A Set-Partitioning-Based Exact Algorithm for the Vehicle Routing Problem

Yogesh Agarwal
AT&T Bell Labs, Technical Staff, West Long Branch, New Jersey 07748

Kamlesh Mathur and Harvey M. Salkin
Department of Operations Research, Case Western Reserve University, Cleveland, Ohio 44106

In this paper, we discuss a computationally viable algorithm based on a set-partitioning formulation of the Vehicle Routing Problem (VRP). Implementation strategies based on theoretical as well as empirical results are developed. Some computational results are presented. It is shown that a set-partitioning formulation to the VRP, although well known for a long time, deserves considerable research efforts beyond those we present here.

1. INTRODUCTION

The Vehicle Routing Problem (VRP), also referred to as the Vehicle Dispatching or the Delivery Problem, addresses the design of vehicle routes from a depot to a number of stops (demand points) requiring deliveries in less than truckload (LTL) quantities. Each route originates at the depot, makes several deliveries not exceeding the capacity of the vehicle, and then returns to the depot. The objective is to design a set of feasible routes so as to minimize the total distribution cost while satisfying all customer demands.

The distribution costs in the real world may be composed of, among others, fuel costs, driver’s pay, depreciation, and maintenance of vehicles. In general, however, most cost elements are positively correlated to the distance traveled, and, thus, for modeling purposes, minimizing the total distance traveled is a reasonable, objective function. In addition to the truck-capacity restriction, other less important constraints may include, among others, total time/distance limit for each route and customer time-window constraints. Ignoring these secondary constraints, a simplified version of the Vehicle Routing Problem, termed Basic VRP, may be stated as follows:

Given a set of n customers, each with demand w_i, a central depot, the matrix of all distances d_{ij}, and the vehicles each with capacity W, find a set of vehicle routes so
as to minimize the total distance traveled such that the total demand on each route does not exceed the truck capacity $W$ and each customer is visited on exactly one route.

The VRP has recently attracted considerable research effort. However, because of the combinatorial nature of the problem, most of the work has been devoted to the development of heuristic algorithms. Some of the well-known heuristic approaches are by Clarke and Wright [6], Gillett and Miller [11], Foster and Ryan [10], and Fisher and Jaikumar [9]. There have been a few attempts to solve the VRP optimally, notably by Christofides et al. [5], and Laporte et al. [13]. Christofides et al.'s approach is based on the shortest path and spanning tree relaxations of the VRP. The relaxations are solved using dynamic programming to produce bounds on the VRP that are used in a branch and bound search to find the optimal solution. The algorithm of Laporte et al. [13] is based on a well-known Traveling Salesman Problem (TSP) formulation involving subtour breaking constraints. Although both approaches can address the basic VRP adequately, there does not appear to be any way of incorporating other constraints such as time/distance or window-time constraints within these approaches.

2. SET-PARTITIONING FORMULATION OF THE VRP

In this paper, we present an exact algorithm as well as a heuristic method for the VRP based on the set-partitioning (SP) formulation. In the SP formulation, each vehicle route is represented by a binary $n$-vector $a_j$. The element $a_{ij}$ of vector $a_j$ is 1 if stop $i$ ($i = 1, 2, \ldots, n$) is visited on route $a_j$, and 0 otherwise. With each column $a_j$, we associate a cost $c_j$ representing the total distance traveled on the route. Note that column $a_j$ represents only the subset of stops visited on the route and not the sequencing of stops on it. Therefore, to associate the cost $c_j$, a TSP must be solved over the depot and the subset of stops on $a_j$, (i.e., the set $\{i : a_{ij} = 1 \in a_j\}$). Also, each route $a_j$ is implicitly assumed to be feasible with respect to the truck capacity and other constraints. Then, the VRP can be posed as the SP problem below:

$$\text{Minimize} \quad \sum_j c_j \cdot x_j$$

(SP) subject to \quad \sum_j a_{ij} \cdot x_j = e

and \quad x_j = 0 \text{ or } 1 \text{ (all } j).$$

Here $e$ is the $n$-vector of all ones, the binary variables $x_j$ determine the inclusion ($x_j = 1$) or exclusion ($x_j = 0$) of the route represented by the column $a_j$ in the solution, and the constraints ensure that each stop is visited on exactly one route. Note that neither the customer demands $w$, nor the vehicle capacity $W$ appear explicitly in the formulation. This follows since each route $a_j$ is implicitly assumed to be feasible or, equivalently, each $a_j$ must satisfy the following constraints:

$$w \cdot a_j \leq W,$$

where $w = (w_1, w_2, \ldots, w_n)$ is the row vector representing stop demands.

The SP formulation of the VRP has been long recognized. It was first proposed by Balinski and Quandt [4] for the delivery problem that is a close relative of the VRP.
The heuristic algorithm of Foster and Ryan [10] and the interactive heuristics of Cullen et al. [7] are also based on the SP formulation. Desrosiers et al. [8] have successfully used the SP formulation to design an exact algorithm for a different routing problem that has time-window constraints but no vehicle-capacity constraints. We compare and contrast their work with ours in the next section. However, no serious attempt has been made to use this approach to design exact algorithms for the standard VRP. The major drawback of this approach is that the number of possible columns $a_j$ representing feasible vehicle routes can be very large even for fairly small problems. It is computationally impractical to generate and store all possible columns or even a reasonable subset, let alone solve the SP problem over them. In this paper, we discuss some results and strategies that make the SP approach computationally more attractive. Moreover, the SP formulation is particularly appealing because most real-world situations can be adequately modeled within its framework. More specifically, since the columns $a_j$ are implicitly assumed to represent feasible routes, virtually any criteria of route feasibility can be enforced by permitting only feasible columns to enter the formulation. Also, since the cost $c_j$ of each column is evaluated separately, even complex cost functions involving (e.g.) a fixed truck cost or waiting cost are permissible.

