An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem

Quan Lu
Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089-0193
Phone: (213) 740-8585   Fax: (213) 740-4016
qlu@scf.usc.edu

Maged Dessouky*
Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089-0193
Phone: (213) 740-4891   Fax: (213) 740-1120
maged@usc.edu

December 12, 2002

* Corresponding Author

An Exact Algorithm
for the Multiple Vehicle Pickup and Delivery Problem

ABSTRACT

We consider the Multiple Vehicle Pickup and Delivery Problem (MVPDP) with objective of minimizing the total travel cost and the fixed vehicle cost. Most of the optimization based approaches for solving the MVPDP are developed for a restrictive hard time window or tight capacity environment that depend significantly on the reduction of the feasible solution space. We develop an alternative optimization solution approach for the MVPDP that does not require these constraints to be tight. The problem is formulated as a 0-1 integer-programming problem. A branch-and-cut algorithm is developed to optimally solve the problem. Four classes of valid inequalities for the MVPDP are proposed. By using the proposed solution approach, we were able to optimally solve problem instances of up to 5 vehicles and 17 customers on problems without clusters and up to 5 vehicles and 25 customers on problems with clusters within a stopping criterion of three CPU hours on a Sun Fire 4800 Server.

Introduction

The Pickup and Delivery Problem (PDP) generally consists of a fleet of vehicles and a set of customer requests. Each request specifies the size of the load to be transported and the locations of two stops: pickup and delivery points. Furthermore, the same vehicle must visit the pickup location before the delivery location. The pickup portion of the request is considered *serviced* when all the units to be transported of the request are on board the vehicle while the delivery portion of the request is considered *serviced* when all the units are disembarked off the same vehicle. Each vehicle has a given capacity, a start location and an end location. The objective is to minimize the total cost, which may include the fixed vehicle cost and the travel cost, while satisfying all customer demand.

The PDP has received attention by many researchers over the last two decades. This interest is due in part to its practical importance. For example, the well-known Dial-a-Ride Problem (DARP) can be considered as a PDP in which the loads to be transported represent people and with some special service-constraints such as maximum ride time. Therefore the size of a load and vehicle capacity must be integer. The rapid growth of the parcel transportation industry due to increasing e-commerce demand and the growth of the paratransit industry due to the American with Disabilities Act (ADA) further highlight the significance of the PDP.

The PDP is known to be NP-hard in the strong sense. Due to its computational difficulty, prior work on exact algorithmic development for the PDP has focused either on the single vehicle problem (referred to as SVPDP) or problems with hard time windows (referred to as PDPHTW).

Psaraftis (1980) proposed the first exact algorithm for the SVPDP, which was based on dynamic programming. Kalantari, Hill, and Arora (1985) developed a branch-and-bound algorithm for the SVPDP, and problem instances with up to 15 customers were optimally solved with this algorithm. Fischetti and Toth (1989) solved the problem by using an additive bounding approach in a branch-and-bound algorithm. Ruland and Rodin (1997) presented a branch-and-cut algorithm, and problem instances with up to 15 customers were optimally solved.

In the PDPHTW, a time window is associated with each stop of the request whether it is a pickup or delivery point. A feasible schedule satisfies all requests within the given time window. The research interest in the PDPHTW is aroused partially by the increasing number of practical applications such as paratransit vehicle routing. With hard time windows, it may be possible to optimally solve moderate size problems using dynamic programming algorithms since hard time windows can reduce the solution space by eliminating a large number of states. The two keys in implementing a dynamic programming approach are state definition and state elimination. Psaraftis (1983) presented a forward dynamic programming algorithm to solve the single-vehicle PDPHTW. Desrosiers, Dumas, and Soumis (1986) provided an alternative dynamic programming algorithm for the same problem. Dumas, Desrosiers, and Soumis (1991) developed a column generation solution procedure to optimally solve the PDPHTW with up to 55 customers using a single vehicle. Their approach works well only under either restrictive capacity or time window constraints. For example, in their problem instance of 55 customers, the restrictive capacity and time window constraints enabled them to significantly reduce the number of arcs in the problem to 1996 from a total of over

12,000. Clearly, the tighter the time window and capacity constraints, the more difficult it is to identify a feasible schedule. However, tight capacity and time window restrictions reduce the solution space significantly, thus making the above approaches computationally tractable to find optimal solutions for small to moderate size problems.

The focus of this paper is to develop an exact solution procedure for the Multiple Vehicle Pickup and Delivery Problem (MVPDP) that does not depend on elimination techniques for reducing the solution space due to violations in capacity and time window constraints. That is, we consider problems where these constraints are not tight. Clearly, problems without hard time windows are similar to problems with hard time windows with arbitrarily large windows. However, as earlier stated, large time windows make dynamic programming approaches computationally intractable. There are many applications such as carrier delivery in which the vehicle capacity is not restrictive and the time windows are sufficiently large (e.g, end of business day). Savelsbergh and Sol (1995) provide other examples where there are not restrictive time window constraints.

Our solution approach is based on a branch-and-cut algorithm. In §4, we show that our algorithm can optimally solve problem instances of up to 5 vehicles and 17 customers on problems without clusters and 5 vehicles and 25 customers on problems with clusters within a stopping criterion of three CPU hours on a SUN Fire 4800 System. Although our focus is on exact solution procedures, we note that there also has been a large body of work that focused on heuristic development for the MVPDP due to the combinatorial nature of the problem. Savelsbergh and Sol (1995) provide an excellent review of this work. More recent approximation solution procedures include the column management scheme by Savelsbergh and Sol (1998), the insertion heuristic developed by

Madsen, Ravn, and Rygaard (1995), the clustering algorithm developed by Ioachim et al. (1995), and the parallel insertion heuristic developed by Toth and Vigo (1997).

The remainder of the paper is organized as follows. In §1, we present a new mixed integer linear programming formulation for the MVPDP that only introduces integer variables representing arc flow. That is, no additional binary variables are needed to represent pairing the pickup with the delivery request on the same vehicle. Furthermore, our formulation does not include an extra index for the binary variables to represent vehicle number. Several classes of valid constraints are then identified in §2. In §3, the details of the branch-and-cut procedure including the prediction strategy and branch methods are presented. Computational results are shown in §4, followed by some concluding remarks in §5.

1. Formulation

We provide an integer programming formulation for the PDP with multiple heterogeneous vehicles, which implies that each vehicle may have its own capacity, depart depot, and return depot. The objective is to find a collection of at most *m* simple circuits (*m* is the number of vehicles in the fleet) with minimum total costs, defined as the sum of the travel costs, and fixed operating cost for each used vehicle.

Let there be n customers. Each customer has a pickup and a delivery request. Node i represents the pickup node for customer i and node n+i represents the delivery node for customer i. Denote N^+_r ={1, 2, …, n} the node set for all pickup requests and N^-_r ={n+1, n+2, …, 2n} the node set for all delivery requests. For example, node 1 and node n+1 represent the pickup and delivery locations for customer 1, respectively. Let $N_r = N^+_r \cup N^-_r = \{1, 2, \ldots, 2n\}$ represent the node set for all service stops.

Let there be m vehicles to be routed and scheduled. In our network we represent the departure and return depots for all vehicles by m+1 nodes. They are node set $N_q = \{2n+1, 2n+2, \ldots, 2n+m+1\}$. Node 2n+1 represents the departure node for the first vehicle. Nodes 2n+v, v=2, 3, …, m represent the return node for the $(v-1)^{th}$ vehicle as well as the departure node for the v^{th} vehicle. Node 2n+m+1 represents the return node for the m^{th} vehicle.

