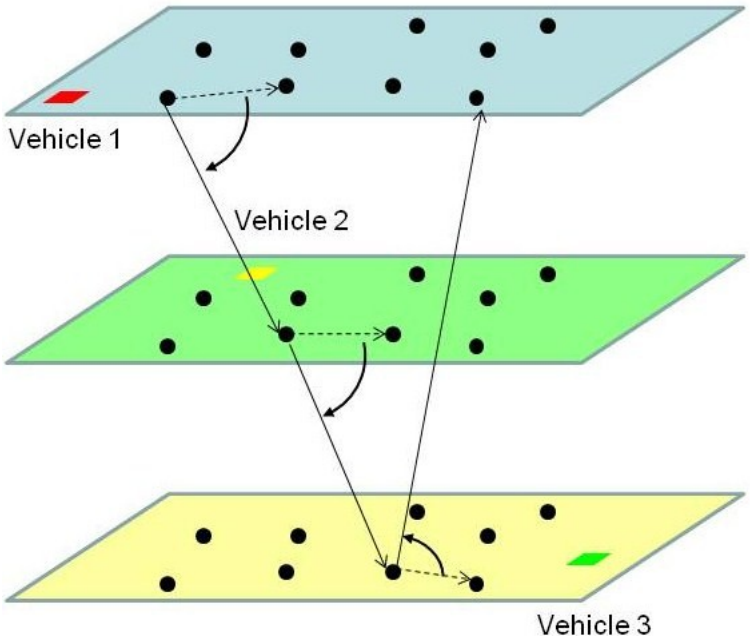


Fig. 23.: Shift interset edge end points 'forward'

Now the ATSP is solved over the modified graph as is shown in Figure 24. In order to avoid clutter edges that connect nodes in the 'plane' of vehicle three to nodes in the 'plane of vehicle one are shown to disappear from the bottom of the figure and reappear at the top of the figure where they meet the proper nodes in the plane of vehicle 1. For the same reason zero cost edges present in the solution are shown as dashed lines. The dashed red edge is the zero cost edge that connects the virtual node corresponding to the third vehicle startig position to the starting position of vehicle 1.

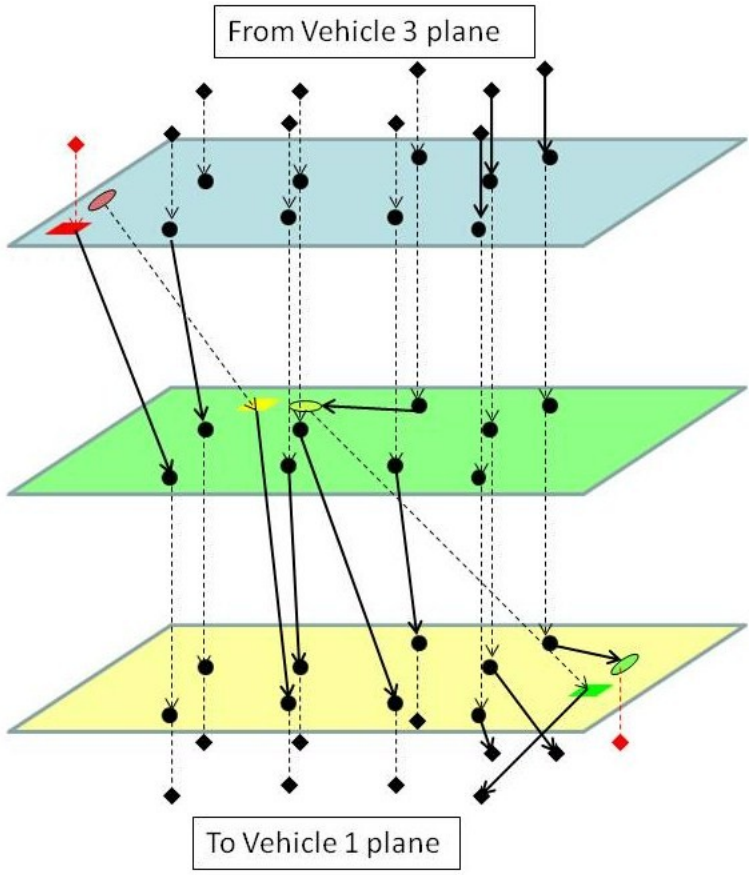


Fig. 24.: Solve ATSP over the modified graph

The solution to the ATSP over the modified graph is now transformed into the solution to the original HTSP. This process can be visualized as projecting every edge present in the ATSP onto the top plane, with the virtual nodes corresponding to vehicle starting positions removed. This is shown in Figure 25.