3. BASIC APPROACH

Our approach consists of three basic steps. In the first step, the Linear Programming (LP) relaxation of the VRP is solved using a column-generation scheme. The column-generation subproblem and its solution are described in Section 4. Since the LP solution is almost certainly fractional, it is of little use by itself. However, the LP optimal basis and corresponding set of dual variables contain valuable information that can be used to generate good heuristic solutions as well as the optimal solution. The second step is to extract a heuristic solution using the columns in the LP optimal basis by solving a set-covering problem. This set-covering problem is defined over the binary columns in the optimal basis and their binary subcolumns ($a_j$ is a subcolumn of $a_k$ if $a_j \subseteq a_k$). The details of this procedure are given in Section 7. In the third step, relatively small set of columns is generated that is guaranteed to contain the optimal VRP solution according to a well-known theorem on SP problems. The optimal solution is then obtained by solving SP over this set of columns. The size of the set depends on the known upper bound on the solution value; the tighter the upper bound, the smaller the set. Hence, the importance of generating a good heuristic solution in the second step.

As mentioned above, Desrosiers et al. [8] use a very similar approach to solve the time-window routing problem (TWRP), though both were developed independently (our interest in the SP formulation was inspired by the work of Foster and Ryan [10]). The TWRP is different from standard VRP in that there are no vehicle capacity constraints, but certain time windows must be observed for each trip. Therefore, the nature of the resulting column-generation subproblem is very different between the two problems. In the TWRP case, the subproblem is a time-window constrained shortest-path problem, which is solved by a dynamic programming approach. In our case, the subproblem is to find a profit-maximizing subtour subject to capacity con-
strain and is solved by the branch and bound method. There are also significant differences in the strategies used to improve convergence of LP and to generate optimal solutions after solving LP relaxation.

In short, though the basic formulation and strategy is similar in both cases, the implementation details are quite different because of the differences in the nature of the two problems. Laporte and Nobert [14] have contrasted the work of Desrosiers et al. [8] with previous expositions of our work (Agarwal [1–3]). We believe that future development of algorithms in this area could benefit by combining ideas from both papers.

4. THE COLUMN-GENERATION SCHEME

The LP relaxation of SP may be written as

\[
\begin{align*}
\text{Minimize} & \quad c \cdot x \\
\text{subject to} & \quad A \cdot x = e \\
\text{and} & \quad x \geq 0.
\end{align*}
\]

As discussed above, the matrix \( A \) contains an enormous number of columns and it is computationally impossible to explicitly generate all columns in advance. In this section, we develop a column-generation scheme where at each iteration of the simplex algorithm only the columns in the current basis \( B \) are known and any other column is generated as and when needed.

At any simplex iteration corresponding to a basis \( B \), if \( u = c_B B^{-1} \) is the vector of dual variables where \( c_B \) is the cost vector corresponding to the basic variables, then among all the remaining columns (which are not known explicitly), we need to identify the column \( a_k \) with the least reduced cost \( \bar{c}_k \); that is:

\[
\bar{c}_k = \min_j (c_j - u \cdot a_j).
\]

If \( \bar{c}_k \geq 0 \), the current solution is optimal; otherwise, column \( a_k \) enters the basis. Note that \( A \) consists of all binary columns \( y = \{y_i\} \), that is \( y_i = 0 \) or \( 1 \) \((i = 1, \ldots, n)\), which satisfies:

\[
\sum_{i=1}^n w_i y_i \leq w.
\]

Thus, column \( a_k \) is equal to \( y^* \), the optimal solution to the problem below:

\[
\begin{align*}
\text{Minimize} & \quad Z = f(y) - \sum_{i=1}^n u_i \cdot y_i \\
\text{subject to} & \quad \sum_{i=1}^n w_i \cdot y_i \leq W \\
\text{and} & \quad y_i = 0 \text{ or } 1 \quad (i = 1, \ldots, n),
\end{align*}
\]

where \( f(y) \) is the cost of the optimal TSP tour over the stops represented by column \( y \).
ALGORITHM FOR THE VEHICLE ROUTING PROBLEM

Note that \( f(y) \) is a very complex function of the vector \( y \) and its value can be obtained only by actually solving a TSP. This makes the subproblem difficult to solve. Note that if \( f(y) \) is replaced by a linear function of \( y \) (say \( f(y) = \sum p_i y_i \)), then the subproblem (P1) reduces to a Knapsack Problem, which is relatively easy to solve. Although we cannot write \( f(y) \) as a linear function, we show in the next section that it is possible to find a linear function that is a lower bounds on \( f(y) \). If (P1) is solved after replacing \( f(y) \) by such a linear function, we obtain a lower bound on the objective function \( Z \). Generating this lower bound at each node in a branch and bound scheme allows us to solve (P1) efficiently. The development of the lower bound function, related theorems, and details of the branch and bound approach are given in the next section.

5. A BRANCH AND BOUND ALGORITHM TO SOLVE (P1)

In the branch and bound algorithm, branching is based on setting the variable \( y_i \) to zero or one, that is, to exclude or include, respectively, stop \( i \) in the route being generated. Thus, at any stage of the branch and bound procedure, each stop will belong to one of three states; namely, included in the route \( (y_i = 1) \), excluded from the route \( (y_i = 0) \), or not decided on \( (y_i \) not used for branching yet). In light of this, we rewrite the original notation and define some additional terms.