We remark that extending the network from a single vehicle to a multiple case for a VRP is straightforward since there is no pairing constraint that requires a pickup to be matched with the delivery on the same vehicle. That is, in the PDP the two nodes representing the pickup and delivery points of a request must be on the same path for a given vehicle. Our network design takes into consideration this pairing constraint while attempting to keep the number of arc flow variables to a minimum.

The network for the problem can be constructed as follows: G (N, A) where the node set $N = N_r \cup N_q$ and A is the arc set. Let $A = A_r \cup A_q \cup A_{r,q}$, where A_r represents the set of arcs between nodes of N_r, A_q represents the set of arcs between nodes of N_q, and $A_{r,q}$ represents the set of arcs connecting N_r and N_q. In A_r we exclude all arcs (i+n, i), $1 \leq i \leq n$ because they violate the requirement of pickup before delivery. An example of graph A_r for a two customer case is shown in Figure 1a, where nodes 1 and 2 represent the pickup nodes and nodes 3 and 4 represent the delivery nodes of customer 1 and 2 respectively. Arc set A_q has exactly m+1 arcs that formulate a loop. Figure 1b shows the arcs of type A_q in a two-vehicle case. In this example node 5 represents the departure node of vehicle 1, node 6 represents the return node of vehicle 1 and the departure node of vehicle 2, and node 7 represents the return node of vehicle 2. Note that the nodes in N_q can represent two different physical locations. Let arc set $A_{r,q} = A_{r-,q} \cup A_{q,r+}$. Arc set $A_{r-,q}$ represents all arcs from N_r^- to $N_q/\{2n+1\}$. Arc set $A_{q,r+}$ represents all arcs from $N_q/\{2n+m+1\}$ to N_r^+. An example of the entire graph for n=2 and m=2 case is showed in Figure 1c. Note that the graph is not a complete graph. This enables us to minimize the number of arcs that will be the integer variables in the formulation.

Let $p_{i,j}$ be the travel cost for arc (i, j) in arc set A. The definition is as follows

$$p_{i,j} = \begin{cases} dis\tan ce\ between\ nodes\ i\ and\ j & (i,j) \in A_r \\ dis\tan ce\ between\ departure\ depot\ of\ vehicle\ i-2n & \\ \qquad and\ pickup\ node\ j & (i,j) \in A_{q,r^+} \\ dis\tan ce\ between\ delivery\ node\ i\ and\ return\ depot & \\ \qquad of\ vehicle\ j-2n-1 & (i,j) \in A_{r^-,q} \\ 0 & (i,j) \in A_q \end{cases}$$

Let U_v be the fixed cost incurred by using vehicle v, v = 1, 2, …, m. Define the cost $c_{i,j}$ on each directed arc connecting node i to j in arc set A as follows.

$$c_{i,j} = \begin{cases} p_{i,j} & (i,j) \in A_r \cup A_q \cup A_{r^-,q} \\ U_{i-2n} + p_{i,j} & (i,j) \in A_{q,r^+} \end{cases}$$

From the definition of the above network, it is easy to see that any feasible solution to the pickup and delivery problem can be represented as a Hamiltonian tour in this network. Hence, the problem can be defined to identify the minimal cost Hamiltonian tour that satisfies all the pickup and delivery constraints in this new graph.

Denote the capacity of each vehicle by D_v, v=1, 2, …, m and d_i the pickup demand at each pickup node i = {1, 2, ..., n}. To represent the vehicle capacity constraints in our formulation, two additional variables f_i and g_i are defined as follows,

$$f_i = \begin{cases} D_1 & i = 2n+1 \\ D_{i-2n} - D_{i-2n-1} & i = 2n+2,\ 2n+3,\ ...,\ 2n+m \\ 0 & Otherwise \end{cases}$$

$$g_i = \begin{cases} d_i & i \in N_r^+ \\ -d_{i-n} & i \in N_r^- \\ 0 & Otherwise \end{cases}$$

This problem is formulated as an integer programming problem. We define two binary decision variables:

$$x_{i,j} = \begin{cases} 1 & \textit{if a vehicle travels from node i to node } j \\ 0 & \textit{otherwise} \end{cases}$$

$$b_{i,j} = \begin{cases} 1 & \textit{if node i is before node } j \textit{ in the tour} \\ 0 & \textit{otherwise} \end{cases}$$

Note that to define the order of nodes in a Hamiltonian tour, we assume that the tour starts from node 2n+1, which is the departure depot for the first vehicle, and ends in node 2n+m+1, which is the return depot for the last vehicle.

We next present a mathematical formulation of the MVPDP referred to as F1. Our formulation does not require an extra index for vehicle number in the arc flow variables to enforce the pairing of the pickup and delivery requests.

$$Min \sum_{i \in N} \sum_{j \in N} c_{i,j} x_{i,j}$$

subject to

$$\sum_{j \in N} x_{i,j} = 1 \qquad for\ all\ i \in N \qquad (1)$$

$$\sum_{i \in N} x_{i,j} = 1 \qquad for\ all\ j \in N \qquad (2)$$

$$b_{k,i} \le b_{k,j} + (1 - x_{i,j}) \qquad for\ all\ arc(i,j) \in A / \{arc\ (2n+m+1,\ 2n+1)\}\ and\ k \in N / \{i\} \qquad (3)$$

$$b_{k,j} \le b_{k,i} + (1 - x_{i,j}) \qquad for\ all\ arc(i,j) \in A / \{arc\ (2n+m+1,\ 2n+1)\}\ and\ k \in N / \{i\} \qquad (4)$$

$$x_{i,j} \le b_{i,j} \qquad for\ all\ arc(i,j) \in A \qquad (5)$$

$$b_{i,i} = 0 \qquad for\ all\ i \in N \qquad (6)$$

$$b_{n+i,i} = 0 \qquad for\ all\ i \in N_r^+ \qquad (7)$$

$$b_{i,n+i} = 1 \qquad for\ all\ i \in N_r^+ \qquad (8)$$

$$b_{i,2n+j} = b_{n+i,2n+j} \qquad for\ all\ i \in N_r^+\ and\ 2n+j \in N_q \qquad (9)$$

$$g_j + \sum_{i \in N} b_{i,j} g_i \le \sum_{k \in N_q} f_k b_{k,j} \qquad for\ all\ j \in N_r^+ \qquad (10)$$

$$b_{i,2n+1} = 0 \qquad for\ all\ i \in N \qquad (11)$$

$$b_{2n+k,2n+j} = 1 \qquad for\ all\ k < j\ and\ 2n+k \in N_q\ and\ 2n+j \in N_q \qquad (12)$$

$$b_{2n+j,2n+k} = 0 \qquad for\ all\ k < j\ and\ 2n+k \in N_q\ and\ 2n+j \in N_q \qquad (13)$$

$$b_{i,2n+m+1} = 1 \qquad for\ all\ i \in N / \{2n+m+1\} \qquad (14)$$

$$x_{i,j} = \{0,\ 1\} \qquad (15)$$

$$b_{i,j} = \{0,\ 1\} \qquad (16)$$

Constraints (1) and (2) are the basic network constraints, which impose that exactly one vehicle enters and leaves each service vertex.

Constraints (3) and (4) are *copy constraints*, because their function is to copy all the values of $b_{k,i}$ to $b_{k,j}$, where $k \in N$ and $k \neq i$. That is, if $x_{i,j} = 1$, meaning node i is immediately before node j, constraints (3) and (4) force $b_{k,i} = b_{k,j}$ for all $k \in N$ and $k \neq i$. Then constraint (5) enforces $b_{i,j} = 1$ and condition (6) defines $b_{i,i}=0$. When $x_{i,j} = 0$, constraints (3) to (5) become redundant.

