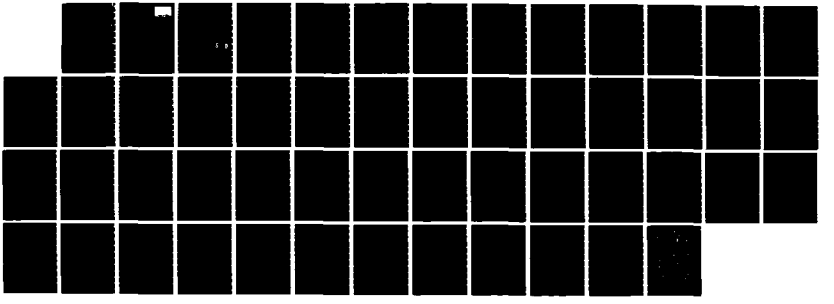
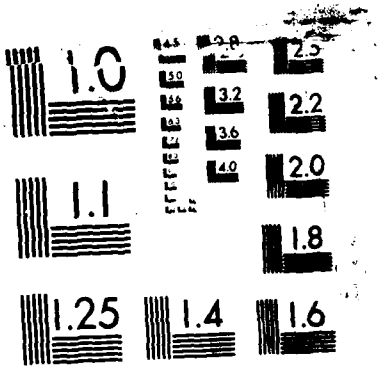
AD-A168 175 PICK-UP AND DELIVERY PROBLEM: MODELS AND SINGLE VEHICLE EXACT PROCEDURES..(U) GEORGIA INST OF TECH ATLANTA PRODUCTION AND DISTRIBUTION RESE.. J J JARVIS ET AL. 1/1

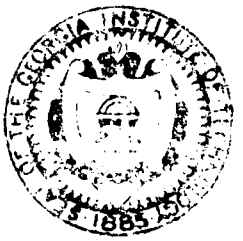
UNCLASSIFIED JUN 85 PRDC-84-12 N00014-83-K-0147 F/G 13/6 NL





MICROCOPY RESOLUTION TEST CHART

Georgia Institute of Technology



PRODUCTION AND
DISTRIBUTION RESEARCH
CENTER

PDRC Report Series 84-12
June 1985

(12)

PICK-UP AND DELIVERY PROBLEM: MODELS AND SINGLE VEHICLE EXACT PROCEDURES

by

John J. Jarvis
Omer Kirca

PDRC 84-12

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

This work was supported by the Office of Naval Research under Contract No. N00014-83-K-0147. Reproduction is permitted in whole or in part for any purpose of the U. S. Government.

ABSTRACT

The pick-up and delivery problem (PDP) is a special case of the vehicle routing problem and can be defined as follows. There exists N tasks to be performed by a fleet of vehicles. Each task consists of picking-up a certain amount of goods from its supply node and transporting these to its associated demand node. The problem is to perform these N tasks so that a certain objective is optimized. The key characteristics of the PDP are the precedence relations in visiting the supply-demand nodes and fluctuations in vehicle capacity usage during the course of the tour.

Two linear binary models which incorporate the special PDP characteristics are presented. The first model is for the single vehicle problem. This model is extended to the multiple vehicle case. The relaxation of certain constraints in the single vehicle PDP model yields lower bounds for the optimal solution. These lower bounding problems are solved by dynamic programming.

The lower bounds obtained are implemented in a branch and bound procedure. The algorithm is able to solve an eight task, single vehicle problem optimally. By dualizing one of the PDP constraints and solving the resulting program by a Lagrangian ascent procedure, better quality lower bounds are obtained. This permits twelve task, single vehicle problems to be solved optimally.

Key words: Vehicle Routing, Pick-up and Delivery Problem, Dynamic Programing, Lagrangian Relaxation, Branch and Bound

1. INTRODUCTION

Vehicle routing and scheduling problems have attracted considerable attention among researchers. Vehicle routing and scheduling is a term given to a wide class of problems. Each problem, involves routing and scheduling of vehicles operating in different environments and performing different tasks. Bodin and Golden [1] attempt to classify vehicle routing and scheduling problems into different groups. Each group concentrates on one aspect of a particular problem. One group of problems is identified as "pick-up and delivery problems," (PDP). In this paper pick-up and delivery problems are analyzed and solution strategies and procedures for the single vehicle case are proposed.

1.1 Pick-up and Delivery Problems

PDP can be described as follows:

There exist N tasks to be performed with a fleet of V vehicles. Each task, r = 1,...,N, consists of picking-up q_r units from a source node a(r) and transporting them to the corresponding demand node b(r). Each task can be specified by three parameters: $(a(r),b(r),q_r)$. The problem is to perform these N tasks with the available vehicle fleet so that a certain objective is optimized.

PDP can be classified with respect to different schemes. Bodin and Golden suggest several classifications including:

A. Fleet Size:

   1. Single vehicle

   2. Multiple vehicle

B. Operation Type:

1. One-to-Many-to-One

2. Many-to-Many

3. Any combination of these

C. Type of Fleet:

1. Homogeneous case (all vehicles are identical)

2. Heterogeneous case (not all vehicles are identical)

D. Objective:

1. Minimize routing costs

2. Minimize sum of fixed and variable costs

3. Minimize fleet size

E. Time Constraints:

1. On routes

2. On tasks

3. Mixed

F. Other (problem dependent) constraints:

1. On vehicle loading/unloading

2. On tasks

3. On operations

The nature of tasks and operation types are the two main characteristics of PDPs which differentiate them from the general vehicle routing problems (VRPs) described by Christofides [14]. In PDPs, both pick-ups and deliveries are involved where in VRPs only pick-ups or only deliveries are performed. Because of this added complexity of PDPs, the solution strategies and procedures which are effective for VRPs cannot be used directly. The main reasons are:

(1) In VRPs, once tasks are assigned to vehicles, the route for each vehicle can be determined by the solution of a Travelling Salesman

Problem (TSP). In PDPs, even if tasks can be asigned to the vehicles, the routing of a particular vehicle is a harder problem than a TSP, due to the precedence relations of the tasks.

(2) In VRPs, the capacity usage of the vehicle is determined at the intial node of its route and the capacity usage declines steadily (in the delivery case) throughout the route. In the case of PDPs, it is not possible to determine the capacity usage in advance since it fluctuates as deliveries and pick-ups are performed.

(3) Operation type is another factor which differentiate VRPs from the PDPs. In VRPs, operations can be identified as one-to-many for single domicile problems and few-to-many for the multiple domicile problems, where a domicile is the node at which goods are initially stored (in delivery case). In both of these problems the goods to be transported can be considered as homogeneous in the sense that they are supplied from specific domiciles. In the case of PDPs the goods can not be considereed as homogeneous, because each demand point of a task can only be supplied from its respective supply point. This non-homogenity of the goods creates a multi-product environment which further complicates the problem.

In this paper "many-to-many" PDPs will be considered. "many-to-many" means that the pick-up and delivery points of each task are all distinct points. In one(few)-to-many-to-one(few) PDPs, vehicles pick-up goods from the domiciles and delivery them to "many" demand points and then pick-up some goods from these demand points and deliver them to the domiciles. As far as the operations are concerned "many-to-many" problems can be considered as a generalization of the

"one-to-many-to-one" problems. The solution procedures for many-to-many problems can easily be adpated to the later problem.

Pick-up and delivery problems are among the most frequently faced problems in transportation and distribution. In practice, these problems arise in different transportation fields, such as naval, air, rail, and land transportation. Also, these problems occur in different environments such as urban transportation, interstate, and intercontinental transportation. Some reported applications of the PDPs are:

a) Tractor trailer routing and scheduling; [5], [7], [11]

b) Navy Supply Center routing: [22], [29]

c) Static dial-a-ride: [10], [12], [27]

d) School bus routing and scheduling: [2], [20], [24], [28]

e) Air Force logistics: [1]

f) Tanker-vessel scheduling: [3], [9], [19].

1.2 Survey of Solution Methods for the Pick-up and Delivery Problems

Unlike VRPs, there exists little or no general literature concentrating on PDPs. Each of the studies involves a particular PDP and proposes solution procedures which makes use of the special characteristics of the particular problem addressed. Also, there currently exists no exact procedures(s) which solves these problems. According to Psaraftis [25], even moderate size static dial-a-ride problems, ($N < 10$) cannot be solved exactly with a reasonable computational effort. Most of the solution methods are inexact procedures.