5.1. NOTATION

- \( n \) number of customers to be served
- \( i, j, k \) indices of stops
- \( (i, j) \) are joining stops \( i \) and \( j \)
- \( u_i \) dual variable associated with constraint \( i \) (stop \( i \)) in problem (LP)
- \( w_i \) demand of customer \( i \)
- \( d_{ij} \) distance from customer \( i \) to customer \( j \)
- \( d_{i0} \) distance from depot to customer
- \( W \) truck capacity
- \( S \) \( \{i: i = 0, 1, 2, \ldots, n\} \) set of all stops and the depot \( (i = 0) \)
- \( S_i \) \( \{y_i = 1\} \) set of stops included in the route
- \( S_0 \) \( \{y_i = 0\} \) set of stops excluded from the route
- \( S_u \) undecided; or set \( S_u = S - S_i \cup S_0 \)
- \( |S| \) cardinality of the set \( S_i \), i.e., the number of stops in the set \( S_i \)
- \( T \) or \( T(S_i) \) set of arcs on a given tour covering stops in \( S_i \)
- \( T^* \) or \( T^*(S_i) \) set of arcs on the optimal tour covering stops in \( S_i \)
- \( f(S_i) \) cost of optimal tour over \( S_i \)
- \( q(i, j, k) \) \( d_{jk} + d_{ik} - d_{ij} \); cost of inserting \( i \) between \( j \) and \( k \)
- \( q(S_i) \) Minimum over \( \{q(i, j, k)\} \)
- \( q(T) \) Minimum over \( \{q(i, j, k)\} \)
- \( W \) or \( W(T) \) \( \sum_{i\in S_i} w_i \); unused capacity of the vehicle

5.2. COMPUTING A LINEAR LOWER BOUND ON \( f(y) \)

To develop a lower bound on the function \( f(y) \), we need the following theorem on the TSP.
Theorem 1. Given a set of points $S$, an optimal TSP tour $T^*(S)$, and a point $i$ not in $S$, if $q_i(T^*) = q_i(S)$, then the tour obtained by inserting $i$ in its best position in $T^*(S)$ will be the optimal TSP tour for the set $S' = S \cup \{i\}$.

Proof. Recall that the length of tour $T^*(S)$ is denoted by $f(S)$. After inserting $i$ in its best position, let the new tour be $T_i(S')$ with length $l_1$. Then

$$l_1 = f(S) + q_i(T^*)$$

or

$$f(S) = l_1 - q_i(T^*).$$

(1)

Let $T^*(S')$ be the optimal tour over the set $S'$ with length $f(S')$. Suppose stop $i$ appears between stops $j$ and $k$ in this tour. Then, we can create a tour $T_2(S)$ by deleting $i$ from $T^*(S)$ and joining $j$ and $k$ directly. Let $l_2$ be the length of this tour. Then

$$l_2 = f(S') - q_(j,k).$$

Since $q_(j,k) \geq q_i(S) = q_i(T^*)$ by definition and by assumption, respectively, we have

$$l_2 \leq f(S') - q_i(T^*).$$

(2)

Since $f(S)$ is the length of the optimal tour over $S$, we must have $l_2 \geq f(S)$. The last inequality with Equations (1) and (2) yield

$$f(S') - q_i(T^*) \geq f(S) = l_1 - q_i(T^*)$$

or

$$f(S') \geq l_1 - q_i(T^*).$$

Since $l_1$ is the distance of the feasible tour $T_1$ and $f(S')$ is the length of the optimal tour, the above relation must hold with equality. Hence, $T_i$ is optimal for $S'$. \hfill \blacksquare$

The corollaries below follow directly from the theorem.

Corollary 1. Given set $S$, an optimal tour $T^*(S)$ and a point $i$ not in $S$; let $S' = S \cup \{i\}$, then $f(S') \geq f(S) + q_i(S)$.

Corollary 2. If $LB(S)$ is a lower bound on the TSP solution over $S$, then $LB(S) + q_i(S)$ is a valid lower bound on $f(S')$.

With reference to the notation, consider two sets $S_1$ and $S_2$ such that $S_1 \subseteq S_2$. Let $m = |S_2| - |S_1|$. Then, the following theorem defines a lower bound on $f(S_2)$.

Theorem 2. Given the sets $S_1$ and $S_2$ such that $S_1 \subseteq S_2$, let $S_3 = S_2 - S_1$, then

$$f(S_3) \geq f(S_1) + \sum_{i \in S_3} q_i(S_1)/m.$$  (3)

Proof. Let $q_{\max} = \text{Maximum}_{i \in S_3} q_i(S_1)$. 