Constraints (7) and (8) are *prior constraints*, which force the pickup node to be visited before the corresponding delivery node.

Constraint (9) is the *pairing constraint*, which ensures that the pickup and delivery requests for each customer must be satisfied by the same vehicle.

Constraint (10) is the *vehicle capacity constraint*, which enforces that at any time the load on the vehicle is under the vehicle capacity. Because the vehicle capacity can only be violated at the pickup locations, the load of the vehicle is checked only at each pickup load. At each pickup node j, the capacity of the vehicle that will visit node j is $\sum_{k \in N_q} f_k b_{k,j}$. The current load of the vehicle before visiting node j is $\sum_{i \in N} b_{i,j} g_i$.

Constraint (11) sets node 2n+1 as the first node in the tour. Constraints (12) and (13) enforce that the departure depots of the vehicle 1, …, m, are visited in this sequence in the tour. Recall that nodes 2n+v, v=2, 3, …, m represent the return node for the (v-1)th vehicle as well as the departure node for the vth vehicle. These two constraints enforce that each vehicle returns to its own depot. Finally constraint (14) sets node 2n+m+1 as the last node in the tour.

A characteristic of the formulation is that the subtour elimination has been implicitly included in the formulation. In other words, any integer solution of the above formulation must represent a Hamiltonian tour in the network.

Proposition 1. There doesn't exist any subtour in any feasible solution of Formulation F1 including the linear relaxation of constraint (16).

PROOF:

The proof is by contradiction. Suppose part of the x variables generate a subtour (n_i, n_{i+1}, …, n_{i+z}, n_j, n_i), where n_i, n_{i+1}, …, n_{i+z}, n_j are arbitrary z+2 number of different nodes in the network ($z < 2n+m-1$), n_i and n_j are two arbitrary adjacent nodes in this subtour. Because $x_{j,i}=1$, constraint (5) enforces $b_{j,i} = 1$. On the other hand, since $x_{i,i+1} = 1$, $x_{i+1,i+2} = 1$, …, $x_{i+z,j} = 1$ and $j\neq i$, $j\neq i+1$, …, $j\neq i+z$ copy constraints (3) and (4) force $b_{j,i} = b_{j,i+1} = b_{j,i+2} = \ldots = b_{j,i+z} = b_{j,j}$. Therefore, $b_{j,j} = b_{j,i} = 1$, which is a contradiction of constraint (6). □

Because of the *b* variables, we do not need a third index in the arc flow variables to represent vehicle number in order to enforce the pairing constraints. Furthermore, the 0-1-integer restriction for the *b* variables (constraint (16)) can be relaxed. This relaxation reduces the number of integer variables by around a half and makes the above MVPDP have only arc flow binary variables equivalent to multiple vehicle routing problems.

Proposition 2. The b variables must be binary in any feasible solution to Formulation F1 with the linear relaxation of constraint (16).

PROOF:

First let {(X, B)} be the solution set including all feasible solutions of the above formulation with the linear relaxation of the b variables, and $\{(X^{'}, B^{'})\}$ be the feasible solution set of the above formulation. Clearly, $\{(X^{'}, B^{'})\}\subset\{(X, B)\}$. On the other hand, for any specific feasible solution $(x, b)\in\{(X, B)\}$, constraint (11) forces all $b_{k,2n+1}=0$, for $k\in N$ and constraint (1) ensures that there exists a node y, where $x_{2n+1,y} = 1$, $y\in N$. Constraint (5) sets $b_{2n+1,y} = 1$ and copy constraints (3) and (4) force all other $b_{k,y} = 0$, $k\in N/\{2n+1\}$. Thus all $b_{k,y}$ are binary for $k\in N$. This process can be repeated starting from node y and until all nodes in node set N are evaluated. Hence, all b variables are binary. Note that all nodes in N will be evaluated is due to Proposition 1, which ensures that no subtour will be created in any feasible solution to the problem including the linear relaxation of constraint (16). Therefore, (x, b) $\in\{(X', B')\}$ and then $\{(X, B)\}\subset\{(X', B')\}$. □

The reverse relationship between the x and b variables does not hold. That is, given that the b variables are integer, it does not imply that the x variables will be integer. This is due to the inequality constraint (5), which is a one way constraint and can only set $b_{i,j}$ from $x_{i,j}$.

Because we only need to enforce the integrality of the arc flow variables, our formulation has $4n^2+2mn+(m+1-3n)$ integer variables. It saves around $O(mn^2)$ integer variables comparing with the formulation of Desaulniers et al. (2000). The addition of more vehicles does not significantly increase the number of binary variables in our formulation.

A simple extension of the above formulation can be used to solve the PDPHTW. Let variable T_i be the time at which service at node i begins, $i\in N_r$. $t_{i,j}$ and s_i represent the travel time from node i to j and the service time at node i, arc $(i, j)\in A$. Next, let $[e_i, l_i]$ denote the service time window for each pickup or delivery node i, i={1, 2, …, 2n}. Let also E_v denote the earliest start time for vehicle v and L_v denote the latest return time for vehicle v, v={1, 2, …, m}. Then the following constraints (17) to (20) can be used to deal with the hard time window case, where M is a very big number.

$$T_i + t_{i,j} + s_i \le T_j + (1 - x_{i,j})M \qquad \textit{for all arc } (i, j) \in A_r \tag{17}$$

$$E_i + t_{2n+i,j} + s_{2n+i} \le T_j + (1 - x_{2n+i,j})M \qquad \textit{for all arc } (2n+i, j) \in A_{q,r^+} \tag{18}$$

$$T_i + t_{i,2n+j} + s_i \le L_{j-1} + (1 - x_{i,2n+j})M \qquad \textit{for all arc } (i, 2n+j) \in A_{r^-,q} \tag{19}$$

$$e_i \le T_i \le l_i \qquad \textit{for all } i \in N_r \tag{20}$$

Constraint (17) 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.

For the PDPHTW there exists an optimal solution that all used vehicles start at their earliest possible start time, therefore constraint (18) forces the first service time for each vehicle assuming that vehicle i starts at time E_i. Constraint (19) ensures that no vehicle violates the time window in the return depot, and constraint (20) ensures that the service start time is within the time window.

2. Identification of Valid Inequalities

Let Γ denote the set that includes all the feasible solutions satisfying constraints (1) to (16). Then Conv(Γ), the convex hull of Γ, is the MVPDP polytope. Let Γ_{relax} include all feasible solutions of the LP relaxation satisfying constraints (1) to (16). Then Conv(Γ) $\subset$ Conv(Γ_{relax}). In order to optimally solve the formulation with a branch-and-cut algorithm, some tight inequalities must be identified to make Conv(Γ_{relax}) approach Conv(Γ). The ideal case is when these inequalities induce all the facets of Conv(Γ). Then, Conv(Γ) =Conv(Γ_{relax}) and no branching is necessary. This section describes four classes of equalities/inequalities used in our algorithm to cut Conv(Γ_{relax}). We next prove that they are valid inequalities for Conv(Γ). We use the term valid inequality to mean that it will not eliminate any extreme points of Conv(Γ).

2.1. Transfer Prior Constraint

Proposition 3 (Transfer Prior Constraint) For any pair of pickup and delivery nodes i and n+i $\in N_r$ *and an arbitrary collection of nodes* $(h_1, h_2, \ldots, h_k) \in N/\{i, n+i\}$, $1 \le k \le |N| - 2$,

$$b_{n+i,h_1} + \sum_{j=1}^{k-1} b_{h_j,h_{j+1}} + b_{h_k,i} \le k \qquad (21)$$

is a valid inequality.

