Chapter 4

Set-Covering-Based Algorithms for the Capacitated VRP

Julien Bramel
David Simchi-Levi

4.1 Introduction

In this chapter we present several set-covering-based approaches for solving the Capacitated VRP (CVRP) and provide an analysis of the effectiveness of the approach. Throughout the chapter we consider the symmetric case, the CVRP, although the presented methods may be applied to the ACVRP as well. For this purpose, let the index set of the n customers be denoted $V = \{1, 2, \ldots, n\}$. We let 0 denote the depot and $V^0 = V \cup \{0\}$ the node set of the corresponding complete graph. Associated with customer $i \in V$ is the *demand* $d_i > 0$, which represents the load that must be picked up at customer i's location. We let C denote the vehicle capacity, and we assume there are K vehicles available to perform the delivery. Clearly, feasibility requires that $d_i \leq C$ for each $i \in V$. Let t_{ij} denote the length of edge (i, j) with $i, j \in V^0$. It is assumed that the distances t_{ij} satisfy the triangle inequality; otherwise, one can add a large constant cost to all nodes (or to all edges).

A classical method, first suggested by Balinski and Quandt [3], for solving the CVRP is based on formulating the problem as a set-covering problem. The idea is as follows. Enumerate all feasible routes, where a feasible route is one that starts and ends at the depot and picks up a total load not exceeding C. Let the index set of all feasible routes be $\mathcal{R} = \{1, 2, \ldots, R\}$. Let c_r be the *cost* (e.g., length) of route r, and let $S_r \subseteq V$ denote those customers appearing in route r for all $r \in \mathcal{R}$. Define

$$\alpha_{ir} = \begin{cases} 1 & \text{if customer } i \text{ is served in route } r, \\ 0 & \text{otherwise} \end{cases}$$

for each customer $i \in V$ and each route $r \in \mathcal{R}$. Also, for every $r \in \mathcal{R}$, let

$$y_r = \begin{cases} 1 & \text{if route } r \text{ is in the optimal solution,} \\ 0 & \text{otherwise.} \end{cases}$$

In the set-covering formulation of the CVRP, the objective is to select a minimum-cost set of feasible routes such that each customer is included in some route. It is

$$\text{(P)} \quad \min \sum_{r \in \mathcal{R}} c_r y_r$$

subject to

$$\sum_{r \in \mathcal{R}} \alpha_{ir} y_r \geq 1 \qquad \forall\, i \in V, \tag{4.1}$$

$$\sum_{r \in \mathcal{R}} y_r \leq K, \tag{4.2}$$

$$y_r \in \{0, 1\} \qquad \forall\, r \in \mathcal{R}.$$

Constraints (4.1) require that each customer appear in at least one route, while constraints (4.2) impose that at most K route be used. Observe that we have written constraints (4.1) as inequality constraints instead of equality constraints. The formulation with equality constraints is equivalent since we have assumed that the distance matrix $\{t_{ij}\}$ satisfies the triangle inequality and therefore each customer will be visited exactly once in the optimal solution. The formulation with inequality constraints will be used here since it turns out to be easier to work with for implementation.

This mathematical programming formulation was used successfully by Cullen, Jarvis, and Ratliff [13] to design heuristic methods for the VRP. Exact algorithms based on this method were developed by Agarwal, Mathur, and Salkin [1] and more recently by Bixby [5] and Hadjiconstantinou, Christofides, and Mingozzi [18]. The method is quite general and can be applied to a large number of problems. We list only a few examples here. For the VRP with time windows and *no capacity constraint*, Desrosiers, Soumis, and Desrochers [16] considered this same model and solved a number of problems to optimality. For the CVRP with time windows (VRPTW) (i.e., with a capacity constraint), Desrochers, Desrosiers, and Solomon [14] devise a branch-and-bound algorithm to solve a number of Solomon's [26] original time-window constrained problems to optimality or near optimality. In the context of the multidepot vehicle scheduling problem, Ribeiro and Soumis [24] successfully used this approach. Similar methods have also been used to solve crew scheduling problems; see, for instance, Hoffman and Padberg [21]. Finally, the survey of Desrosiers et al. [15] is an excellent source for column generation–based approaches to crew scheduling problems, particularly in urban transit systems and for airline companies.