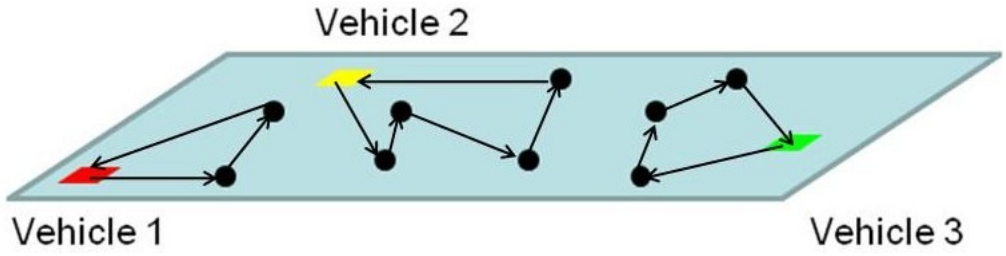


Fig. 25.: Convert One in a Set TSP solution into HTSP solution

In practice, the HTSP is useful and can be used to solve a variety of practical problems. One such practical problem is one in which a set of heterogeneous vehicles must visit a set of nodes that occupy a space which is heterogeneous. In other words, a subset of the vehicles are penalized if they ever occupy a certain region of the space. This can be thought of as a mountainous region that must be surveyed by a set of vehicles that include ground vehicles and air vehicles. Ground vehicles are penalized for visiting areas at a high elevation as they expend more energy to cover the distance as well as the elevation, while air vehicles that cruise at a high enough elevation can fly over areas of any elevation without expending extra energy. With this motivation define a 'forbidden region' as a region which a subset of the vehicles are forbidden from occupying. Define a 'caution region' as a region for which a subset of vehicles incur a penalty as a function of location within the region. A five vehicle instance of this problem was considered. Each vehicle accumulates a cost proportional to the distance it travels, therefore every vehicle has an associated cost per unit distance. The quantity to be minimized was total cost of visiting all the nodes. In the instance considered there was one disk shaped forbidden region centered at $(375, 100)$ with a 75 unit radius and two caution regions. The caution regions covered the entire space and assigned additional cost to vehicles based on a gaussian function. The two caution regions were centered at $(150, 300)$ and $(400, 350)$. The five vehicles behave

as follows: Vehicles 1,2, and 3 have an indentical cost per distance,are constrained by forbidden regions and incur additional cost due to the caution regions. Vehicle 4 has a higher cost per distance than Vehicles 1,2 and 3 and incurs additional cost due to the caution regions only, but is not constrained by the forbidden region. Vehicle 5 has the highest cost per distance and incurs no extra cost, nor is it constrained by the forbidden region. A sample solution to this problem with randomly generated node locations and vehicle starting positions is shown below in Figure 26. In this solution the forbidden region is represented as a red circle, while the caution regions are represented as concentric circles that represent lines of constant penalty.

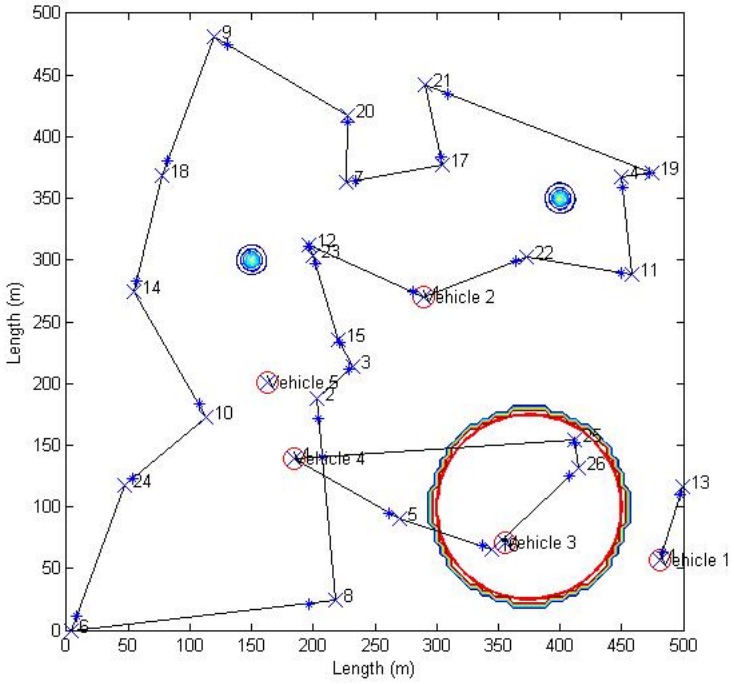


Fig. 26.: Example solution for HTSP with penalty regions

This solution shows that Vehicles 3 and 5 remain inactive. This is expected as vehicle 5 is the most costly to operate, and vehicle 3 has a starting point which is

inside the area which it is forbidden to travel. This problem was formulated using code written in MATLAB and solved using the LKH TSP solver written by Keld Helsgaun [5].