PROOF.

The proof is by contradiction. Suppose the above inequality is not satisfied we have

$$b_{n+i,h_1} + \sum_{j=1}^{k-1} b_{h_j,h_{j+1}} + b_{h_k,i} \ge k+1.$$

Since there are exactly k+1 binary variables in the inequality, we have

$$b_{n+i,h_1} + \sum_{j=1}^{k-1} b_{h_j,h_{j+1}} + b_{h_k,i} = k+1$$

From the integer characteristic of the b variables (Proposition 2), we have

$$b_{n+i,h_1} = 1,\ b_{h_j,h_{j+1}} = 1 \ where\ j = 1,\ldots,k-1,\ and\ \ b_{h_k,i} = 1.$$

From $b_{n+i,h_1} = 1\ and\ \ b_{h_j,h_{j+1}} = 1\ where\ j = 1,\ldots,k-1,$ we know that node n+i is before node h$_k$. But from $b_{h_k,i} = 1$ node h$_k$ is before node i. Therefore the prior constraint is violated. Hence, $b_{n+i,h_1} + \sum_{j=1}^{k-1} b_{h_j,h_{j+1}} + b_{h_k,i} \le k$. □

2.2. Adjacent Prior Constraint

Proposition 4. (Adjacent Prior Constraint). For any pair of pickup and delivery nodes i and i+n∈N$_r$ and an arbitrary node k, k∈N/{i, i+n},

$$b_{k,i} + b_{k,i+n} \ge x_{i,k} + x_{k,i} \qquad (22)$$
$$b_{i,k} + b_{i+n,k} \ge x_{i+n,k} + x_{k,i+n} \qquad (23)$$

are two valid inequalities.

PROOF.

We prove inequality (22). If node k and node i are not adjacent, then the right part of inequality (22) equals 0. It’s satisfied.

If node i is immediately before node k, then node i+n must be after node k. Thus $b_{k,i} = 0$ and $b_{k,i+n}=1$. It’s satisfied.

If node i is immediately after node k, then node i+n must be after node k too. Thus $b_{k,i} = 1$ and $b_{k,i+n}=1$. It’s satisfied.

Inequality (23) can be proved similarly. □

2.3. Pairing Prior Constraint

Proposition 5. (Pairing Prior Constraint). For any pair of pickup and delivery nodes i and i+n $\in N_r$,

$$\sum_{k=1}^{2n+m+1} b_{k,i} + 1 \le \sum_{k=1}^{2n+m+1} b_{k,i+n}. \qquad (24)$$

PROOF. For any node $i \in N$, let $S_i = \sum_{k=1}^{2n+m+1} b_{k,i}$. From the definition of the b variables, S_i represents the sequence number of node i in the tour (e.g., $S_{2n+1}=0$ and $S_{2n+m+1}=2n+m$). Therefore $S_i + 1 \le S_{i+n}$. □

2.4. Valid Equalities

One of the constraints for the MVPDP is that each pair of pickup and delivery requests must be satisfied by the same vehicle. Although constraint (9) ensures this requirement is met for integer solutions, we can add another equality referred to as Vehicle Return

Constraint that cuts some fractional solutions that violate this requirement. For any node i, i$\in$N$_q$,

$$\sum_{k=1}^{2n+m+1} b_{k,i} g_k = 0 \qquad (25)$$

This equality is a valid cut because the load of each vehicle must equal zero after the vehicle returns to the depot.

Another trivial but useful equality which can be derived directly from the definition of the b variables is as follows

$$b_{i,j} + b_{j.i} = 1 \qquad \textit{for all } i, j \in N \textit{ and } i \neq j \qquad (26)$$

2.5. Computational Savings of Using Valid Inequalities

In Table 1 we compare the quality of the lower bound Z_{LP} obtained in the root of the branch-and-cut tree, the integrality gap between Z_{LP} and the optimal solution Z_{OPT}, and the entire computational time with and without the proposed valid inequalities. This experiment uses Solomon's benchmark problems for the vehicle routing problem. These problems can be found in the following web page:

http://www.cba.neu.edu/~msolomon/problems.htm

Since the benchmark problems are intended for the vehicle routing problem, we randomly selected pickup and delivery pairings. The first column of Table 1 shows the number of customers for the problem. The number of nodes used from the benchmark problems is twice the number of customers plus two since these experiments assume a single vehicle. The node sets are derived from Solomon's benchmark problem R101.

Num. of Cust.	Without using valid inequalities				Using valid inequalities			
	Z_{LP}	Z_{OPT}	Gap (%)	CPU Time (Sec.)	Z_{LP}	Z_{OPT}	Gap (%)	CPU Time (Sec.)
5	151.1	154	1.8	1.7	154.0	154	0.0	0.8
8	212.3	240	11.5	109.1	231.4	240	3.6	46.5
10	247.0	276	10.5	712.4	265.2	276	3.9	293.5
12	290.5	318	8.6	905.3	310.3	318	2.4	382.4

Table 1 Computational Comparison between using Valid Inequalities and without using Valid Inequalities

From the above Table 1, we can observe that using the proposed valid inequalities clearly outperforms without using them. Note that a more detailed analysis of the complete solution procedure is provided in §4. These experiments are meant to show the usefulness in the computation of these proposed valid inequalities.

3. Branch-and-cut Algorithm

We next describe a branch-and-cut algorithm that finds an optimal solution for the MVPDP. The performance of this algorithm will be analyzed through computational experiments in §4.

3.1. Bounding

At each node of the search tree, a linear relaxation is defined by (1) to (16) as well as additional constraints (21) to (26). Note that the sum of the number of constraints of type (3) and (4) for a n-customer problem is around $16n^3$ and the number of constraints of type

(5) is around $4n^2$. When $x_{i,j} = 0$, the corresponding 4n+2m+1 constraints (3) to (5) will be redundant. According to our computational experience, a large number of x variables out of the total of around $4n^2+2mn$ equal to zero in each iteration. There is no need to include all these redundant constraints at every step of the process. Therefore we treat constraints (3) to (5) in the same manner as constraints (21) to (26). That is, they will be added to the problem only if they are violated when not included.

A computational comparison between including all constraints from (3) to (5) in the formulation with adding them iteratively is shown in Table 2. Similar to the previous experiments, we selected Solomon's benchmark problem R101 and R102 for this set of experiments. Since the benchmark problems are intended for the vehicle routing problem, we randomly selected pickup and delivery pairings. The first column of Table 2 shows the number of customers for the problem. The number of nodes used from the benchmark problems is twice the number of customers plus two since these experiments assume a single vehicle. When the number of customers is 50 or less, we strictly used the data set R101. When the number of customers is greater than 50, we used both R101 and R102 problems, since each of these problems have a maximum of a 101 locations.

The second and third columns of Table 2 show the CPU Time for solving the LP-relaxation by CPLEX when including all constraints (3) to (5) in the problem and the number of these types of constraints, respectively. Columns four and five show the CPU time of solving the LP relaxation if these constraints are added iteratively and the total number of these types of constraints that were iteratively added in finding the optimal solution to the LP relaxation, respectively. As the table shows, the iterative procedure can greatly reduce the solution time of solving the LP relaxation.