Then, for $i \in S_3$, we have

$$(q(S_i)/m) \leq (q_{\text{max}}/m)$$

or

$$\sum_{i \in S_3} (q(S_i)/m) \leq m \cdot (q_{\text{max}}/m) = q_{\text{max}}.$$

Then using Corollary 1

$$f(S_1) + \sum_{i \in S_3} (q(S_i)/m) \leq f(S_1) + q_{\text{max}} \leq f(S_2).$$

It is obvious that if in Equation (3) $m$ is replaced by an upper bound on $|S_2| - |S_1|$ and $f(S_1)$ by a lower bound, the right-hand side will still be a lower bound on $f(S_2)$. Given sets $S_1$, $S_2$, and $S_0$, if we know that the utmost $m$ more stops can join set $S_1$, then according to the theorem, the linear function below defines a lower bound on $f(y)$ at the current node of the branch and bound tree:

$$LB(f(y)) = f(S_1) + \sum_{i \in S_3} (q(S_i)/m) \cdot y_i.$$

Also, given sets $S_1$, $S_2$, and $S_0$ at any stage of the branch and bound, we can, based on the truck capacity, compute the maximum number of additional stops that can join the set $S_1$ denoted by the number $m(S_1)$. Specifically, to find $m(S_1)$, we solve the Knapsack Problem below:

Maximize

$$m(S_1) = \sum_{i \in S_3} y_i$$

subject to

$$\sum_{i \in S_3} w_i \cdot y_i \leq W_r$$

and

$$y_i = 0 \text{ or } 1 \text{ for } i \in S_3.$$

This problem is solved trivially by ranking $w_i$ in increasing order and finding the largest $k$ such that the sum of the first $k$ values does not exceed $W_r$. Then, $m(S_1) = k$.

Having computed $m(S_1)$, we define a quantity $p_i$ for each $i \in S_3$ using

$$p_i = (q(S_i)/m(S_1)) - u_i \quad i \in S_3.$$

Then, the problem below gives a valid lower bound on the subproblem objective function $Z$:

Minimize

$$\sum_{i \in S_1} u_i + \sum_{i \in S_3} p_i \cdot y_i$$

subject to

$$\sum_{i \in S_3} w_i \cdot y_i \leq W_r$$

and

$$y_i = 0 \text{ or } 1 \text{ for } i \in S_3.$$
Since the solution of the Knapsack Problem (P3) is used solely to provide a lower bound for the subproblem objective function $Z$, it makes sense to relax the integrality constraints. This allows us to solve the Knapsack Problem by inspection at each node of the branch and bound tree while still yielding a valid lower bound. Similarly, rather than solving a TSP at each node to obtain $f(S_r)$, we can replace $f(S_r)$ with a lower bound. Such a lower bound can be easily computed by virtue of Corollary 1 using the information at the parent node. This strategy results in a substantial saving of computer time to solve the subproblem.

5.3. Node Selection and Branching Procedures

We follow a combination of “depth-first” node selection criterion and “least lower-bound” strategy for branching. Depth-first branching is pursued by selecting the most recently created node for further branching. As in most depth-first searches, the branching scheme is designed to produce a good incumbent solution quickly. The branching variable at each node is selected based on the least lower-bound criterion. In particular, at any node, given $S_1$, $S_0$, and $S_2$, we choose the branch variable so as to minimize the following function:

$$\text{Minimize } \sum_{i \in S_1} (q(S_i) - u_i)/w_i.$$ 

Here, $q(S_i)$ is a good measure of the contribution of stop $i$ to $f(y)$. Therefore, the numerator $q(S_i) - u_i$ is a good measure of the contribution made by stop $i$ to the subproblem’s objective function if it joins the set $S_1$. On the other hand, $w_i$ is the resource requirement (in terms of the truck-capacity constraint) associated with this contribution. Thus, the branching strategy seeks to provide the least contribution to the objective function to be minimized for the unit consumption of the truck capacity.

6. SOLVING THE LP RELAXATION OF VRP

Having designed a column-generation scheme, we attempt to solve the LP relaxation starting from an identity basis. In this basis, each identity column $e_i$ represents a route visiting a single stop (i.e., stop $i$). The cost of column $e_i$ is the round-trip distance of stop $i$ from the depot, that is, $c_i = 2 \cdot d_{ir}$, and the dual vector $u$ is given by $u = c_p \cdot B^{-1} = c_p$, or $u_i = c_i$. At each step of the simplex algorithm, the column generation scheme is used to generate a single column with the least reduced cost to enter the basis. The procedure terminates when the column generated has a nonnegative reduced cost.

This relatively straightforward procedure for solving the LP relaxation turned out to be computationally inefficient and demonstrated slow convergence properties for the following reasons:

1) The values of the dual variables fluctuated widely from iteration to iteration and sometimes acquired large negative values. It can be shown that in the optimal LP solution the dual variables will always be nonnegative if the triangular inequality is satisfied in the distance matrix. Therefore, the presence of dual variables with negative values means that the generated column is unlikely to be in the optimal basis.
TABLE I. Initial vs. optimal values of the dual variables for a 15-stop problem.

<table>
<thead>
<tr>
<th>Stop no.</th>
<th>Initial value</th>
<th>Optimal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>66</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>52</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>44</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>46</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>56</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>24</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>58</td>
<td>42</td>
</tr>
<tr>
<td>14</td>
<td>36</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>16</td>
</tr>
</tbody>
</table>

(2) The optimal values of the dual variables are generally much smaller than are the initial values \( u_i = c_i = 2d_{ip} \) corresponding to an identity basis (cf. Table I). Therefore, columns generated early in the algorithm are not likely to stay in the optimal basis.

(3) Since only one column is generated at a time, the column-generation problem has to be solved many times, which is computationally expensive.

(4) The initial value of the objective function is quite high compared to its final value. If the procedure could be started from a heuristically generated solution closer to the LP optimal, many simplex iterations could be saved.

Based on the above observations, the following strategies were used to improve computational efficiency:

1. (1) The values of the dual variables fed into the column-generation subproblem should be closer to the optimal values (which are not known a priori). This can be done if we (somehow) estimate the optimal values of the dual variables and not allow their values to fluctuate widely from iteration to iteration. We discuss these strategies in Sections 6.1 and 6.2, respectively.

2. (2) To reduce the computational effort, the number of times the column-generation subroutine is used should be minimized.