The general algorithmic form common to the set-covering-based methods described in this chapter is as follows. In the first step, the linear programming relaxation of the set-covering problem (obtained by removing the integrality constraint on the y variables) is solved using the method of *column generation* (without enumerating all possible routes). The resulting fractional optimal solution value is then a lower bound on the value of the optimal integer solution. Then, from the set of columns generated so far (which may only be a small part of $\mathcal{R}$), an integer solution is sought using, e.g., a cutting plane or branch-and-cut approach. This solution is not guaranteed to be the optimal integer solution over all columns of $\mathcal{R}$, but it is likely to be close. If a branch-and-price approach is used, additional columns

are generated at each node of the branch-and-bound tree, resulting in the optimal integer solution over all columns in $\mathcal{R}$.

In the next section, we describe the column generation problem. In section 4.3, we review a number of methods that have been developed to solve the linear programming relaxation of problem P, specifically, the column generation problem. In section 4.4, we describe some of the methods that can be used to find an optimal or near-optimal integer solution to P. In section 4.5, we present some of the computational results on the methods we have described. Finally, in section 4.6, we provide analyses that shed some light on why a method of this type can be effective.

4.2 Solving the Linear Programming Relaxation of P

To solve the linear programming relaxation of problem P without enumerating all the routes, we can use the column generation technique. A detailed explanation of this method is given below, but the general idea is as follows. A portion of all possible routes is enumerated, and the linear relaxation with this partial route set is solved. The solution to this linear program is then used to determine if there are any routes not included in the formulation that can further reduce the objective function value. This is the column generation step. Using the values of the optimal dual variables (with respect to the partial route set), we solve a simpler optimization problem where we identify if there is a route that should be included in the formulation. Then the linear relaxation of this expanded problem is resolved. This is continued until no additional routes are found that can reduce the objective function value. In that case, we can show that an optimal solution to the linear program is found, one that is optimal for the complete route set.

Specifically, we first enumerate a partial set of routes $\mathcal{R}' \subseteq \mathcal{R}$ and formulate the corresponding linear relaxation of the set-covering problem with respect to this set:

$$(\mathrm{P}') \quad \min \sum_{r \in \mathcal{R}'} c_r y_r$$

subject to

$$\sum_{r \in \mathcal{R}'} \alpha_{ir} y_r \geq 1 \qquad \forall\, i \in V, \tag{4.3}$$

$$\sum_{r \in \mathcal{R}'} y_r \leq K, \tag{4.4}$$

$$y_r \geq 0 \qquad \forall\, r \in \mathcal{R}'.$$

Let $\bar{y}$ be the optimal solution to problem P′, and let $\bar{\pi} = \{\bar{\pi}_1, \bar{\pi}_2, \ldots, \bar{\pi}_n\}$ be the corresponding optimal dual variables associated with constraints (4.3). Let $\bar{\theta}$ be the optimal dual variable associate with constraint (4.4). We would like to know whether $\bar{y}$ (or, equivalently, $(\bar{\pi}, \bar{\theta})$) is optimal for the linear relaxation of problem P (respectively, the dual of the lin-

ear relaxation of problem P). To answer this question, observe that the dual of the linear relaxation of problem P is

$$(\mathrm{P}_D) \quad \max \sum_{i \in V} \pi_i - K\theta$$

subject to

$$\begin{aligned} &\sum_{i \in V} \alpha_{ir}\pi_i - \theta \le c_r \qquad \forall\, r \in \mathcal{R}, \\ &\pi_i \ge 0 \qquad \forall\, i \in V, \\ &\theta \ge 0. \end{aligned} \tag{4.5}$$