1.2.1 Full Load PDP

The PDPs, where $q_r = C = 1$ for all tasks, are called the full load PDPs. In the absence of time restrictions, the full load pick-up and

delivery problem is formulated by Bodin et al in two different equally effective ways. One approach is based on the travelling salesman formulation, where the vehicle is restricted to visit the delivery node of a particular task immediately after visiting its associated pick-up node. The same problem can also be formulated as a chinese postman problem (CPP), where the arcs to be visited are identified as the arcs between the pick-up and delivery points of each task. Depending on the formulation any TSP or CCP procedure can be applied to the problem.

For time constraints, Ball et al [7] propose two different heuristics. The first is a two phase method where the first phase consists of using either a TSP or a CPP procedure to create a large route. In the second phase, this large route is partitioned into smaller feasible routes which can be handled by the available vehicles. In the second method, a greedy insertion routine is used in which origin/destination pairs are added to the vehicle routes iteratively.

1.2.2 Partial Load PDP

PDPs which have task loads less than the vehicle capacities are called partial load problems. The partial load problem can also solve full load or static dial-a-ride problems. School bus routing, airplane routing, and ship scheduling all fall under partial load problems. All procedures available in the literature attempt to solve this problem heuristically.

Ali et al [1] generate "potentially good feasible" routes and, with a mixed integer program, try to select a set of best routes satisfying routing and scheduling requirements. The authors apply the procedure to a 60 base, 61 task problem of the U.S. Air Force. As they point out, the

selection of "potentially good" routes is the crucial step. As the ability to identify "good" routes increases, the quality of the resulting solution also improves.

The second approach for these type of problems are the TSP based heuristics. Inexact TSP procedures are modified with respect to the special characteristics of each problem.

Assad et al [5] propose a two phase approach to a partial load PDP. Phase one contains the routing component and the second phase involves the scheduling of the routes found in the first phase. The routing phase involves an insertion algorithm where origin and destination nodes of a task are inserted into one of the existing vehicle routes. At the end of the routing phase the vehicle routes are feasible with respect to:

1) precedence relations for each task,

2) vehicle capacities, and

3) maximum route lengths.

The scheduling phase scans each vehicle route from start to finish and attaches times to each of the operations involved. This phase minimizes the service times of each vehicle. If any infeasibilities are found with respect to time constraints they are ignored.

Whinchel et al [29] propose a single phase method for the scheduling of palletized cargo delivery among the warehouses in a Navy Supply Center. In their approach, vehicle routes are created, one-at-a-time, by attaching a task to the end of the route under construction. The criterion used is the savings concept of Clark and Wright [18]. For the same problem, an insertion routine, which examines all possible insertions, was developed by Jarvis and Kirca [22]. This procedure resulted in improved solutions of the problem.

1.2.3 Static Dial-a-Ride

The static dial-a-ride problem is one of the most widely studied pick-up and delivery problems. Procedures for the dial-a-ride problem will be examined under two parts: single vehicle and multiple vehicle cases.

1.2.3.1 Single Vehicle Dial-a-Ride Problem

When there are no time windows on the tasks, Psaraftis [25] proposes an optimal procedure. He presents a mathematical program for this problem and solves it by dynamic programming. The major disadvantage of his approach is the "curse of dimensionality". There exist N state variables in each of the 2N stages in his formulation. The required computer memory is on the order of 3^N. Due to this high memory requirements his approach fails for most applications. Computational experience with his method did not extend beyond problems having nine tasks. Even eight and nine task (customer) problems required 149 and 591 CPU seconds respectively in finding the optimal solutons.

Psaraftis [26] also develops two heuristic proedures for the same problem. In the firt heuristic, an approximate TSP tour is constructed using Christofides' [13] procedure. Then, employing improvement routines, the initial TSP tour is forced to be feasible with respect to both precedence and capacity constraints. The second heuristic uses the minimal spanning tree portion of Christofides' algorithm. The initial TSP tour is formed and then this tour is improved. Psaraftis also shows that his first algorithm has a worst case bound of 3.0.

Armstrong [4] proposes a dynamic programming based exact algorithm for the problem. Although in his work the general PDP is discussed, all

the procedures developed and tested are for the static dial-a-ride problem. The exact algorithm was able to solve highly restricted 15 customer (task) dial-a-ride problems. Armstrong also proposed several heuristic procedures which utilize the Clarke-Wright savings technique, greedy approach, and an insertion routine. he attempted to improve the tours found by each of the above procedures by a 3-opt improvement routine. Computational results demonstrate that the procedures which selects the best of five 3-optimal random tours perform better than the other procedures considered in his work.

For the case involving time windows, Sexton and Bodin [27] and Baker [6] propose two procedures. Sexton and Bodin [35] try to minimize the total inconvenience which a customer may experience due to excess ride time and deviation from the desired delivery time. They formulate a mixed integer program and apply Benders' partitioning method to this formulation. However the partitioned program is not solved exactly. Instead they make use of the master-subprogram mechanism of this partitioning in order to find a heuristic solution. The master program is a routing scheme where the order of pick-ups and deliveries are decided. In the scheduling phase, the exact times of pick-ups and deliveries are specified. Then, with respect to the schedules, the objective function coefficients of the routing problem are adjusted. This procedure alternates between the routing and scheduling problems until no improving solutions can be found.

Baker [6] proposes another method which minimizes the total servicing time of the vehicle. he formlates a nonlinear program and atempts to solve this program by several procedures.

1.2.3.2 Multiple Vehicle Dial-a-Ride Problem

There do not exist effective exact solution procedures for these type of problems. Cullen, Jarvis and Ratliff [18] and Bodin and Sexton [10] present two heuristic procedures.

Cullen et al propose an interactive optimization scheme to develop "good" vehicle routes. The promising alternatives are suggestd through the use of shadow prices for a set partitioning formulation of the problem.

Bodin and Sexton propose a three step approach. In the first step customers are assigned to vehicles. The second step attempts to improve this assignment. In the final step each vehicle route is developed. As the authors point out, finding an initial set of "good" clusters is still an unsolved problem. They propose two alternative methods for the initial assignment of tasks to the vehicles. The authors applied the algorithm to the problem of a subscriber dial-a- ride system and were able to find improved solutions compared to those of manual scheduling.

2. MODELS FOR THE PICK-UP AND DELIVERY PROBLEM

Christofides discusses mathematical modeling approaches for vehicle routing problems. Although his discussion is for the general VRP, most of his comments about the modeling approaches are valid for PDPs. He identifies two schemes for the routing problem, namely, set partitioning based models and TSP based models. Further, he discusses some advantages and disadvantages of each scheme.

One disadvantage of a set partitioning model is that for most practical problems it is impossible to generate and store all the alternative feasible routes. On the other hand, a TSP based model is not able to express most realistic constraints explicitly.

2.1 Single Vehicle Pick-up and Delivery Model

Consider a pick-up and delivery problem where there exist N tasks and a vehicle with a capacity of C units. The following characteristics will be assumed for the operation of the system:

1) The vehicle is initially stationed at the base located at node 1.
2) The route starts and ends at the base.
3) No task splitting is allowed. That is, if a supply node is visited the vehicle picks-up the entire load and delivers it to the associated demand node on its first visit.
4) All supply and demand nodes are distinct in the sense that each supply node supplies only one demand node and each demand node obtains it's requirements from only one supply node. (This results in no loss of generality.)

5) The objective is to service all tasks with a minimum route length.

Let

N = number of tasks to be serviced

M = total number of nodes. (M = 2N if the base node is also a supply node, or M = 2N + 1 otherwise)

S = set of supply nodes

D = set of demand nodes

a(r) = supply node of task r

b(r) = demand node of task r

q_r = units of load for task r

C = capacity of the vehicle

d_{ij} = distance between node i and node j

In the classical TSP models, based on the assignment of arcs, the decision variable determines which arcs are in the tour and which arcs are not visited by the tour. In such a formulation the sequence of node visits are not readily available, and the model does not keep track of the sequence of nodes visited. In the case of PDPs, in order to enforce the precedence and capacity constraints, it is necessary to keep track of the sequence of node visits as well as the arcs traversed. The model must track the sequence of the nodes visited so that a visit to the demand node of a particular task can be restricted to a position higher in the sequence than its associated supply node. Also, the model should be able to keep track of the load on the vehicle so that the vehicle capacity constraint can be enforced.