6.1. A Priori Estimation of the Dual Variables

The first strategy was implemented using an a priori estimation of the optimal values of the dual variables. Specifically, it was conjectured that the optimal values of the dual variables are related to the problem attributes such as stop demands, distance of stops from the depot, and the geographical clustering of stops. The motivation behind the conjecture is that in the subproblem the dual variable \( u_i \) may be considered to represent the reward associated with including the stop \( i \) on the route. Therefore, it makes sense to say that the reward should reflect the relative cost of servicing the
stop. This cost is positively correlated to the stop demand and the distance of the stop from the depot. Also, if the stop is isolated from other stops, it costs more to service when compared with one close to other stops.

Our method of estimating the values of dual variables requires a reasonably good heuristic solution to the VRP, such as the one obtained by the Clarke and Wright [6] procedure. Note that in the optimal basis of the LP relaxation of VRP each column in the basis has a reduced cost of zero. This means that the sum of the values of the dual variables for stops on a route is equal to the route cost. Although this property may not hold for any actual VRP solution, it seems reasonable to assume that the reduced cost of each route in a good VRP solution will be relatively close to zero. Thus, the problem of estimating values of the dual variables reduces to the problem of allocating the cost of each route among the stops on the route for the heuristic solution.

Let column $a_j$ with cost $c_j$ represent a given route in the heuristic solution and $S_j = \{i:a_{ij} = 1\}$, the set of stops on route $j$. Then, we propose the statistical model below for estimating values for the dual variable $u_i$ corresponding to stop $i \in S_j$:

$$u_i = \beta_1 \bar{d}_i + \beta_2 \bar{w}_i + \beta_3 \bar{q}_i, \quad i \in S_j,$$

where

$$\bar{u}_i = \frac{\hat{u}_i}{c_j},$$

$$\bar{d}_i = \frac{d_i}{\sum_{i \in S_j} d_{ij}},$$

$$\bar{w}_i = \frac{w_i}{\sum_{i \in S_j} w_j},$$

$$\bar{q}_i = \frac{q_i}{\sum_{i \in S_j} q_i},$$

$$q_i = \min_{j,k} \{d_{ji} + d_{kj} - d_{ik}; w_i + w_j + w_k \leq W\}$$

and $\hat{u}_i = estimated\ value\ of\ the\ ith\ dual\ variable.$

A possible value for the $\beta$'s can be $\beta_1 = \beta_2 = \beta_3 = 1/3$. (Note that $\beta_1 + \beta_2 + \beta_3 = 1.0$.) However, on a large set of randomly generated problem, we found that $\beta_1 = 0.55$, $\beta_2 = 0.48$, and $\beta_3 = 0.11$ produces much better starting values for the $u_i$'s.

When the optimal values of the dual variable were regressed against the values using the above model for seven test problems, a combined correlation coefficient of 0.85 was obtained. That the model produces reasonable estimates for the optimal dual values is demonstrated by Table II, which lists the computed and actual values of optimal
TABLE II. Optimal vs. estimated values of the dual variables for a 15-stop problem.

<table>
<thead>
<tr>
<th>Stop no.</th>
<th>Optimal dual variables</th>
<th>Estimated dual variables</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abs.</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>38</td>
<td>29</td>
<td>−9</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>34</td>
<td>32</td>
<td>−2</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>18</td>
<td>−10</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>42</td>
<td>32</td>
<td>−10</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Sum</td>
<td>324</td>
<td>333</td>
<td></td>
</tr>
</tbody>
</table>

Average absolute error = 4.7; maximum absolute error = 10.

...duals for a 15-stop problem. However, further work in this direction may produce still better methods.

6.2. Imposing Upper Bounds on the Dual Variables

Excessive fluctuations in the value of the dual variables, which is a major cause of slow convergence, can be avoided by imposing artificial upper bounds on the dual variables. More specifically, consider the dual (D) problem below, where upper-bound constraints of the form \( u_i \leq t_i \) have been added.

\[
\begin{align*}
\text{Maximize} & \quad u \cdot e \\
\text{(D)} & \quad \text{subject to:} \quad u \cdot A & \leq c \\
& \quad u \cdot I & \leq t \\
& \quad u & \text{unrestricted.}
\end{align*}
\]

Taking the dual of the dual problem, we get the following modified primal:

\[
\begin{align*}
\text{Minimize} & \quad c \cdot x + t \cdot y \\
\text{(LP')} & \quad \text{subject to:} \quad A \cdot x + I \cdot y = e \\
& \quad x \geq 0 \\
& \quad y \geq 0.
\end{align*}
\]

Note that each upperbound constraint \( u_i \leq t_i \) [where \( u = (u_i) \) and \( t = (t_i) \)] is equivalent to introducing an additional identity column \( e \), in the primal with cost \( t \), and
TABLE III. CPU time saved by imposing upper bounds on the dual variables.