Clearly, if $(\bar{\pi}, \bar{\theta})$ satisfy every constraint in (4.5), then it is optimal for problem P_D and therefore $\bar{y}$ is optimal for the linear programming relaxation of problem P. How can we check whether $(\bar{\pi}, \bar{\theta})$ satisfies every constraint in problem P_D? Observe that the vector $(\bar{\pi}, \bar{\theta})$ is not feasible in problem P_D if we can identify a single constraint, r, such that

$$\sum_{i \in V} \alpha_{ir}\bar{\pi}_i > c_r + \bar{\theta}. \tag{4.6}$$

Consequently, if we can find a column r minimizing the quantity $c_r - \sum_{i \in V} \alpha_{ir}\bar{\pi}_i$ and this quantity is less than $-\bar{\theta}$, then a violated constraint is found. In that case the current vector $(\bar{\pi}, \bar{\theta})$ is not optimal for problem P_D. The corresponding column just found can be added to the formulation of problem P, which is solved again. The process repeats itself until no violated constraint (negative reduced cost column) is found; in this case we have found the optimal solution to the linear relaxation of problem P (the vector $\bar{y}$) and the optimal solution to problem P_D (the vector $(\bar{\pi}, \bar{\theta})$).

The column-generation problem is then to identify a feasible route $r \in \mathcal{R}$ that satisfies (4.6). Define $\bar{c}_r$ to be the *reduced cost* of column r, i.e., $\bar{c}_r = c_r + \bar{\theta} - \sum_{i \in S_r} \bar{\pi}_i$, for each $r \in \mathcal{R}$. Also define $d(S) = \sum_{i \in S} d_i$ for any $S \subseteq V$. The task is then to solve the column generation problem, which is

$$(\mathrm{CG}) \qquad \bar{c}_{\min} = \operatorname{Min} \left\{ \bar{c}_r : d(S_r) \le C \right\}.$$

It is not clear how this column-generation problem, CG, should be solved. Problem CG is itself NP-hard since, even given S_r, evaluating $\bar{c}_r$ (or c_r) requires solving the Traveling Salesman Problem (TSP) with respect to vertex set $S_r \cup \{0\}$. We consider several approaches to this subproblem in the next section. This includes the work of Agarwal, Mathur, and Salkin [1], Bixby, Coullard, and Simchi-Levi [6], Bixby [5], and Hadjiconstantinou, Christofides, and Mingozzi [18] on the CVRP, and the related work of Desrochers, Desrosiers, and Solomon [14] on the VRPTW.

In summary, the column-generation algorithm for solving the linear relaxation of problem P can be described as follows:

Column-Generation Algorithm

Step 1. Generate an initial set of columns $\mathcal{R}'$.

Step 2. Solve problem P′ and get optimal primal variables, $\bar{y}$, and optimal dual variables, $(\bar{\pi}, \bar{\theta})$.

Step 3. Solve problem CG, or identify routes $r \in \mathcal{R}$ satisfying $\bar{c}_r < 0$.
Step 4. For every $r \in \mathcal{R}$ with $\bar{c}_r < 0$ add the column r to $\mathcal{R}'$ and go to Step 2.
Step 5. If no routes r have $\bar{c}_r < 0$, i.e., $\bar{c}_{\min} \geq 0$, then stop.

The procedure produces a vector $\bar{y}$ which is the optimal solution to the linear relaxation of problem P. The objective function value $\sum_{r \in \mathcal{R}'} c_r \bar{y}_r$ is then a lower bound on the optimal solution value to the CVRP, i.e., the optimal integer solution value to P.

We note here a number of implementation tricks that can improve the convergence of the column generation algorithm. The column generation step (Step 3) usually turns out to be the most time consuming. To reduce the computation time of this step, the following additional features can be implemented. First, it is important to generate a good set of initial routes in Step 1. To do this, a large number of quick heuristics for the CVRP can be used. In fact, if a good dual solution is available, then it can be used to help generate routes with low reduced cost (with respect to this dual solution). Several methods for estimating good dual variables were given by Agarwal, Mathur, and Salkin [1] and Hadjiconstantinou, Christofides, and Mingozzi [18]. Second, it is important that in each iteration of Step 3 a number of routes with negative reduced cost be generated, not just one. In addition, it is particularly helpful to generate sets of new columns that are disjoint (as in an integer solution).

4.3 Set-Covering-Based Solution Methods

We describe four methods that have been developed to solve, or nearly solve, the linear programming relaxation P′ of the set-covering problem. The first three deal specifically with solving CG or generating a lower bound on $\bar{c}_{\min}$, the minimal reduced cost of a feasible route. The last method diverges from these in that it attempts to solve directly the dual P_D using a branch-and-bound method. In section 4.4 we consider several approaches for solving the integer program. We then give computational results on all the methods we have described.