If there are N tasks to be performed, the vehicle has to make M = 2N + 1 stops, i.e. one stop for each pick-up or delivery node plus one for

the base node. The problem is to determine the nodes visited at each stop, so that all tasks are performed with a minimum tour length. The problem could be formulated as an M-stage(stop) problem, where the node visited at each stage is identified.

The model presented here considers the routing problem as a stage-by-stage process. In each stage the vehicle moves from one node to another. Since there are M nodes to be visited, the model makes M decisions, one in each stage. Let

$$x_{ij}^{t} = \begin{cases} 1 & \text{if the vehicle moves from node i to j at stage t} \\ 0 & \text{otherwise} \end{cases}$$

Then the single vehicle pick-up and delivery model becomes

$$\text{Min} \quad \sum_{t=1}^{M} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ij} x_{ij}^{t} \tag{2.1}$$

$$\text{s.t.:} \quad \sum_{j=2}^{M} x_{1j}^{1} = 1 \tag{2.2}$$

$$\sum_{j=2}^{M} x_{j1}^{M} = 1 \tag{2.3}$$

$$\sum_{i=1}^{M} \sum_{j=1}^{M} x_{ij}^{t} = 1 \qquad t = 2,\ldots,M \tag{2.4}$$

$$\sum_{t=1}^{M} \sum_{j=1}^{M} x_{ij}^{t} = 1 \qquad i = 2,\ldots,M \tag{2.5}$$

$$\sum_{j=1}^{M} x_{ij}^{t} - \sum_{j=1}^{M} x_{ji}^{t-1} = 0 \qquad t = 2,\ldots,M;\ i = 2,\ldots,M \qquad (2.6)$$

$$\sum_{r=1}^{N} \sum_{\ell=1}^{t} \sum_{j=1}^{M} q_r (x_{a(r)j}^{\ell} - x_{b(r)j}^{\ell}) \leq C \qquad t = 2,\ldots,M \qquad (2.7)$$

$$\sum_{t=1}^{M} \sum_{j=1}^{M} t(x_{b(r)j}^{t} - x_{a(r)j}^{t}) \geq 1 \qquad r = 1,\ldots,N \qquad (2.8)$$

$$x_{ij}^{t} = 0 \text{ or } 1 \qquad i,j,t = 1,\ldots,M \qquad (2.9)$$

Constraints (2.2) and (2.3) state that the route must start and end at the base node 1. Constraints (2.4) force the vehicle to visit exactly one node at each stage and (2.5) restricts it to visit each node exactly once. Constraints (2.6) are the flow equations for each node at each stage, stating that if the vehicle enters a node at stage (t-1) then it must leave that particular node at stage t. Constraints (2.2) through (2.6), together with the integrality constraint (2.9), define a TSP tour which begins and ends at node 1.

Constraints (2.7) are the vehicle capacity constraints for each stage and constraints (2.8) are the task precedence constraints. These restrict the vehicle to visit a supply node before visiting its associated demand node. The model has M^3 binary variables and M^2+2+4M constraints.

Time window constraints on the tasks can be imposed on the model as follows. Let

$(\underline{e}_i, \bar{e}_i)$ = earliest and latest pick-up or delivery times at node i

z_t = elapsed time at stage t. ($z_1 = 0$)

p_{ij} = time to move from node i to j.

v_i = total loading and unloading time at node i

Then,

$$z_{t-1} + \sum_{j=1}^{M} \sum_{i=1}^{M} (p_{ji} + v_j) x_{ji}^t = z_t \qquad t = 2,\ldots,M \quad (2.10)$$

$$\sum_{i=1}^{M} \underline{e}_i \sum_{j=1}^{M} x_{ij}^t \leq z_t \leq \sum_{i=1}^{M} \bar{e}_i \sum_{j=1}^{M} x_{ij}^t \qquad t = 2,\ldots,M \quad (2.11)$$

Note that (2.10) defines the elapsed time at stage t and (2.11) enforces the time windows for each stage. If node s is visited at stage t, then $\sum_{j=1}^{M} x_{sj}^t = 1$ and $\sum_{j=1}^{M} x_{ij}^t = 0$ for all $i = s$, hence (2.11) becomes:

$$\underline{e}_s \leq z_t \leq \bar{e}_s$$

Constraints (2.10) and (2.11) can be combined into a single equation of the form:

for $t = 2,\ldots,M$

$$\sum_{i=1}^{M} \underline{e}_i \sum_{j=1}^{M} x_{ij}^t \leq \sum_{\ell=1}^{t-1} \sum_{i=1}^{M} \sum_{j=1}^{M} (p_{ij} + v_i) x_{ij}^{\ell} \leq \sum_{i=1}^{M} \bar{e}_i \sum_{j=1}^{M} x_{ij}^t \quad (2.12)$$

2.2 Multiple Vehicle Pick-up and Delivery Model

The multiple vehicle case can be formulated as an extension of the single vehicle model. Using the assumptions of Section 2.1, suppose there are V vehicles available to service N tasks. Let the capacity of vehicle k be C_k respectively. Further let

$$y_{rk} = \begin{cases} 1 & \text{if task r is processed by vehicle k} \\ 0 & \text{otherwise} \end{cases}$$

$$x^t_{ijk} = \begin{cases} 1 & \text{if vehicle k moves from node i to j at stage t} \\ 0 & \text{otherwise} \end{cases}$$

Then the model becomes

$$\text{Min} \quad \sum_{k=1}^{M} \sum_{t=1}^{M} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ij} x^t_{ijk} \tag{2.13}$$

s.t.

$$\sum_{k=1}^{V} y_{rk} = 1 \qquad r = 1,\ldots,N \tag{2.14}$$

$$\sum_{t=1}^{M} \sum_{j=1}^{M} x^t_{a(r)jk} = y_{rk} \qquad r = 2,\ldots,N;\ k = 2,\ldots,V \tag{2.15}$$

$$\sum_{t=1}^{M} \sum_{j=1}^{M} x^t_{b(r)jk} = y_{rk} \qquad r = 1,\ldots,N;\ k = 1,\ldots,V$$

$$\sum_{i=1}^{M} \sum_{j=1}^{M} x^t_{ijk} \leq 1 \qquad t = 2,\ldots,M;\ k = 1,\ldots,V \tag{2.16}$$

$$\sum_{j=1}^{M} x^t_{ijk} - \sum_{j=1}^{M} x^{t-1}_{jik} = 0 \qquad i = 2,\ldots,M;\ t = 2,\ldots,M;\ k = 1,\ldots,V \tag{2.17}$$

$$\sum_{j=1}^{M} x^1_{1jk} = 1 \qquad k = 2,\ldots,V \tag{2.18}$$

$$\sum_{\ell=2}^{t} \sum_{r=1}^{N} \sum_{j=1}^{M} q_r(x^{\ell}_{a(r)jk} - x^{\ell}_{b(r)jk}) \leq C_k \qquad t = 2,\ldots,M \tag{2.19}$$

$$k = 1,\ldots,V$$

$$\sum_{t=1}^{M} \sum_{j=1}^{M} t(x^t_{b(r)jk} - x^t_{a(r)jk}) \geq 1 \qquad r = 1,\ldots,N;\ k = 1,\ldots,V \tag{2.20}$$

$$x^t_{ijk} = 0 \text{ or } 1 \qquad i,j,t = 1,\ldots,M;\ k = 1,\ldots,V \tag{2.21}$$

$$y_{rk} = 0 \text{ or } 1 \qquad r = 1,\ldots,N;\ k = 1,\ldots,V \tag{2.22}$$

In constraints (2.14) each task is assigned to one vehicle. Constraints (2.15) ensure that the supply and demand nodes of each task are visited only by the vehicle to which the task is assigned. The remaining constraints ensure feasible routes for the vehicles similar to the single vehicle model. This model has V times as many binary variables and constraints as the single vehicle model.