<table>
<thead>
<tr>
<th>Problem no.</th>
<th>No. stops</th>
<th>Without upperbounds on dual variable</th>
<th>With upperbounds on dual variable</th>
<th>Abs.</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>64</td>
<td>75</td>
<td>-11</td>
<td>-17</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>32</td>
<td>43</td>
<td>-11</td>
<td>-34</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>30</td>
<td>18</td>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>40</td>
<td>18</td>
<td>22</td>
<td>55</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>81</td>
<td>52</td>
<td>29</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>&gt;180*</td>
<td>32</td>
<td>150</td>
<td>82</td>
</tr>
</tbody>
</table>

*Run was terminated after 180-second time limit.

associated variable \( y_i \). Clearly, such upper bounds artificially constrain the dual space and, hence, overrelax the primal space. This means that a solution to LP' may not be feasible for the original problem LP. Such a condition will be indicated by one or more variables \( y_i \) being basic at a nonzero value in the optimal solution of LP'. However, it is intuitively obvious and can be easily shown that if in the optimal solution of LP' all variables \( y_i \) with artificial cost \( t_i \) are nonbasic or are basic at value zero, then the solution is also optimal for LP. If the optimal solution of LP' does contain artificial variables \( y_i \) at nonzero levels, then the upper bounding constraints for such variables can be relaxed gradually by setting \( t_i \) to \((1 + \alpha)t_i\) \((\alpha > 0)\), and solving LP' again. This procedure is repeated until no artificial variable is basic at nonzero values.

Note here that the value of the upper bound \( t_i \) will never exceed \( 2 \cdot d_{xi} \) because the column \( e_i \) with cost \( 2 \cdot d_{xi} \) also corresponds to a valid vehicle route visiting a single stop \( i \). Therefore, if \( t_i = 2 \cdot d_{xi} \), \( y_i \) may be basic at a nonzero level and the solution to LP' will still be feasible for LP.

Initially, the upper bounds are set to the estimated values obtained by the procedure described in the previous section. Then, they are gradually relaxed until they are no longer binding. This procedure produces substantial reduction in CPU time needed for solving the LP relaxation as indicated in Table III.

Several other techniques were used to improve computational efficiency, some of which are briefly mentioned below; details are in Agarwal [1].

(1) It was noticed that the dual variables sometimes acquire large negative values that tend to slow down the convergence rate. It can be shown that if the triangular inequality is satisfied in the distance matrix, the optimal dual values will always be nonnegative. Nonnegativity of the dual variables can be ensured by adding the negative identity columns with zero cost to the primal problem. (Note that this is equivalent to replacing equality constraints with \( \geq \) constraints in the LP.) It can be shown that the value of the optimal solution is the same for both LPs. Computational experience demonstrates that the convergence rate is significantly improved by introducing the negative identity columns.
(2) The major bottleneck in solving the LP relaxation is in solving the column-generation subproblem. It was noticed that generating several columns rather than one requires only marginally more time. If \( k \) best columns are generated at each interaction, chances are that several of them will be able to enter the basis to provide faster convergence. This is analogous to multiple pricing in simplex algorithm.

7. GENERATING A HEURISTIC SOLUTION TO VRP FROM THE OPTIMAL LINEAR PROGRAMMING SOLUTION

The LP relaxed solution of the VRP, in general, is of little value by itself because it will usually be noninteger. However, the optimal LP basis and the dual variables contain useful information that can be used to generate heuristic as well as optimal solutions to the VRP. In our method, described below, of generating an optimal solution to the VRP, the availability of a good heuristic solution is of key importance. Therefore, we first suggest a way of extracting good heuristic solutions to the VRP from the optimal LP basis.

It can be shown that if the triangular inequality is satisfied in the distance matrix, the VRP can equally well be posed as the set-covering (SC) problem given below rather than as a SP problem.

\[
\begin{align*}
\text{Minimize} & \quad c \cdot x \\
\text{subject to} & \quad A \cdot x \geq e \\
& \quad x_i = 0 \text{ or } 1 \quad (i = 1, \ldots, n).
\end{align*}
\]

We make the following observations relating to the SP and the SC formulations of the VRP. LSP and LSC are used to define the linear programming relaxation of SP and SC, respectively.

1. Any feasible solution of SP is also a feasible solution of SC, though the converse is not true.

2. Any feasible solution to SC is either feasible to SP or can be converted into a feasible solution to SP with lower cost. In particular, if given a feasible solution to SC not feasible to SP, it implies that one or more stops are covered on multiple routes. Each such stop can be eliminated from all routes but one. Since the triangular inequality is satisfied in the distance matrix, such elimination would result in a decreased total cost while producing a feasible solution to SP.

3. An optimal solution of SP is also optimal to SC and vice versa. (There may be exceptions to this rule in the case where the triangular inequality satisfies \( d_{ik} = d_{ij} + d_{jk} \). In such a case, there may exist optimal SC solutions not feasible to SP. However, they can be reduced to feasible and optimal solutions to SP with the same cost.)

4. There exists an optimal solution to LSP for which the dual variables are non-negative.

5. If the triangular inequality is satisfied, any optimal solution of LSP is also optimal for LSC.

6. Any feasible basis for LSP or LSC corresponds to a feasible solution of SC though not necessarily of SP.
Based on the above observations, we can use the following approach to generate a heuristic solution to the VRP from an LP relaxed solution.

**Step 1.** Construct a good set covering solution using the columns in the LP basis (per Observation 1).

**Step 2.** Reduce this solution to the best possible SP solution (per Observation 2).

A procedure for Step 1 is given in Salkin [16]. Step 2 can be solved by generating the necessary subcolumns of columns in the SC solution and solving SP over them. A subcolumn is defined as a column obtained by replacing one or more ones by zeros in the given column. Note that if the original column represents a feasible vehicle route, then the subcolumns are also feasible. In most cases, the number of subcolumns to be generated will be relatively small because, generally, the SC solution found via Stop 1 will have few overlapping columns (i.e., columns having one or more stops in common). This general approach can be enhanced further to produce a better heuristic solution (see Agarwal [1]). Test results given in Table IV indicate that our approach is quite effective. The heuristic procedure produced optimal solutions on four out of seven problems tested and was only slightly higher than the optimal solution in the other three cases.