4.3.1 Branch-and-Bound Algorithm for Problem CG

Agarwal, Mathur, and Salkin [1] devised a branch-and-bound algorithm to solve problem CG. This branch-and-bound algorithm is based on developing a lower bound on $\bar{c}_r$ for any route $r \in \mathcal{R}$.

The branch-and-bound approach constructs a route of minimum reduced cost that satisfies the constraint on the vehicle capacity. Branching is based on selecting a customer $i \in V$ and considering the two subproblems: find a minimum reduced cost route that includes i and find a minimum reduced cost route that excludes i. For this purpose, the method uses the following *branching* variables:

$$x_i = \begin{cases} 1 & \text{if customer } i \text{ is in the route,} \\ 0 & \text{otherwise} \end{cases}$$

for each $i \in V$. At each node of the branch-and-bound tree, there are three sets of interest. Let $S_1 \subseteq V$ denote those customers that must be included in the route (i.e., those customers

$i \in V$ for whom x_i has been set to 1). The set $S_0 \subseteq V$ consists of those customers that cannot be included in the route (i.e., those customers i for whom x_i has been set to 0). The set $S_x = V \setminus (S_0 \cup S_1)$ consists of those customers that have not yet been branched on.

At each node of the branch-and-bound tree, a lower bound on $\bar{c}_{\min}$ is calculated. For this purpose, let $c(S)$ denote the length of an optimal traveling salesman tour through set $S \cup \{0\}$ with $S \subseteq V$. This is a lower bound on the minimal reduced cost of a feasible route. Let S denote the route with minimal reduced cost in this part of the branch-and-bound tree. Then $S_1 \subseteq S$ and $S \subseteq S_1 \cup S_x$. To construct the bound, for any set $T \subset V$ and a customer $i \notin T$ define $q_i(T) = \min_{j,k \in T}\{t_{ji} + t_{ik} - t_{jk}\}$. Then it is easy to see that for all $T \subset V$ and $i \notin T$,

$$c(T \cup \{i\}) \geq c(T) + q_i(T). \tag{4.7}$$

In general, if $M = S \setminus S_1$ and $m = |M|$ then it is clear that

$$c(S) \geq c(S_1) + \sum_{i \in M} q_i(S_1)/m.$$

Therefore, given S_1, S_0, and S_x (i.e., at a particular node of the branch and bound tree), if we know that at most m additional customers can be added to the route, then

$$c(S) \geq c(S_1) + \sum_{i \in S_x} x_i \cdot \frac{q_i(S_1)}{m}.$$

In addition, it is simple to get an estimate of the value of m based on the demand sizes and the remaining vehicle capacity $\bar{C} = C - \sum_{i \in S_1} d_i$. This can be done by ranking the demand sizes in S_x in increasing order and finding the largest value of k such that the sum of the first k values does not exceed $\bar{C}$. Then m is set to this k. Define p_i, for each $i \in S_x$, as follows:

$$p_i = q_i(S_1)/m - \bar{\pi}_i.$$

Then a lower bound on $\bar{c}_{\min}$ is given by

$$\text{(LB)} \quad \min\ LB = c(S_1) - \sum_{i \in S_1} \bar{\pi}_i + \min \sum_{i \in S_x} x_i p_i$$

subject to

$$\sum_{i \in S_x} d_i x_i \leq \bar{C},$$
$$x_i \in \{0, 1\} \qquad \forall i \in S_x.$$

This last problem is a knapsack problem for which effective, although nonpolynomial, algorithms exist (see the excellent book by Martello and Toth [22]). In addition, since only a lower bound on $\bar{c}_{\min}$ is sought, it is not necessary to solve this knapsack problem as an integer program. Since the linear program is solved trivially, this is often a better approach in practice. In addition, rather than solving a TSP at each node (to evaluate $c(S_1)$), a lower bound can be computed based on (4.7).