Number of Customers	CPU Time for Solving the Entire LP Problem (Sec.)	Total Number of Constraints (3) to (5)	CPU Time for Solving the LP Problem Iteratively (Sec.)	Number of Constraints (3) to (5) Added
20	1	123,260	1	61
40	21	1,004,920	5	151
60	138	3,412,980	13	189
80	505	8,115,440	50	274
100	1536	15,880,300	188	420

Table 2 Computational Comparison between Including all Constraints from (3) to (5) in the Formulation with Adding them Iteratively

This table is meant to illustrate the potential benefit of the iterative procedure over solving the entire LP-relaxation.

The complete steps of the algorithm are shown in Figure 2. At each node, we first solve the LP-relaxation of the problem including only constraints (1), (2), and (6)-(16). The additional constraints are only included to the LP-relaxation if they are violated. We note that although the integer solution of our formulation does not include any subtours (Proposition 1), subtour elimination constraints are necessary for fractional solutions to improve the lower bound. We used the algorithm from Padberg and Rinaldi (1990,1991) to identify the subtour elimination inequalities for fractional solutions. The trigger level of 6n, where n is the number of customers, in step 4 and the trigger level of 4 in step 7 were experimentally determined. Using zero as a trigger level may cause the addition of some constraints to the LP-relaxation that will be naturally eliminated in a subsequent step of the procedure. That is, these triggers tend to reduce the number of repeated LPrelaxations that needed to be solved. We will discuss our branching strategy, integer prediction, and elimination of ineffective constraints next.

3.2. Branching

When the bounding procedure fails to yield an integer solution, branching is performed on one of the fraction nodes of the branching decision tree. We have tested two different branching strategies.

The first branching method is the adaptation of the common fractional variable branching scheme. First, define $Q_{i,j}$, where i<j, as follows,

$$Q_{i,j} = \begin{cases} x_{i,j} + x_{j,i} & \textit{where } arc(i,j) \in A \textit{ and } arc(j,i) \in A \\ x_{i,j} & \textit{where } arc(i,j) \in A \textit{ and } arc(j,i) \notin A \\ x_{j,i} & \textit{where } arc(i,j) \notin A \textit{ and } arc(j,i) \in A \end{cases}$$

The fractional $x_{i,j}$ variable with the highest $Q_{i,j}$ value, which is incident to the nodes connected by the edges that is fixed to one, will be given the highest priority to be branched next. Branching creates two or three child nodes. If arc $(i, j) \in A$ and arc $(j, i) \in A$, there are three child nodes: $x_{i,j} = x_{j,i} = 0$; $x_{i,j} = 1$ and $x_{j,i} = 0$; $x_{i,j} = 0$ and $x_{j,i} = 1$; Otherwise two child nodes are generated.

The second branching method is due to the characteristic of the formulation. In §2.3, we defined $S_i = \sum_{k=1}^{2n+m+1} b_{k,i}$, $1 \le i \le 2n+m+1$, for each node i and pointed out that S_i implies the sequence number of node i in the tour. Therefore, solving the MVPDP can also be expressed as finding an assignment of the integer values 0, 1, …, 2n+m to the integer variables $S_1, S_2, \ldots, S_{2n+m+1}$ satisfying the capacity constraints, prior constraints, and pairing constraints. At each fractional node, we calculate $Q'_{i,j} = |S_i - S_j|$ for each pair of i, j, i<j and $1 \le i, j \le 2n+m+1$. We branch the pair (i, j) with the minimal $Q'_{i,j}$ value. Two child nodes are created. One has the new constraint $S_i - S_j > 1$ and the other has the new constraint $S_i - S_j \le 1$.

3.3. Integer Prediction

In the branch and bound algorithm, finding a good integer solution early can reduce the number of explored nodes, which makes the procedure more efficient. At each fraction node in our branching decision tree, we predict the corresponding integer solution from the current fractional solution. Our prediction procedure relies on the S_i variables too. First the S_i for each node i is calculated, then the nodes are sorted in increasing order of S_i. Assume they are $\{L_1, L_2, \ldots, L_{2m+n+1}\}$. We know it is a Hamiltonian tour in our network, and due to the *Pairing Prior Constraint* in §2.3 all prior constraints have been satisfied. But because of possible fractional values of the solution, the pairing and capacity constraints may not be satisfied. Thus we need to check whether these two constraints are satisfied. If not, a simple heuristic procedure is applied to adjust the sequence to satisfy all these constraints. The heuristic is as follows. First, the violations of the pairing constraints are resolved. Assume there are j pairing constraints violated. For each of these J customers, the pairing constraint can be satisfied by either inserting the pickup location to the tour of the vehicle visiting only the delivery location or inserting the delivery location to the tour of the vehicle visiting only the pickup location. The option with the lower total travel cost is chosen. Then the violations of the capacity constraints are resolved by moving some delivery request forward within the same vehicle's tour. Now we have a feasible solution for the MVPDP. Its cost will be computed and compared with the current best integer solution. If it is better than the current best integer solution, it will be set as the new best integer solution.

In Table 3 we compared the elapsed CPU time and the number of explored nodes of finding the optimal integer solution with and without using prediction. Similar to the previous experiments, we selected Solomon's benchmark problem R101 for this set of experiments. The first column in Table 3 shows the number of customers. The second and third columns in the table show the CPU time and the number of explored nodes without using the prediction strategy. The fourth and fifth columns show the CPU time and the number of explored nodes using prediction. The sixth column shows the average number of the violated pairing constraints at each branching node. Note that the saving in the CPU time is not as much as the saving in the number of the explored nodes due to the additional computation time needed in the prediction. As the table shows, prediction shows a slight benefit in reducing the CPU time of finding an optimal solution.

Num. of Cust.	Without Using Prediction		Using Prediction		
	CPU Time of Finding Optimal Solution (Sec.)	Number of Nodes Explored When Finding Optimal Solution	CPU Time of Finding Optimal Solution (Sec.)	Number of Nodes Explored When Finding Optimal Solution	Average Number of Violated Pairing Constraints
5	1.6	9	1.3	7	0
8	23.5	109	14.7	86	0.3
10	126.5	176	102.7	143	0.8
15	2876.0	1038	2335.4	895	2.1

Table 3 Computational Comparison between using Prediction and without using Prediction

3.4. Eliminate Ineffective constraints

From the above description, the procedure of iteratively solving the LP relaxation by adding violated constraints can cause other constraints to become redundant in solving any node in the branch tree. These redundancy constraints are eliminated after solving each node by checking whether the slack variable of the constraint is greater than zero. As observed by Laporte et al. (1985), the elimination of these ineffective constraints can result in computational savings, even though occasionally such constraints may need to be reintroduced. Our experiments also showed that purging of these ineffective constraints saved computational time, especially for the problems with more customers. For example, for problems with 15 customers, eliminating ineffective constraints saves on average 12% computational time over the case without eliminating ineffective constraints.

4. Computational Results

We implemented our algorithm using C++ on a SUN Fire 4800 System (12 900MHZ CPUs). The LP solver is CPLEX 4.0. This section presents the computational results from our branch and cut algorithm.

As opposed to the vehicle routing problem, there are few clearly defined benchmark problems for the pickup and delivery problem. We test our approach based on two data sets. The first data set is the modification of Solomon's benchmark problems for the vehicle routing problem. The second data set is generated using the same method of Savelsbergh and Sol (1998). In all our experiments, we assume that the distance between two points is the Euclidean distance. The travel cost and travel time between two points are equal to the distance between these points. All numbers are rounded to the nearest integers.