Both the single and multiple vehicle models are large binary problems which are not likely to be solved by direct optimization methods. In the next section some alternative procedures, based on the above formulations, will be discussed. These may be helpful in solving some moderate size problems.

III. LOWER BOUNDS FOR THE SINGLE VEHICLE PICK-UP AND DELIVERY PROBLEM

Christofides et al [15,16] propose a class of lower bounding procedures for the TSP and VRPs. They are based on state space relaxations of the dynamic programming formulation of the respective problems. Using these ideas, potentially effective lower bounding schemes can be developed for the formulation given in Section 2.1.

3.1 Lower Bounds Through Minimum M-tours

Consider the single vehicle model of Section 2.1. Suppose the constraints (2.5), (2.7), and (2.8) are relaxed. Then the resulting program is

$$\text{Min} \quad \sum_{t=1}^{M} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ij} x_{ij}^{t} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j=2}^{M} x_{1j}^{1} = 1 \tag{3.2}$$

$$\sum_{j=2}^{M} x_{j1}^{M} = 1 \tag{3.3}$$

$$\sum_{i=1}^{M} \sum_{j=1}^{M} x_{ij}^{t} = 1 \qquad t = 2,\ldots,M \tag{3.4}$$

$$\sum_{j=1}^{M} x_{ij}^{t} - \sum_{j=1}^{M} x_{ji}^{t-1} = 0 \qquad t = 2,\ldots,M;\ i = 2,\ldots,M \tag{3.5}$$

$$x_{ij}^{t} = 0 \text{ or } 1 \qquad i,i,t = 1,\ldots,M \tag{3.6}$$

In the above program, (3.1) through (3.6) define a minimum M-tour which visits M nodes and begins and ends at node 1. Since constraints (2.5) are relaxed, the tour may contain subtours in the sense that a node may be visited more than once. Further, the tour may not be feasible with respect to the capacity and precedence constraints of (2.7) and (2.8) respectively. As Houck et al [21] show, the value of the minimum M-tour is a lower bound for the TSP and also for the single vehicle PDP.

The minimum M-tour problem is a shortest path problem which can be solved efficiently by dynamic programming. In solving the problem, 2-node subtours, which are of the form i-j-i, can be prohibited by the procedure suggested by Christofides [14].

3.2 Lower Bounds Through Minimum QM-tours

Let Q_T be the total number of units of load that should be serviced by the vehicle, i.e.

$$Q_T = \sum_{r=1}^{N} q_r \tag{3.7}$$

We can add the following additional constraint to the M-tour model of Section 3.1.1 (which was redundant for the single vehicle model).

$$\sum_{t=1}^{M} \sum_{r=1}^{N} \sum_{j=1}^{M} q_r x^t_{a(r)j} = Q_T \tag{3.8}$$

Constraint (3.8) forces the minimum M-tour to visit a combination of supply nodes such that a total of Q_T units of load are picked-up. Such an M-tour will be called a QM-tour and the value of this tour is a valid

lower bound on the single vehicle model. The minimum QM-tour problem can be solved by dynamic programming as follows. Let

$f_i^t(Q)$ = length of the shortest path from node 1 to i which visits t nodes and picks up a total load of Q.

Then for $i = 2,\ldots,M$; $t = 3,\ldots,M$; and $Q = 0,\ldots,Q_T$

$$f_i^t(Q) = \underset{j \neq i}{\text{Minimum}} \begin{cases} f_j^{t-1}(Q-q_i) + d_{ji} & \text{if } i \varepsilon S \\ f_j^{t-1}(Q) + d_{ji} & \text{if } i \varepsilon D \end{cases}$$

where,

$$f_i^2(Q) = \begin{cases} d_{1i} & \text{if } Q = q_i \text{ and } i \varepsilon S \\ \infty & \text{otherwise} \end{cases}$$

Then the length of the minimum QM-tour can be computed as follows.

$$LB2 = \underset{i \alpha D}{\text{Minimum}} \{f_i^M(Q_T) + d_{i1}\}$$

3.3 Lower Bounds Through Minimum CM-tours

The capacity constraints (2.19) of the single vehicle model can be imposed on the M-tours such that the vehicle capacity would not be violated at any stage. The minimum M-tour which does not violate the capacity constraints will be called a CM-tour and the dynamic programming recursive equations can be written as follows. Let

$f^t_i(U)$ = length of the shortest path from node 1 to i visiting t nodes and having a total load of U units at node i.

Then for all i = 2,...,M; t = 3,...,M and U = 0,...,C

$$f^t_i(U) = \underset{j \neq i}{\text{Minimum}} \begin{cases} f^{t-1}_j(U-q_i) + d_{ji} & i \in S \\ f^{t-1}_j(U + q_i) + d_{ji} & i \in D \end{cases}$$

where

$$f^2_i(U) = \begin{cases} d_{1i} & \text{if } U = q_r \text{ and } i \in S \\ \infty & \text{otherwise} \end{cases}$$

Then the length of the minimum CM-tour is a lower bound on the single vehicle model, and is given by

$$LB3 = \underset{i \in D}{\text{Minimum}} \{f^M_i(0) + d_{i1}\}$$

At stage M, the load at every node must be zero since all the deliveries must be completed up to this stage.

3.4 Lower Bounds Through Minimum CQM-tours

Both the capacity and the total load constraints (3.8) can be enforced on the minimum M-tours. Such an M-tour will be called a CQM-tour and the value of the minimum CQM-tour, LB4, is also a valid lower

bound. The dynamic programming recursive equations can be written as follows. Let

$f_i^t(Q,U)$ = length of the minimum t-path from node 1 to i which picks up a total load of Q and having a load U at node i.

Then for $i = 2,\ldots,M$; $t = 3,\ldots,M$; $Q = 0,\ldots,Q_T$ and $U =),\ldots,C$

$$f_i^t(Q,U) = \underset{j \neq i}{\text{Minimum}} \begin{cases} f_j^{t-1}(Q-q_i,U-q_i) + d_{ji} & \text{if } i \in S \\ f_j^{t-1}(Q,U+q_i) + d_{ji} & \text{if } i \in D \end{cases}$$

where

$$f_i^2(Q,U) = \begin{cases} d_{1i} & \text{if } Q = U = q_i \text{ and } i \in S \\ \infty & \text{otherwise} \end{cases}$$

Then LB4 can be computed as follows:

$$LB4 = \underset{i \in D}{\text{Minimum}} \{f_i^M(Q_T,0) + d_{i1}\}$$

The CQM-tour assures that any minimum M-tour will visit a combination of supply nodes such that a total load of Q_T will be picked up and the capacity will never be exceeded. Further, by setting $U = 0$ at stage M the M-tour is forced to visit a combination of demand nodes such that a total load of Q_t will be delivered.

The M-tours obtained by each of the lower bounding procedures will be connected tours. However these tours may contain subcycles which may be repeated several times.

Considering the various formulations, LB4 $\geq$ LB2 and LB4 $\geq$ LB3, since both state variables of LB2 and LB3 are imbedded in calculating LB4. However the calculations for LB4 would potentially require more computational effort compared to LB2 or LB3.

Each of the previous lower bounds can further be strengthened by appropriate Lagrangian ascent procedures (for example, see Bazaraa and Goode [8]).

4. A BRANCH AND BOUND ALGORITHM FOR THE SINGLE VEHICLE PICK-UP AND DELIVERY PROBLEM

The lower bounding schemes discussed in Section 3 were implemented in a branch and bound algorithm for the single vehicle PDP. Due to the lack of standard test problems in the literature, randomly generated test problems were used for testing the performance of the algorithm. The specifics of the algorithm, test problems and the computational results are given below.

4.1 The Dynamic Programming Network

The M-tour problem given by (3.1) through (3.6) is a shortest path problem over an acyclic M-partite network. In this network, arcs are defined by the x_{ij}^t variables, where each arc connects two different nodes at two consecutive stages. If there had been no precedence and capacity constraints on the tours, each node at any stage will be linked to every other node at the next stage. By making use of the special characteristics of the problem this network can be reduced, thus lower bounds can be strengthened. The reductions are as follows.