**8. OPTIMAL SOLUTIONS TO THE VRP**

The approach for optimal solutions to the VRP is based on the well-known result below (see Pierce and Lasky [15]).

**Theorem 3.** In SP, if the cost coefficients $c_i$ are replaced by the reduced costs $\hat{c}_i = c_i - u \cdot a_i$ then the resulting set-partitioning problem $SP'$ has the same set of optimal solutions as does the original problem.

We use the reduced cost corresponding to the LP optimal solution to define $SP'$. Since the reduced costs at the LP optimal are nonnegative, we have the corollary below.

**Corollary 3.** If $Z_d$ is the value of the LP optimal solution and $Z_U$ is a known upper
bound on the SP solution, then the optimal SP solution cannot contain a column with reduced cost greater than \( Z_U - Z_L \).

Using these results, an obvious approach to obtain an optimal solution to the VRP is

**Step 1.** Solve the LP relaxation of the VRP and let the value of the objective function be \( Z_L \).

**Step 2.** Extract a good SC solution from the optimal LP basis and convert it to the best possible SP solution using the approach discussed in Section 7. Let the objective function value of this solution be \( Z_U \).

**Step 3.** Generate all columns with reduced cost less than or equal to \( Z_U - Z_L \) using the column-generation scheme.

**Step 4.** Solve SP over the set of columns generated. The resulting solution is the optimal solution to the VRP.

The difficulty with this approach is that if the gap \( Z_U - Z_L \) is large, the number of columns with reduced cost less than \( Z_U - Z_L \) may be significant. In that case, the computer time required to generate all such columns and then to solve the SP problem over them may be excessive.

However, a closer inspection of the above approach reveals that it may not be necessary to generate all columns with \( \bar{c} \leq Z_U - Z_L \). In particular, suppose that in the optimal solution the column with the largest value for \( \bar{c} \) has \( \bar{c} = t \). Then, we need only enumerate columns with \( \bar{c} \leq t \) and solve SP over these in order to obtain the optimal solution. In most cases, \( t \) will be much smaller than \( Z_U - Z_L \). Let \( n(\emptyset) \) represent the number of columns with \( \bar{c} = \emptyset \). Then, \( n(t) \) may be considerably smaller than \( n(Z_U - Z_L) \). However, the value of \( t \) is not known in advance. If we use an arbitrary value of \( t \) with the above approach and \( Z_{SP} - Z_L \) (here \( Z_{SP} \) is the value of the SP solution over the columns generated) turns out to be smaller than \( t \), then we can be sure that the solution is optimal. However, note that if \( t \) is not sufficiently large the columns generated may not contain any SP solution at all.

One way to circumvent this problem is to set \( t \) to the largest \( \bar{c} \) in the heuristic solution. By doing this, we can be sure that the columns generated contain at least one SP solution because the heuristic solution itself is one such solution. Based on this observation, we can use the following procedure to get the optimal solution.

**Step 1.** Generate the heuristic solution to the VRP using the approach discussed above and let \( t \) be the largest reduced cost in the solution.

**Step 2.** Generate all columns with reduced cost smaller than or equal to \( t \) and solve SP over these. If the difference \( Z_{SP} - Z_L \) is less than or equal to \( t \), then it is the optimal solution; stop. Otherwise, go to Step 3.

**Step 3.** Generate all columns with reduced cost smaller than or equal to \( Z_{SP} - Z_L \) and solve SP over these. The resulting solution is optimal; stop.

If we terminate in Step 2, the computational effort may be much smaller than what is needed in enumerating all columns up to a reduced cost of \( Z_U - Z_L \). If the test in
TABLE V. Approach for optimal solutions—relevant computational figures.

<table>
<thead>
<tr>
<th>Prob. no.</th>
<th>$Z_t$</th>
<th>$Z_U$ heuristic</th>
<th>$Z_U - Z_t$</th>
<th>$n(Z_U - Z_t)$</th>
<th>$\hat{t} = \max \hat{c}$ in heuristic solution</th>
<th>$n(t)$</th>
<th>$Z_{SP}$</th>
<th>$Z_{SP} - Z_t$</th>
<th>$\hat{t} = \max \hat{c}$ in optimal solution</th>
<th>$N(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>374</td>
<td>375</td>
<td>1</td>
<td>35</td>
<td>1</td>
<td>35</td>
<td>1</td>
<td>375</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>479</td>
<td>505</td>
<td>26</td>
<td>$&gt; 800$</td>
<td>18</td>
<td>331</td>
<td>15</td>
<td>494</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>326</td>
<td>332</td>
<td>6</td>
<td>36</td>
<td>5</td>
<td>29</td>
<td>6</td>
<td>332</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>268</td>
<td>279</td>
<td>12</td>
<td>258</td>
<td>7</td>
<td>95</td>
<td>8</td>
<td>276</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>430</td>
<td>430</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>430</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>351</td>
<td>376</td>
<td>25</td>
<td>$&gt; 800$</td>
<td>12</td>
<td>370</td>
<td>7</td>
<td>358</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>606</td>
<td>607</td>
<td>1</td>
<td>29</td>
<td>1</td>
<td>29</td>
<td>1</td>
<td>607</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 2 fails, the procedure may take even more time than using $t$ equal to the differences $Z_U - Z_t$. Nevertheless, the failure of the test in Step 2 does not always mean that the effort spent in Step 2 is entirely wasted. If $Z_{SP}$ turns out to be smaller than the original upper bound $Z_t$, then this procedure may still turn out to be more efficient than starting with $t$ equal to the difference $Z_U - Z_t$.