4.1. Solomon's benchmark problem

We next compare our algorithm on Solomon's vehicle routing test problems. From Solomon's benchmark problems, we studied two different categories of problems: R1 and C1. We skip the R2 and C2 type problems because they are the same as R1 and C1 respectively when the time window is not considered. In each category, two cases are studied. They are R101, R102, C101 and C102. The R-type problems have all locations randomly generated while the C-type problems have locations that are clustered.

Since Solomon's data sets are for the VRP, we randomly paired the requests to form the customers. For each data set, ten different random pairings were generated. For the C-type problems, three sampling methods were tested. They were:

- Ensuring pickup and delivery locations of a customer were in the same cluster
- Ensuring pickup and delivery locations of a customer were not in the same cluster
- No restriction was made on the pairing between the pickup and delivery locations

The fleet size was set to five with the vehicle capacity set to 200. The fixed cost of using a vehicle was set to zero to encourage multiple vehicle usage. Note that if the location of the depots for all vehicles are identical and there are no time window or capacity constraints, there exists an optimal solution that uses only one vehicle for the MVPDP. Therefore, in our experiments each vehicle has its own unique departure/return depot, and it was randomly selected from the data set.

The following Tables 4 to 8 show the average results of the ten samples of each case. In each table, the first column shows the number of customers. The average linear programming bound at the root (Z_{LP}) is shown in the second column. The third and fourth columns show the average value of the optimal solution (Z_{OPT}) and the total CPU time of solving the whole problem. The average number of used vehicles is displayed in the fifth column. The sixth column shows the average integrality gap: Z_{OPT}-Z_{LP}.

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	195.1	214.8	33.3	1.1	19.74
10	268.1	289.1	258.9	1.7	21.03
12	321.0	342.3	428.2	1.5	21.34
15	351.3	374.1	1325.1	1.7	22.79
17	377.3	401.5	10425.4	2.3	24.2

Table 4 Results of MVPDP using Problem Set R101

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	206.57	225.2	28.7	1.1	18.63
10	271.74	291.8	228.0	1.7	20.06
12	319.66	340.7	395.6	1.9	21.04
15	354.63	379.3	1384.3	2.1	24.67
17	377.90	403.2	10583.2	2.4	25.30

Table 5 Results of MVPDP using Problem Set R102

As expected, there is not much of a difference in the solving time between R101 and R102 problems (Tables 4 and 5), and the number of used vehicles increases as the problem size increases. However, the average gap between the LP solution and the optimal integer solution doesn't increase significantly as the problem size increases. We note that we were able to find the optimal solution in all ten samples for each problem size listed in the table within the stopping criterion of three CPU hours.

Tables 6 and 7 show the results of solving Solomon's C type benchmark problems when there is no restriction placed on the pairing between the pickup and delivery location. Recall, for these data sets the demand locations are clustered.

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	152.66	161.0	221.2	1.3	8.34
10	169.51	184.2	1466.0	1.9	14.69
12	174.77	191.5	4728.3	2.6	16.73

Table 6 Results of MVPDP using Problem Set C101

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	151.7	159.8	197.6	1.3	8.13
10	170.1	183.9	1448.9	1.7	13.75
12	175.1	192.1	4821.0	2.5	17.05

Table 7 Results of MVPDP using Problem Set C102

Comparing Tables 4 and 5 with Tables 6 and 7, we find that our algorithm on average can solve the R-type problem faster than the same size C-type problem. The average gap between the LP-relaxation and the optimal solution in the C-type problem is less than that for the corresponding R-type problem, which may be due to the fact that the optimal integer solution for the C-type problem is smaller.

To further understand the relationship between the solving time and the data sets, we ran additional experiments in which we did not make a complete random pairing between the pickup and delivery locations of the same request. In one set of experiments, the pickup and delivery locations for each customer are paired exclusively in the same cluster, and in the other set of experiments they are restricted to be in different clusters. The computational results are shown in the Tables 8 and 9.

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	78.58	79.3	1.1	1.7	0.72
10	82.82	84.0	6.0	1.9	1.18
12	87.34	89.2	16.8	2.7	1.86
15	99.88	102.0	63.4	2.8	2.12
17	138.03	140.6	245.0	3.6	2.57
20	141.4	144.5	737.5	3.8	3.10
22	162.19	165.6	1598.0	4.8	3.41
25	164.84	169.4	3626.0	4.8	4.56

Table 8 Results of MVPDP using Problem Set C101 When the Pickup and Delivery Locations for the Same Customers are in the Same Cluster

Number of Customers	Average Z_{LP}	Average Z_{OPT}	CPU (Sec.)	Average Used Vehicles	Average Gap
7	157.45	165.3	234.5	1.2	7.85
10	176.16	198.5	2615.7	1.6	22.34
12	196.57	224.5	9732.5	1.9	27.93

Table 9 Results of MVPDP using Problem Set C101 When the Pickup and Delivery Locations for the Same Customers are in Different Clusters

Comparing Tables 8 and 9, problems in which the pickup and delivery locations are in different clusters are harder to solve than problems in which the pickup and delivery locations are in the same cluster by using our algorithm. Comparing Tables 8 and 9 with the results in Table 6, it is clear that problems in Table 8 are the easiest category of problems for our formulation. They can even be solved faster than the same size R-type problem. Problems in Table 9 are the hardest set of problems to solve for our formulation. Since problems in Table 6 can be viewed as random combinations of those in Tables 8 and 9, they can be solved faster than problems in Table 9 but slower than problems in Table 8.

4.2. Test Problems for m-PDPHTW

Our formulation can be easily extended to deal with the hard time window problem by including constraints (17) to (20). The purpose of this analysis is to study the sensitivity of the gap between the linear and integer optimal solution as the capacity and time window constraints are relaxed. We note that the gap will increase when the time window increases regardless of the solution technique applied to the problem. The purpose of this analysis is to study the sensitivity of this gap as the constraints are relaxed.

We attempted to solve the problem instances used by Dumas, Desrosiers, and Soumis (1991), but unfortunately they are no longer available. Thus, we used a similar method to that of Savelsbergh and Sol (1998) to randomly generate test instances for the hard time window case. Their method can be described as follows.

One hundred points are randomly generated within a 200×200 square. All pickup and delivery locations are randomly selected from these 100 points. The load for each customer is randomly selected from an interval [q^{min}, q^{max}]. The capacity for each vehicle is set to Q. The time window for each vehicle is [0, 600]. Let t_i be the travel time from the pickup location i^+ of customer i to the delivery location i^- of customer i. Randomly select a number u_i from the interval [0, 600-t_i], then define the time window for i^+ as [u_i , u_i+60] and time window for i^- as [u_i+t_i , u_i+t_i+60]. If there are n customers, the fleet will have n available vehicles. The departure and return depots are the same for the same vehicle. The vehicle depots are also selected from those 100 points, but on the condition that the number j vehicle could serve customer j within its requested time windows. Therefore, there exists at least one feasible solution that has each vehicle servicing only one customer. Table 10 lists the problem classes with the parameters.

Class	|N|	|M|	q^{min}	q^{max}	Q	w
A15	15	15	5	15	15	60
B15	15	15	5	20	20	60
C15	15	15	5	15	15	120
D15	15	15	5	20	20	120
E15	15	15	5	15	∞	∞

Table 10 Problem Classes

Before solving the problem, we use the same steps as Dumas, Desrosiers, and Soumis (1991) to determine the admissible arcs in the hard time window and vehicle capacity conditions. In Table 11 we show the computational results of solving the above five problem classes. The vehicle cost was set to 10000 except for the E class problem because without the restrictive capacity and time windows, the optimal solution will only include a single vehicle if the fixed vehicle cost is set to a high number for the E class problem. Z_{OPT} is the optimal solution and Z_{LP} is the linear programming bound at the root of the branch-and-bound tree.