CHAPTER VI

CONCLUSIONS

The PCATSP is a general problem to which many related problems can be transformed, making a quick solution algorithm useful and broad in scope. An underlying structure must be recognized in order to attack the problem efficiently. Sub-optimal solution methods relying on an integer programming formulation with a tight LP relaxation have had mixed results. A split dual algorithm to solve the precedence constrained traveling salesman problem is presented that leverages the relative tightness of the PCATSPxy formulation and the speed of the Lin-Kernighan Heuristic. Computational results show the split dual generates a high quality feasible solution, although the lower bound converges slowly. Further work will include cutting plane methods[14] applied to the relaxed PCATSPxy subproblem of the split dual and the search for a combinatorial structure which can play a role similar to the one that the one-tree plays to the ATSP. For problems which can be posed as a One in a Set TSP, the Noon-Bean transformation is useful tool in their solution as it allows them to be solved a single ATSP. A transformation is shown that allows the Asymmetric traveling salesman problem (ATSP) with motion constraints to be formulated as a single ATSP. The transformed problem is then solved using a standard ATSP solver which gives a high quality solution quickly. A transformation is shown that allows the Heterogeneous traveling salesman problem (HTSP) to be formulated as a single ATSP. The transformed problem can then solved using a standard ATSP solver which gives high quality solutions quickly.

REFERENCES

[1] Subhash C. Sarin, Hanif D. Sherali, and Ajay Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Operations Research Letters*, vol. 33, pp. 62–70, 2005.

[2] Waqar Malik, Sivakumar Rathinam, Swaroop Darbha, and Daniel Jeffcoat, "Combinatorial motion planning of reeds-shepp vehicles," in *Proceedings of the IEEE Conference on Decision and Control*. IEEE, December 2006, pp. 5299–5304.

[3] Saikrishna Yadlapalli, Waqar Malik, Swaroop Darbha, and Meir Pachter, "A lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem," in *Proceedings of the American Control Conference*, 2007, pp. 4027–4032.

[4] Saikrishna Yadlapalli, Waqar Malik, Sivakumar Rathinam, Swaroop Darbha, and Meir Pachter, "A lagrangian-based algorithm for an asymmetric multiple depot, multiple travelling salesmen problem," in *Proceedings of the IEEE Conference on Decision and Control*, December 2007, pp. 5979–5984.

[5] Keld Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[6] J.C. Bean and C.E. Noon, "An efficient transformation of the generalized traveling salesman problem," *Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[7] C. Miller, A. Tucker, and R. Zemlin, "Integer programming formulation of traveling salesman problems," *J. Assoc. Comput. Mach.*, vol. 7, pp. 326–329, 1960.

[8] Manfred Padberg and Ting-Yi Sung, "An analytical comparison of different formulations of the traveling salesman problem," *Mathematical Programming*, vol. 52, pp. 315–357, 1992.

[9] H.P. Williams and A.J. Orman, "A survey of different integer programming formulations of the travelling salesman problem," in *Optimisation, Econometric and Financial Analysis*, E.J Kontoghiorghes and C. Gatu, Eds., vol. 9, pp. 91–104. Springer, Berlin, 2007.

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

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

[12] H.D. Sherali and P. Driscoll, "On tightening the relaxations of miller-tucker-zemlin formulations for asymmetric traveling salesman problems," *Operations Research*, vol. 50, no. 4, pp. 656–669, 1991.

[13] 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*, pp. 19–28, 1990.

[14] Egon Balas, Matteo Fischetti, and William R. Pullyblank, "The precedence constrained asymmetric traveling salesman polytope," *Mathematical Programming*,

vol. 68, pp. 241–265, 1995.

[15] 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*, vol. 3, pp. 25–42, 1993.

[16] 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*, vol. 17, no. 1, pp. 61–84, 2000.

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

VITA

Name:	Paul Victor Oberlin
Address:	3123 TAMU, College Station TX 77843-3123
Email:	paul.v.oberlin@gmail.com
Education:	B.S. Mechanical Engineering, Texas A&M University, 2007
	M.S. Mechanical Engineering, Texas A&M University, 2009