Table V lists computational experience with this procedure. Note that in all seven cases the number of columns generated in Step 2 is sufficient to produce the optimal solution as well as to guarantee optimality. Columns 5 and 7 of the table list the values of $n(Z_U - Z_t)$ and $n(t)$, respectively. Note that $n(t)$ is much smaller than $n(Z_U - Z_t)$ in three out of seven cases. The difference will get more and more pronounced as the problems get larger.

9. COMPUTATIONAL TESTING

We tested our algorithm on the seven problems reported in Christofides et al. [5] for which the authors have reported the optimal solution. These problems are the ones numbered 4-10 in Christofides et al. [5]. Problems 1-3 were not used because the data for them are given in the form of a distance matrix rather than of stop coordinates. The problems are given in Table VI. Note that the CDC 7600 computer used by
Christofides et al. is roughly 7.5\textsuperscript{1} times faster than is the IBM 370/4281 that we used. Taking this into account, our algorithm is roughly 13 times faster than is the Christofides' algorithm. However, we are using a heuristic procedure to solve the TSP at the column-generation stage. An optimizing algorithm is used only when the column generated by the heuristic algorithm has positive reduced cost. Moreover, since the size of the TSP's solved is very small (maximum of six stops on any route among all seven problems), the TSP heuristic is almost certain to produce the optimal solution (see Appendix).

Slight differences in the solution values reported by the two algorithms are due to the fact that in our program all distance data are represented as integers rather than as real values.

10. CONCLUSION

We have presented an exact algorithm for the VRP based on the SP approach. Although the design of heuristics was not addressed in detail, the ideas presented here lead to the development of effective heuristics based on the SP approach. The SP formulation of the VRP, though well known for a long time, has received only limited attention. Based on the findings reported here, we believe that it has far more potential for the development of effective algorithms for the VRP than hitherto perceived and that the SP approach deserves considerably more serious research efforts beyond what we have presented here.

APPENDIX. A HEURISTIC ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM (TSP)

Various insertion heuristics based on convex hulls as a starting TSP tour have been proposed to solve the TSP (see Golden and Stewart [12] for a good survey and computational results). In this appendix, however, we describe an insertion heuristic procedure that is a variation of convex hull insertion procedure of Stewart [17]. This modified algorithm has been found to be more suitable for our algorithm because, beside giving a good TSP solution, it also provides a good lower bound on the optimal solution.

It is well known that for plainer TSP's, the relative order of points in the optimal tour is the same as in the convex hull. In other words, if points \( j \) and \( k \) are two nonadjacent points in the convex hull and \( i \) is a point not in a convex hull, then the sequence \( j-i-k \) will never appear in the optimal tour. It, therefore, seems logical to use the convex hull as an initial subtour and then insert other points (or stops) based on some insertion rule.

If \( S \) is the set of points in the convex hull and \( A \) is the set of nonadjacent point pairs in the convex hull, then let us define a new quantity \( q_j(S) \) as follows:

\[
q_j(S) = \min_{\{i \in S, j \notin S \}} q(j, k).
\]

\textsuperscript{1}CDC 7600 has a processing speed of 15 mips (million instructions per second) as opposed to 2 mips of IBM 370/428a; figures were obtained from IBM and CDC, respectively.
TABLE AI. TSP heuristic and lowerbounds—computational results.

<table>
<thead>
<tr>
<th>No. stops</th>
<th>Average LB</th>
<th>Average UB</th>
<th>Ratio (LB/UB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>219</td>
<td>219</td>
<td>.999</td>
</tr>
<tr>
<td>8</td>
<td>257</td>
<td>260</td>
<td>.989</td>
</tr>
<tr>
<td>10</td>
<td>277</td>
<td>283</td>
<td>.977</td>
</tr>
<tr>
<td>12</td>
<td>284</td>
<td>297</td>
<td>.957</td>
</tr>
<tr>
<td>14</td>
<td>307</td>
<td>330</td>
<td>.931</td>
</tr>
<tr>
<td>16</td>
<td>313</td>
<td>344</td>
<td>.910</td>
</tr>
<tr>
<td>18</td>
<td>320</td>
<td>357</td>
<td>.899</td>
</tr>
<tr>
<td>20</td>
<td>323</td>
<td>380</td>
<td>.850</td>
</tr>
</tbody>
</table>

Using this definition, we now state an algorithm that gives a heuristic solution of the TSP as well as a good lower bound on the optimal solution.

**Algorithm**

Convex Maximum Insertion Heuristic With Lower Bound

**Step 0.** Construct the convex hull of points in the stop set with $S$, and $A$ defined as above. Let $S = S$, $T = T$, LB = the distance of tour $T$, and UB = LB.

**Step 2.** Select $i$ $\in S$ that maximizes $q_i(T)$ and compute $q_i(S)$ for this $i$. Insert $i$ in its best position in $T$, and add it to set $S$. Let LB = LB + $q_i(S)$ and UB = UB + $q_i(T)$.

**Step 3.** If all points have been inserted, stop; otherwise, go to Step 2.

The algorithm described above was tested on randomly generated problems of sizes ranging from six to 20 stops in increments of two stops. For each problem size, 20 problems were generated. For each set of problems, the average tour length, average lower bound, and the ratio of lower bound to upper bound are reported in Table AI. Note that for small problem sizes (up to 10 stops), the LB/UB ratio is very close to one. This implies that the solution is optimal most of the time in this size range.

**References**


Received October 1987
Accepted January 1989