Problem	Z_{OPT}	Z_{LP}	Gap	IP CPU Time (Sec.)		Problem	Z_{OPT}	Z_{LP}	Gap	IP CPU Time (Sec.)
A15.1	61036	61036.0	0	4.3		B15.1	51216	51212.6	0.28	15.3
A15.2	51444	51442.3	0.11	18.1		B15.2	51379	51373.6	0.39	12.7
A15.3	41101	41032.1	6.26	11.7		B15.3	61145	61145.0	0	4.3
A15.4	51222	51180.4	3.40	262.3		B15.4	71263	71220.8	3.34	74.5
A15.5	61319	61316.8	0.17	13.2		B15.5	41207	41207.0	0	3.7
A15.6	51384	51372.1	0.86	91.2		B15.6	61252	61252.0	0	3.0
A15.7	41186	41186.0	0	2.9		B15.7	51063	51035.9	2.55	175.7
A15.8	91152	91152.0	0	4.0		B15.8	71028	71028.0	0	5.4
A15.9	61092	61092.0	0	3.5		B15.9	41493	41455.9	2.48	138.4
A15.10	61264	61254.1	0.78	19.8		B15.10	91125	91125.0	0	2.3
C15.1	50962	50937.4	2.56	436		D15.1	41024	40974.0	4.87	1050
C15.2	31056	31040.4	1.48	279		D15.2	30948	30933.7	1.51	194
C15.3	50984	50975.0	0.89	65		D15.3	50794	50787.3	0.84	106
C15.4	30946	30926.9	2.01	596		D15.4	30869	30863.9	0.59	87
C15.5	31009	30989.4	1.94	493		D15.5	50967	50957.0	1.02	71
C15.6	40891	40874.3	1.87	480		D15.6	21019	20957.8	6.01	1878
C15.7	50930	50905.0	2.70	1078		D15.7	40895	40881.1	1.55	409
C15.8	41017	41002.7	1.41	103		D15.8	30914	30891.9	2.42	461
C15.9	30887	30862.8	2.73	574		D15.9	50936	50905.6	3.25	877
C15.10	50893	50867.4	2.87	645		D15.10	41020	41009.3	1.05	172
E15.1	50819	50803.5	1.89	208						
E15.2	30845	30816.0	3.20	2064						
E15.3	40651	40617.5	5.15	1213						
E15.4	30848	30809.2	4.58	6142						
E15.5	30821	30819.0	0.24	22						
E15.6	20858	20801.7	6.67	17359						
E15.7	10847	10807.5	6.69	2983						
E15.8	50731	50706.3	3.38	775						
E15.9	40858	40812.0	5.36	3238						
E15.10	20681	20633.8	6.93	5243						

Table 11 Computational Result for Five Problem Classes

We remark that in their experiments Savelsbergh and Sol (1998) do not consider the E class of problems since their set of experiments focused on tight time window and capacity constraints using a Column Generation Method to solve the problem sets. They also run problem sets consisting of 30 customers as opposed to our 15 since it is computationally prohibitive to optimally solve the larger problem instances for the E class of problems.

Using the same definition of Savelsbergh and Sol (1998), the integrality gap is calculated by (Z_{OPT} - Z_{LP})/(Z_{OPT} – Total cost of using vehicles). Table 11 shows that the gap of our formulation does not deteriorate significantly when the time window and capacity constraints are relaxed. For example, the average gap for the five classes of problems are: A class 1.158; B class 0.896; C class 2.046; D class 2.311; E class 4.409. We note that for the problem sets with tight capacity and time window constraints (classes A, B, C, and D) our gap is higher than those reported by Savelsbergh and Sol (1998). This result is to be expected since the Column Generation Method is a very effective technique to optimally solve the problem when a significant number of arcs can be eliminated. Our proposed approach is primarily intended for problems from class E.

5. Conclusions

This paper presents a new 0-1 linear programming formulation for the multiple vehicle pickup and delivery problem. Our solution approach is based on a branch-and-cut algorithm. By using the proposed solution approach, we were able to optimally solve problem instances of up to 5 vehicles and 17 customers on problems without clusters and up to 5 vehicles and 25 customers on problems with clusters within a stopping criterion of three CPU hours on a SUN Fire 4800 System.

Acknowledgements

The research reported in this paper was partially supported by the National Science Foundation under grant DMI-9732878.

References

G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis, "The VRP with Pickup and Delivery," in *The Vehicle Routing Problems*, P. Toth and D. Vigo (eds), 225-242, SIAM, Philadelphia, 2001.

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 Science* 6**,** 301-325 (1986).

Y. Dumas, J. Desrosiers, and F. Soumis, "The Pickup and Delivery Problem with Time Windows," *European Journal of Operational Research* 54**,** 7-22 (1991).

M. Fischetti and P. Toth, "An Additive Bounding Procedure for Combinatorial Optimization Problems," *Operations Research* 37**,** 319-328 (1989).

I. Ioachim, J. Desrosiers, Y. Dumas, M. M. Solomon, and D. Villeneuve, "A Request Clustering Algorithm for Door-to-door Handicapped 'Transportation'," *Transportation Science* 29, 63-78 (1995).

B. Kalantari, A. V. Hill, and S. R. Arora, “An Algorithm for the Traveling Salesman Problem with Pickup and Delivery Customers,” *European Journal of Operational Research* 22, 377-386 (1985).

G. Laporte, Y. Norbert, and M. Desrochers, “Optimal Routing under Capacity and Distance Restrictions,” *Operations Research* 33**,** 1050-1073 (1985).

O. B. G. Madsen, H. F. Ravn, and J. M. Rygaard, “A Heuristic Algorithm for a Dial-a-ride Problem with Time Windows, Multiple Capacities, and Multiple Objectives,” *Annals of Operations Research* 60, 193-208 (1996).

M. Padberg and G. Rinaldi, “An Efficient Algorithm for the Minimum Capacity Cut Problem,” *Mathematical Programming* 47, 19-36 (1990).

M. Padberg and G. Rinaldi, “A Branch-and-cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems,” *SIAM Review* 33, 60-100 (1991).

H. Psaraftis, “A Dynamic Programming Solution to the Single Vehicle Many-to-many Immediate Request Dial-a-ride Problem,” *Transportation Science* 14**,** 130-154 (1980).

H. Psaraftis, “An Exact Algorithm for the Single Vehicle Many-to-many Dial-a-ride Problem with Time Windows,” *Transportation Science* 17**,** 351-357 (1983).