For convenience suppose

$$d_{ij}^t = d_{ij} \quad \text{for } i = j = t = 1,\ldots,M$$

a) $d_{1j}^t = \infty \qquad t = 2,\ldots,M; \quad j = 2,\ldots,M$

$d_{j1}^t = \infty \qquad t = 1,\ldots,M-1; \quad j = 2,\ldots,M$

The tour must start and end at the base node 1.

b) $d^{1}_{1j} = \infty \qquad j \in D$

After node one the vehicle can only go to a supply node.

c) $d^{M}_{b(r)a(r)} = \infty \qquad j \in S$

At stage M the vehicle must be at one of the demand nodes.

d) $d^{t}_{b(r)a(r)} = \infty \qquad r = 1,\ldots,N; \quad t = 1,\ldots,M$

A supply node a(r) cannot be visited immediately after its associated demand node b(r).

e) $d^{2}_{a(r)b(s)} = \infty \qquad r \neq s = 1,\ldots,N$

At stage 2 a demand node of an order cannot be visted after visting the supply node of another order.

f) $d^{3}_{b(r)b(s)} = \infty \qquad r \propto s = 1,\ldots,N$

At stage three two demand nodes cannot be visited one after another.

g) $d^{M-2}_{a(r)a(s)} = \infty \qquad r \propto s = 1,\ldots,N$

At stage M-2 two supply nodes cannot be visited.

h) $d^{M-1}_{a(r)b(s)} = \infty \qquad r \propto s = 1,\ldots,N$

At stage M-1 a supply node must be connected to only its associated demand node.

i) Capacity related reductions:

$$d^t_{a(r)a(s)} = \infty$$
$$d^t_{b(r)b(s)} = \infty \qquad \text{if } q_r + q_s > C \quad r \neq s = 1,\ldots,N;\ t = 1,\ldots,M$$
$$d^t_{a(r)b(s)} = \infty$$

Two nodes cannot be connected if the sum of their loads is greater than the capacity of the vehicle.

In enforcing the total load constraints, i.e., equation (3.7), the variable Q at each stage can only assume certain values. For example, at stage two, Q cannot be equal to the total load, since it is not possible to pick-up all the load at the first pick-up node visited. Likewise at stage M, the total pick-ups must equal to Q_T. For the vehicle there are two extreme node visit sequences in terms of pick-ups. The first one is to visit nodes in such a sequence that a minimum amount goods are picked-up. The other extreme is to visit nodes in an order such that a maximum amount of goods are picked-up. Suppose the tasks are ordered with respect to the nondecreasing order of loads, i.e.,

$$q_{[1]} < q_{[2]} < \cdots\cdots < q_{[N]}$$

Then the sequence which picks-up the minimum amount of load is the one that visits supply and demand nodes of each task with respect to the above ordering of tasks, Figure 1.

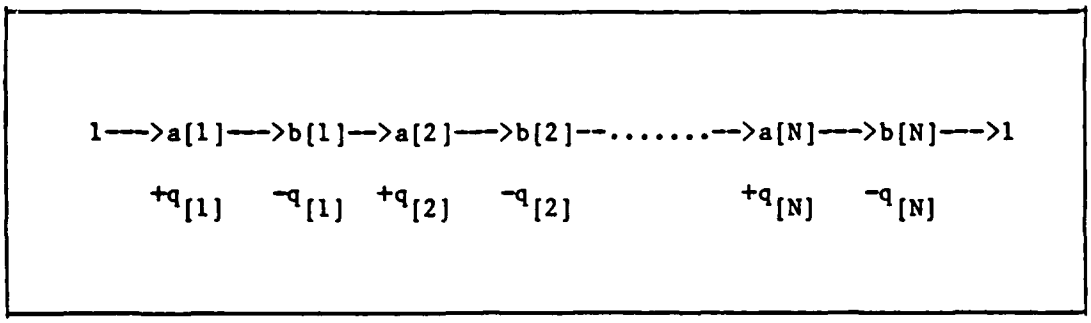


Figure 1. Representation of a Vehicle Tour Which Pick-up Minimum Load

A node sequence which picks-up the maximum amount of load would be the one which visits all the supply nodes first and then visits the remaining demand nodes. In this case the supply nodes are visited with respect to the noninceasing order of task loads, Figure 2.

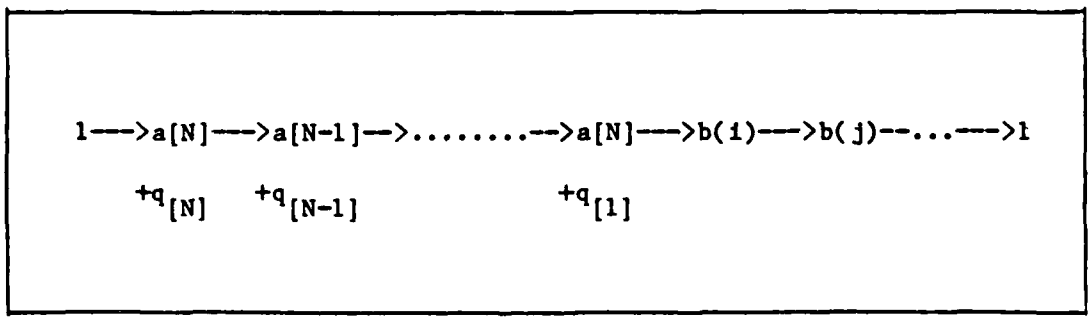


Figure 2. Representation of a Vehicle Tour Which Pick-ups Maximum Load

With the above properties, for each stage the minimum and maximum number of possible load pick-ups could be identified. Let (ℓ_t, u_t) be these minimum and maximums respectively. Then

$$\ell_{2t} = \sum_{n=1}^{t} q_{[n]} \qquad \text{for } t = 1,\ldots,N$$

$$\ell_{2t+1} = \ell_{2t}$$

$$u_t = \sum_{n=1}^{t-1} q_{[N-n-1]} \qquad \text{for } t = 2,\ldots,N+1$$

$$u_t = Q_t \qquad \text{for } t = N+2,\ldots,M$$

At each stage t the variable Q can assume values $\ell_t \leq Q \leq u_t$.

The 2-node subcycles could be eliminated without increasing the state space, by the procedure given Christofides et al [16]. It is sufficient to identify the best and the next best t-tours for every possible value of state variables at each stage.

Using the above described state space restrictions a computer code which computes minimum CQM-tours was developed by Kirca [23]. By proper definition of the state variables, namely C and Q, any of the lower bounds discussed can be computed by this algorithm.

4.2 Branch and Bound Algorithm

A branch and bound algorithm was developed for the single vehicle PDP. The algorithm is capable of utilizing different lower bounding schemes. The procedure creates partial tours and computes lower bounds for each of those. If the lower bound for any of these tours is greater than the value of the incumbent solution (upper bound), this partial tour is discarded from further consideration. In order not to repeat the already evaluated partial tours a systematic way of creating and evaluating partial tours is required. One such procedure is summarized below.

Step 0. Initialization:

Let $L = 1$, $X_1 = D$, $T = M + 1$, $G_1(T) = 1$

$P_1 = \{G_1(T)\}$, $Z^* = \infty$, $P^* = 0$

Compute a lower bound, Z_1, for the partial tour P_1. If $Z_1 > Z^*$ stop, no feasible solution exists. Otherwise go to Step 1.

Step 1. Candidate tour selection:

Pick a partial tour $P_\ell, 1 \leq \ell \leq L$, with $Z_\ell < Z^*$. If none exists stop, P^* is the optimal tour with value Z^*. Otherwise go to Step 2.

Step 2. Branching:

Let $T = M + 2 - 2\ P_\ell 1$

for each $i \in X$ repeat steps (2.1) through (2.3) and at end go to Step 3.

Step 2.1. $L = L + 1$

$G_L(t) = G_\ell(t) \qquad t = T + 1, \ldots, M + 1$

$G_L(T) = i$

$X_L = X_\ell - \{i\}$

if $i \in D$

$X_L = X_L \cup \{\text{source node of } i\}$

$P_L = P_\ell \cup \{G_L(T)\}$

Step 2.2. Compute a lower bound, Z_L, for the partial tour P_L. If $Z_L > Z^*$ let $Z_L = \infty$, and go to Step 2.2. Otherwise go to Step 2.3.