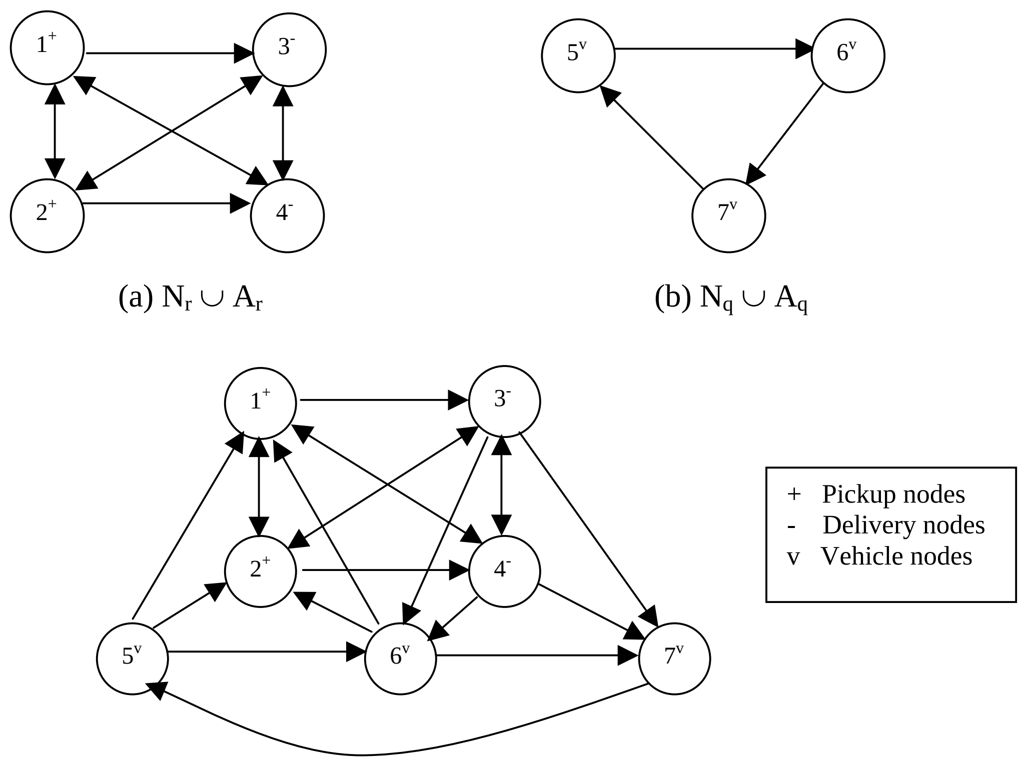
K. S. Ruland and E. Y. Rodin, “The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm,” *Computers & Mathematics with Applications* 33(12), 1-13 (1997).

M. W. P. Savelsbergh and M. Sol, “The General Pickup and Delivery Problem,” *Transportation Science* 29**,** 17-29 (1995).

M. W. P. Savelsbergh and M. Sol, “Drive: Dynamic Routing of Independent Vehicles,” *Operations Research* 46, 474-490 (1998).

T. R. Sexton and Y. Choi, “Pickup and Delivery of Partial Loads with Soft Time Window,” *American Journal of Mathematical and Management Science* 6**,** 369-398 (1986).

P. Toth and D. Vigo, “Heuristic Algorithms for the Handicapped Persons Transportation Problem,” *Transportation Science* 31, 60-71 (1997).

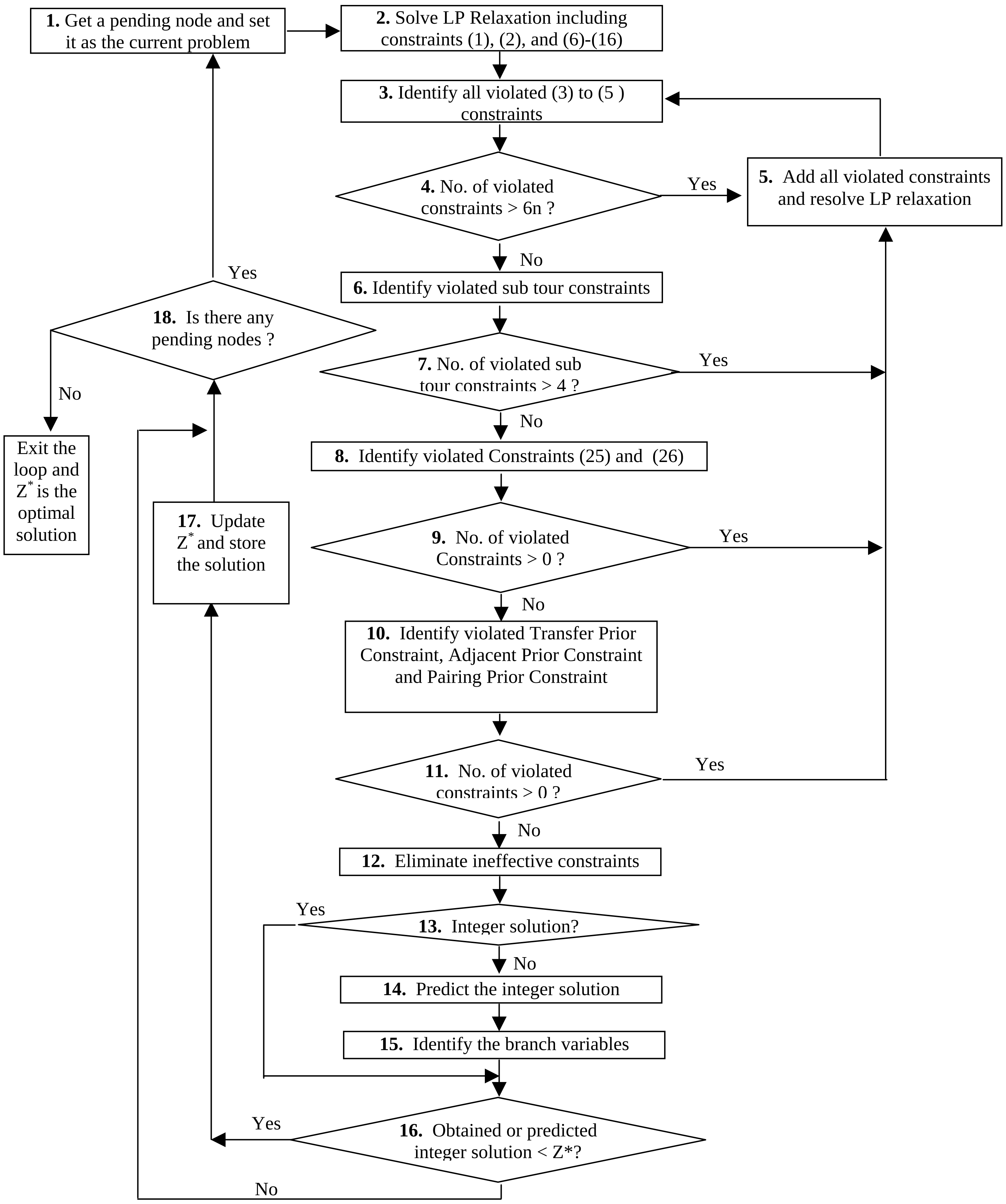


(a) $N_r \cup A_r$

(b) $N_q \cup A_q$

(c) $N_r \cup N_q \cup A_q \cup A_r \cup A_{r,q}$

Figure 1 Network for a 2-customer and 2-vehicle Case



1. Get a pending node and set it as the current problem
2. Solve LP Relaxation including constraints (1), (2), and (6)-(16)
3. Identify all violated (3) to (5) constraints
4. No. of violated constraints > 6n ?
Yes
5. Add all violated constraints and resolve LP relaxation
No
6. Identify violated sub tour constraints
7. No. of violated sub tour constraints > 4 ?
Yes
No
8. Identify violated Constraints (25) and (26)
9. No. of violated Constraints > 0 ?
Yes
No
10. Identify violated Transfer Prior Constraint, Adjacent Prior Constraint and Pairing Prior Constraint
11. No. of violated constraints > 0 ?
Yes
No
12. Eliminate ineffective constraints
13. Integer solution?
Yes
No
14. Predict the integer solution
15. Identify the branch variables
16. Obtained or predicted integer solution < Z*?
Yes
No
17. Update Z* and store the solution
18. Is there any pending nodes ?
Yes
No
Exit the loop and Z* is the optimal solution

Figure 2 Flow Chart of the Entire Algorithm

List of Tables And Figures