Step 2.3. If $T \neq 2$ to to Step 2.2. Otherwise let $Z^* = Z_L$ and $P^* = P_L$. Go to Step 2.1.

Step 3. Let $Z_\ell = B$ and go to Step 1.

By selection of the lower bounding scheme for Steps 0 and 2.2, and the partial tour selection rule for Step 1, the branch and bound algorithm can be specified. The algorithm developed here utilizs a "depth first" strategy. The partial tour which has the least lower bound among the highest level active partial tours is selected for further branching. This procedure minimizes computer storage requirements. At any instance, not more than M active branches exist in the search tree. Furthermore, this strategy reaches good quality upper bounds faster so that the "fathoming" efficiency at Step 2.2 is increased.

The upper bound is updated at Step 2.3 whenever a complete tour of M nodes, with a value less than the value of the incumbent solution is reached. In the test code, updating is performed whenever the lower bounding procedure yields a feasible M-tour with a value less than the value of the incumbent.

The algorithm is capable of computing lower bounds with any of the three M-tour lower bounding schemes described in Section 3. The lower bounds were computed by the algorithm given earlier in this section. Two different versions of the algorithm were developed. The first version called MULSOL, solves a new dynamic program over a new updated network for each of the partial tours evaluated. In the second version, called ONESOL, the dynamic program is solved only once at the root node, i.e., at the initialization step. Then the dynamic program functional values, $f_1^t(\ )$s, are stored. Those stored values are later used to compute lower bounds for each of the partial tours generated in Steps 2 through 4. Those bounds are computed as follows.

The advantage of ONESOL over MULSOL is that it requires less computational time since the dynamic program is solved only once. However the quality of the lower bounds are expected to be poorer compared to MULSOL. Further, in MULSOL, storage requirements are less than ONESOL since the dynamic program functional values need not be stored.

4.3 Computational Results

The branch and bound procedures were tested on randomly generated PDPs. For each problem, nodes were placed randomly on a two dimensional space and the distances between nodes are specified as integer Euclidia. For a given number of tasks, N, and a vehicle capacity C, 2N + 1 nodes were specified. Node 1 is fixed as the base node and supply and demand nodes for each task are determined as $(1 + r, N + r + 1)$ for $r = 1,\ldots,N$. The load for each task was drawn from a uniform distribution between 1 and C. The details of problem generation and problems tested can be found in reference [23].

A second set of problems with time constraints were also tested. Time windows on the pick-ups and deliveries were imposed on each of the test problems discussed above. The time windows were defined in terms of stages as suggested by Armstrong. A vehicle tour has 2N + 1 stages (stops). By specifying "Not Earlier than Delivery", NED, and "Not Later than Delivery", NLD, stages for each task time, constraints are implicitly defined.

All algorithms were all coded in FORTRAN IV, compiled in FTN (OPT=2), and executed on a CDC CYBER 730.

The initial test results with the Branch and Bound algorithms suggest that the ONESOL routine is faster than MULSOL. MULSOL was applied only with the CQM-tour bounding scheme.
The CQM-tour routine was not applied in ONESOL because of the excessive storage requirements for larger problems.

The algorithms are applied to the 5, 6, 7, and 8 task problems. for each task size, five different random problems were tested. In Tables 1 and 2 performance of the procedures are tabulated. A 120 cpu. sec.

Table 1. Computational Results with the M-tour Bounds for Single Vehicle Problems (N = 5 and N = 6)

PROBLEM N(M)	#	# of Branches	cpu. sec.	# of Branches	cpu. sec.	# of Branches	cpu. sec.
	51	11	13,003	223	.626	207	.731
	52	26	20.379	277	.645	123	.606
5(11)	53	13	15.705	461	1.040	409	1.101
	54	11	5.996	39	.286	24	.341
	55	18	16.847	408	.827	135	.562
	61	72	108.765	30222	43.882	10929	16.049
	62	108[1]	120.000[2]	1767	2.579	1440	2.709
6(13)	63	8	14.505	519	1.000	156	.999
	64	34	120.000[2]	2154	3.306	1549	3.167
	65	59[1]	120.000[2]	518	1.400	712	2.136

(1) Optimal reached at termination
(2) Terminated at this time limit

Table 2. Computational Results with the M-tour Bounds for Single Vehicle Problems (N+7 and N=8)

PROBLEM N(M)	#	ONESOL CM-TOUR # of Branches	ONESOL CM-TOUR cpu. sec.	ONESOL QM-TOUR # of Branches	ONESOL QM-TOUR cpu. sec
	71	1427	2.588	1487	3.867
	72	41524	65.207	58081	92.504
7(15)	73	1923	3.445	1015	3.078
	74	9765	15.494	4928	9.474
	75	213	1.026	468	2.638
	81	1678	4.056	-	-
	82	31379	54.201	-	-
8(17)	83	72865	120.000[1]	-	-
	84	68814	120.000[1]	-	-
	85	13094	21.804	-	-

(1) Terminated at this time limit

time limit was set for all problems, and whenever this limit was reached the procedure was terminated with the best solution found so far.

Computational results with the test problems suggest that the weakness in lower bounds with ONESOL is highly compensated by the computational speed. Furthermore the difference between the CM-tour and QM-tour bounding schemes was not apparent. Therefore a preference between routine is more favorable for the CM-tour procedure as compared to the QM-tour procedure.

As the number of tasks increases the solution effort also increases for all procedures. This fact suggests that the size of problems solvable by these procedures is limited due to the computational effort required. Also the storage requirements increase as the problem size increases. This is another factor limiting the successful implementation of the exact B&B procedures for the single vehicle PDPs with large numbers of tasks. The computational effort required and computer storage space requirements for different schemes are tabluated in Table 3.

Table 3. Computational Complexity and Memory Requirements with Different Bounding Schems and B&B Strategies

Routine	D.P. Effort	MULSOL Branches Effort	Memory Space	D.P. Effort	ONESOL Branches Effort	Memory Space
		CQM-tour				
M	M^2CQ	M^3CQ	2MCQ	M^3CQ	M^3CQ	2MCQ
QM-tour	M^3Q	M^3Q	2MQ	M^3Q	M	M^2Q
CM-tour	M^3C	M^3C	2MC	M^3C	M	M^2C

Computational results for the time constrained PDPs are summarized in Table 4. On the unconstrained problems, time windows were imposed.

Table 4. Time Constrained Random PDPs Solved with the CM-tour Lower Bounds

PROBLEM N	#	Optimal Value	Initial LB	No. of Branches	Total cpu. sec.
	51	544	427	14	.463
	52	476	472	7	.462
5	53	520	351	90	.535
	54	629	627	8	.392
	55	346	346	0	.398
	61	529	354	125	.852
	62	551	454	29	.664
6	63	502	408	19	.638
	64	664	486	52	.823
	65	428	405	44	.721
	71	518	516	7	.819
	72	603	547	207	1.081
7	73	699	622	46	.908
	74	713	672	52	.850
	75	627	583	11	.906
	81	652	532	742	2.343
	82	721	593	65	1.367
	83	706	647	121	1.419
	84	724	475	5873	10.555
	85	621	518	128	1.474

The effect of the time constraints are very significant in the sense that they reduce the state space for the lower bound computations, thus higher values of lower bounds were reached. Also the feasible space for the overall problems is reduced, and thus the search effort in the B&B phase is reduced. As a result, larger size PDPs are solvable in the presence of these constraints.

4.4 Bounds Through Lagrangian Relaxation

The major determinant of the efficiency of the B&B procedure is the strength of the lower bounds used. As the size of the problem increases the quality of the lower bounds decreases rapidly. One reason the strength of the lower bounds diminishs rapidly is that, as the problem size increases, the subcycle elimination procedures described earlier loose their effectiveness. Therefore, the M-tours find contain subcycles which repeat themselves several times. In the M-tours, the requirement to visit each node exactly once was relaxed. Hence, the procedure does not have control over the subcycles or force the tour to visit each node exactly once. One way to enforce these node visit constraints would be to dualize them and apply appropriate Lagrangian procedures.

Suppose we relax the precedence constraints in (2.8) and dualize the node visit constraints in (2.5). Then the resulting program is

$$\text{Max} \quad F(w) = \text{Min} \quad \sum_{t=1}^{M} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ij} x_{ij}^{t} + \sum_{i=1}^{M} w_i \left(\sum_{t=1}^{M} \sum_{j=1}^{M} x_{ij}^{t} - 1 \right) \qquad (4.1)$$

s.t. (2.2), (2.3), (2.4), (2.6), (2.7) and (2.9)

The objective function in (4.1) can be rewritten as

$$\max_{w} F(w) = \min \sum_{t=1}^{M} \sum_{i=1}^{M} \sum_{j=1}^{M} (d_{ij} + w_i)\, x_{ij}^{t} - \sum_{i=1}^{M} w_i \qquad (4.2)$$

For any value of the Lagrangian multipliers w, the solution of the above program in (4.2) results in a CM-tour problem, where the tour visits exactly M nodes and the vehicle capacity constraints are not violated. However the resulting tour may visit some nodes more than once and some nodes may not be visited at all.

It is obvious that any value of F(w) is a lower bound for the overall PDP. However, larger values of F(w) will yield larger lower bounds which could help significantly to reduce the search effort in the B&B phase. For this purpose a subgradient optimization procedure, suggested by Bazaraa and Goode for the TSP, was utilized to improve the value of F(w). The algorithm is called the "Ascent Procedure" is this discussion.

The algorithm developed for the single vehicle PDP is as follows.

STEP 1. Apply a Lagrangian Ascent Procedure. If the procedure results in a feasible PDP tour stop. F^{*} is the optimal value of the PDP and x^{*} is the optimal tour. Otherwise to go Step 2.

STEP 2. Branch and Bound Procedure: Apply the B&B procedure described in Section 4.2 (ONESOL version). Replace,

$$d_{ij} \leftarrow d_{ij} + w^* \qquad \text{for all } i,j$$

$$f_i^t(U) \leftarrow f_i^{*t}(U) \qquad \text{for all } i,t \text{ and } 0 \leq U \leq C$$

where $f_i^{*t}(U)$'s are the dynamic program functional values correspondong to the dual and primal variables (w^*, x^*) of the ascent phase.

The procedure was tested on the same randomly generated problems given in Tables 5 and 6. The computational results are encouraging.

Table 5. Computational experience with Single Vehicle Ascent+CM-tour routine for $5 \leq N \leq 8$

(all times are in cpu. sec.'s)

PROBLEM N	#	Before Ascent LB	After Ascent LB	Optimal Value	Ascent Time	B&B Time	Total Time
	51	338	446	446	2.291	-	2.291
	52	306	452	452	1.254	-	1.254
5	53	337	454	454	4.655	-	4.655
	54	510	557	557	.474	-	.474
	55	210	305	305	2.709	-	2.709
	61	171	451	451	7.742	-	7.742
	62	356	532	533	6.422	.035	6.457
6	63	319	444	444	3.385	-	3.385
	64	453	617	617	4.493	-	4.493
	65	297	317	348	6.506	.073	6.579
	71	344	460	460	4.059	-	4.059
	72	357	494	508	34.353	.672	33.681
7	73	391	502	502	13.048	-	13.048
	74	287	659	677	11.281	-	11.281
	75	320	406	406	12.057	-	12.057
	81	371	420	420	10.440	-	10.440
	82	449	589	622	42.439	.798	43.237
8	83	398	559	630	41.475	5.384	46.859
	84	288	462	477	69.771	1.159	70.930
	85	284	498	511	26.567	.976	27.543

Table 6. Computational Experience with Single Vehicle Ascent+CM-tour Routine for $9 \leq N \leq 12$

(all times are in cpu. sec.'s)

PROBLEM N	#	Before Ascent LB	After Ascent LB	Optimal Value	Ascent Time	B&B Time	Total Time
	91	269	492	537	80.796	5.365	86.161
	92	283	356	426	74.557	36.658	111.215
9	93	525	741	748	23.685	.524	24.209
	94	394	537	597	49.708	17.704	67.452
	95	409	589	638	55.554	3.005	58.559
	101	334	569	620	78.016	5.312	83.328
	102	346	576	646	89.089	150.912	240.000[1]
10	103	457	645	680	65.840	2.864	68.704
	104	353	498	647	102.705	137.303	240.000[1]
	105	306	465	546	106.937	19.488	126.425
	121	451	592	646	232.983	20.208	253.191
	122	365	781	826	197.360	40.312	237.672
12	123	477	673	734	140.545	65.055	205.600
	124	449	831	868	162.417	26.455	188.872
	125	410	655	695	105.568	10.288	115.856

(1) termination at this time limit

are encouraging. In Table 6 the computational results with the above parameters are tabulated. Column (2) gives the lower bounds obtained before the ascent procedure. Column (3) gives the lower bounds obtained at the termination of the ascent procedure.

As can be seen in Table 5, the ascent procedure results in either the optimal solution for the PDP, or produces lower bounds so close to the value of the optimal PDP, that the B&B phase quickly finds and verifies the optimal solution. Encouraged by these results for the small problems ($5 \leq N \leq 8$), the algorithm was tested on 9, 10, and 12 task problems. The results are given in Table 6, for randomly generated problems.

The computational results suggest that the Lagrangian ascent procedure produces good quality lower bounds, at the expense of additional computational effort. Throughout these experiments the ascent procedure resulted in good quality lower bounds (i.e., 90% of optimal) in early iterations. Then either no improvement was achieved or the improvements were not significant enough to justify the remaining computational effort. By proper selection of search parameters good quality lower bounds can be obtained more efficiently. Since the computational effort in the B&B phase is O(M) at each branch, a trade-off between the good quality lower bounds and B&B procedure could be investigated.

Although precedence constraints of (2.8) are relaxed for the above tests, the resulting CM-tours at the termination of the ascent phase were either feasible or near feasible with respect to these constraints. This may have resulted from the fact that in order for the M-tour be feasible with respect to the enforced constraints it should also perform pick-ups and deliveries such that the precedence constraints are satisfied, (or least violated).

In each of the ascent iterations a CM-tour is solved. The computational requirement for this problem is $O(m^3C)$. In the B&B phase, the lower bounds are computed by utilizing the dynamic programming

functional values found at the end of the ascent phase. In this case the computational complexity at each branch is O(M). The computer memory requirement for the whole procedure is $O(M^2C)$.

4.6 Bounds Through 1-trees

If the capacity and precedence constraints of the PDP model of Section 2.1 are relaxed, the resulting problem is a TSP. Thus any optimal solution or a lower bound for the TSP is also a lower bound for the PDP. Utilizing TSP lower bounds and enforcing the capacity and precedence constraints in the B&B phase could be one approach solving the PDP. If, on average, the value of the optimal PDP tour is not very far away from the value of the optimal TSP tour, then 1-tree bounds coupled with a duality approach may result in good quality lower bounds for the PDP.

4.7 Comparison of the Exact Single Vehicle Procedures

The performance of the three single vehicle PDP procedures presented here are compared in Tables 7 and 8. From these tables it can be seen

Table 7. Comparison of the Computational Effort for the Single Vehicle Methods

(all times are in cpu. sec.)

Problem Size N	CM-tour Total Time	Ascent+(CM-tour) Ascent Time	Ascent+(CM-tour) Total Time	Ascent+(1-tree) Ascent Time	Ascent+(1-tree) Total Time
5	.689	2.277	2.277	.167	.548
6	10.430	5.710	5.717	.240	.671
7	17.552	14.825	14.825	.392	3.416
8	64.012	38.138	39.810	.570	9.293
9	-	56.86	69.609	.571	97.901
10	-	88.517	151.691	-	-
12	-	-	200.238	-	-

Table 8. Average Strength of Lower Bounds Computed by Different Schemes (lower bound/optimal value)

Problem Size	CM-tour	Ascent CM-tour	Ascent 1-Tree
5	.768	1.000	.917
6	.667	.989	.939
7	.665	.987	.881
8	.672	.950	.868
9	.638	.921	.807
10	.572	.877	-
12	.571	.937	-

clearly that the procedures which utilize Lagrangian relaxation are superior to the one that does not. This results from the fact that Lagrangian relaxation yields good quality lower bounds. Comparison of the Lagrangian CM-tour and 1-tree approaches would be misleading. By more efficient ascent strategies a trade-off between the asent and B&B phases could be reached for the CM-tour scheme. This may result in reducing the computational effort at the ascent phase significantly. As can be seen for the eight and nine task problems the lower bounds obtained by the 1-tree approach are weaker than the CM-tour bounds. Thus the 1-tree routine requires more computational effort. It appears that for larger problems 1-tree bounds will be weaker and thus require more computational effort compared to the CM-tour scheme.

Thus far, in the literature there exist only two exact procedures for the single vehicle PDP. These are the procedures due to Psaraftis [25] and Armstrong [4]. Both of these procedures utilize direct application of dynamic programming technique on the PDP's. In Tables 9 and 10 the performance of these two methods are compared with the ones

Table 9. Comparison of the Single Vehicle Procedures with the Methods Exist in the Literature

Problem Size N(M)	Existing Methods Psaraftis (1)	Armstrong (2)	Proposed Methods(3) CM-tour	Ascent CM-tour	Ascent 1-tree
2(5)	.6	3.92	-	-	-
3(7)	.7	4.08	-	-	-
4(9)	1.1	-	-	-	-
5(11)	2.7	5.23	.685	2.277	.548
6(13)	9.6	-	10.430	5.717	.671
7(15)	46.8	124.72	17.552	14.825	3.416
8(17)	149.7	-	64.012	39.810	9.293
9(19)	591.4	-	-	69.609	97.901

(1) VAX/VMS of MIT

(2) Includes I/O operations and data generation on IBM 370/3303

(3) Average of five random problems for each N on CDC Cyber 730

Table 10. Computer Memory Requirements for Different Single Vehicle PDP Methods

Problem Size	Psaraftis Armstrong	CM-tour	Ascent CM-tour	Ascent 1-tree
N	$2N3^{N-1}+1$	$(2N+1)^2C$	$(2N+1)^2C$	$(2N+1)^2$
5	811	1210	1210	121
10	393611	4410	4410	441
15	1.43×10^8	9610	9610	961
20	4.64×10^{10}	16810	16810	1681
30	4.11×10^{15}	36000	36000	3600

proposed here. Although both authors report their computational results for the dial-a-ride problems, for the same number of tasks, N, the problems they report and the problems tested here may be considered essentially equivalent.

Due to differences in problem assumptions, computers, and coding, a direct comparison of the three methods is difficult. Hence, tables 9 and 10 "suggest" that, for the instances compared, the procedures proposed here seem to be more efficient than the ones in Psaraftis and Armstrong. Even the least efficient method, the CM-tour without the ascent phase, performs better than those procedures compared. In the case of computer memory requirements the procedures suggested here (see Table 8) compare favorably to the ones available in the literature. In the algorithms of Psaraftis and Armstrong, memory requirements are exponential whereas here the proposed methods require polynomial memory with respect to the problem size N.

5.0 CONCLUSIONS

Several models and algorithms for the single-vehicle-exact pickup and delivery problem were presented and analyzed. Computational results indicate that they compare favorably with those available in the literature. There is still a ways to go to solve more than small ($N \leq 12$) PDP problems exactly.

REFERENCES

1. Ali, A., R. Helgason, and J. Kennington (1981). "An Airforce Logistics Decision Support System Using Multicommodity Network Models", Tech. Report OR 79012, Department of General Business, University of Texas, Austin, (Feb.)

2. Angel, R., W. Caudle, R. Noonan, and A. Whinston (1972). "Computer Assisted School Bus Scheduling", Man. Scie., 18(6), B279-88.

3. Appelgren, L. (1969). "A Column Generation Algorithm for a Ship Scheduling Problem", Trans. Scie., 3, 53-68.

4. Armstrong, G. (1981). "The Single and Multiple Vehicle Pickup and Delivery Problem: Exact and Heuristic Algorithms", Ph.D. Thesis, The University of Tennessee, Knoxville (June).

5. Assad, A., M. Ball, L. Bodin, and B. Golden (1981). "Combined Distribution Routing and Scheduling in a Large Commercial Firm", Proceedings of 1981 Northeast AIDS Conference, (R. Pavan, P. Anderson, eds.) Boston, 99-102.

6. Baker, E. (1980). "Time Oriented Vehicle Routing and the Travelling Salesman Problem", School of Business, University of Florida, (August).

7. Ball, M., L. Bodin, B. Golden, A. Assad, and C. Stathes (1980). "A Strategic Truck Fleet Sizing Problem Analysed by a Routing Heuristic", Working Paper #81-006, University of Maryland (February).

8 Bazaraa, M. and J. Goode (1977). "The Travelling Salesman: A Duality Approach", Math. Prog. 13, 221-37.

9. Bellmore, M., G. Bennington, and S. Lubore (1968). "A Maximum Utility Solution to a Vehicle Constrained Tanker Scheduling Problem", Nav. Res. Log. Q., 15, 403-11.

10. Bodin, L. and T. Sexton (1979). "The Subscriber Dial-a-Ride Problem", University of Maryland, Report #UMCP-UMTA-1-79.

11. Bodin. L. and B. Golden (1981). "Classification in Vehicle Routing and Scheduling", Networks, 12(2), 97-108.

12. Bodin. L., B. Golden, A. Assad, and M. Ball (1981). "The State of the Art in the Routing and Scheduling of Vehicles and Crews", University of Maryland, Report #UMTA/BMGT/MSS #81-001, (September).

13. Christofides, N. (1976). "Worst-Case Analaysis of a New Heuristic for the Travelling Salesman Problem", Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (February).

14. Christofides, N. (1976). "The Vehicle Routing Problem", R.A.I.R.O., 10, 55-70.

15. Christofides., N., A. Mingozzi, and P. Toth (1981). "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Math Prog., 20, 255-82.

16. Christofides, N., A. Mingozzi, and P. Toth (1981). "State Space Relaxation Procedures for the Computation of Bounds to Routing Problems", Networks, 11(2), 145-64.

17. Clark, G., and W. Wright (1964). "Scheduling of Vehicles from A Central Depot to a Number of Delivery Points", Oper. Res., 12, 568-81.

18. Cullen, F., J. Jarvis and H. Ratliff (1981). "Set Partitioning Based Heuristics for Interactive Routing," Networks, 11(2), 125-44.

19. Dantzig, G. and P. Fulkerson (1954). "Minimizing the Number of Tankers to Meet a Fixed Schedule", Nav. Res. Log. Q., 1, 217-22.

20. Desrosiers, J., J. Ferland, J. Rousseau, G. Lapalme, and L. Chapleau (1980). "A School Busing System", Publication #164, Centre de Recherche sur Les Transports, University of Montreal, (April).

21. Houck, D., J. Picard, M. Quayrenne, R. Vemuganti (1977). "The Travelling Salesman Problem and Shortest n-paths", University of Maryland.

22. Jarvis, J. and O. Kirca (1982). "Heuristic for Pallet Movement in Naval Supply", PDRC Report #83-10, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia (October).

23. Kirca, O. (1983). "Models and Procedures for the Pick-up and Delivery Problem", Unpublished Doctoral Dissertation, Georgia Institute of Technology, October.

24. Newton, R. and W. Thomas (1974). "Bus Routing in a Multi-School System", Compt. and Ops. Res., 1, 213-22.

25. Psaraftis, H. (1980). "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Dial-a-Ride Problem", Trans. Scie., 2, 130-54.

26. Psaraftis, H. "Two Heuristic Algorithms for the Single Vehicle Many-to-Many Dial-a-Ride Problems", submitted to Trans. Res.

27. Sexton, T. and L. Bodin (1980). "The Single Vehicle Many-to-Many Routing and Scheduling Problem with Desired Delivery Times", Working Paper #80-014, University of Maryland.

28. Verderber, W. (1974). "Automated Pupil Transportation", Compr. and Ops. Res., 1, 235-45.

29. Winchell, R., R. Melton, and M. Natrella (1981). "Automated Vehicle Scheduling (AVS). Programmer's Instruction Manual for Burroughs B3500 Computers", Report #DTNSRDC-81/017, David W. Taylor Naval Ship Research and Development Center, Maryland.

END

DTIC

7-86