To make the algorithm run more efficiently, Agarwal, Mathur, and Salkin implemented a number of additional features. To avoid excessive fluctuations of the value of the dual variables from iteration to iteration, Agarwal, Mathur, and Salkin imposed an additional set of constraints on the dual variables. They add to problem P_D the constraints

$$\pi_i \leq t, \qquad i \in V,$$

for some constant t. The value of t can be increased gradually toward the end of the algorithm to avoid any of these constraints being tight in the final linear programming solution. According to the authors, this technique substantially increased the convergence rate.

Similar approaches to (computationally) stabilize column generation procedures were applied by Du Merle et al. [17]. There, stronger primal and dual components were used, in particular, the perturbation of the right-hand side together with the introduction of the available or expected dual information. See Du Merle et al. [17] for details.

4.3.2 Polyhedral Branch-and-Bound Algorithm

Bixby, Coullard, and Simchi-Levi [6] developed a cutting plane algorithm for solving problem CG. To present their approach we first define a few terms. Let E denote the set of edges, and let (i, j) denote a particular edge. For any set of nodes $S \subseteq V^0$, let $\delta(S)$ denote those edges of E that have exactly one end in S. Below set $\bar{\pi}_0 = 0$. They consider the following integer programming formulation of problem CG:

$$\min \sum_{(i,j)\in E} t_{ij}x_{ij} - \sum_{i\in V^0} \bar{\pi}_i y_i$$

subject to

$$\sum_{(i,j)\in\delta(\{i\})} x_{ij} = 2y_i \qquad \forall\, i \in V^0, \tag{4.8}$$

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq -2 + 2y_k + 2y_\ell \qquad \forall\, S \subset V^0, k \in S, \ell \in V^0 \setminus S, \tag{4.9}$$

$$\sum_{i\in V} d_i y_i \leq C, \tag{4.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in E, \tag{4.11}$$

$$y_i \in \{0, 1\} \qquad \forall\, i \in V, \tag{4.12}$$

$$y_0 = 1, \tag{4.13}$$

where x_{ij} is 1 if edge (i, j) is in the tour and 0 otherwise, y_i is 1 if node i is in the tour and 0 otherwise. Note that, because of (4.11), routes with only one customer are not allowed.

Constraints (4.8) are the *assignment constraints*, requiring that for every node in the tour there are exactly two adjacent edges. Constraints (4.9) are the *subtour elimination constraints*, which ensure that for all sets S, such that both S and $V^0 \setminus S$ contain nodes in the tour, there have to be at least two edges in $\delta(S)$ included in the tour. So, (4.8), (4.9), and (4.13) along with the integrality conditions in (4.11) define the set of subtours or cycles through the depot node.

To solve this integer program, Bixby, Coullard, and Simchi-Levi [6] introduced a number of cuts, including the subtour elimination constraints (4.9) and the *two-matching constraints:*

$$\sum_{(i,j)\in\delta(H)\setminus T} x_{ij} + \sum_{(i,j)\in T} (1 - x_{ij}) \geq 1,$$

where $(H, V^0 \setminus H)$ is a partition of the nodes V^0 and T is a node disjoint subset of the edges in $\delta(H)$ with $|T| \geq 3$ and odd. These are exactly the same as the two-matching constraints for the TSP.

Another useful set of cuts for the subtour polytope are the *cocycle* or *cone inequalities*, as in Seymour [25]. (See also Bauer [4] on facets of the cycle polytope.) They can be stated as follows:

$$\sum_{(i,j)\in\delta(S)\setminus(k,\ell)} x_{ij} - x_{k\ell} \geq 0, \qquad S \subset V^0, k \in S, \ell \in V^0 \setminus S.$$

Finally, the authors used a type of cutting plane for the knapsack polytope based on minimal covers. A *minimal cover* is a subset $S \subseteq V$ for which $\sum_{i\in S} d_i > C$ and $\sum_{i\in S\setminus\{j\}} d_i \leq C$ for all nodes j in the set S. For each such set we have the following set of valid minimal cover constraints:

$$\sum_{i\in S} y_i \leq |S| - 1.$$

Some of these inequalities have been incorporated in a branch-and-cut algorithm. The algorithm was tested on several instances of problem CG, arising from CVRPs, with up to 51 customers. Some of these results are reported in Table 4.2. For further details, see Bixby, Coullard, and Simchi-Levi [6] and Bixby [5].

4.3.3 Pseudo-Polynomial Lower Bound on $\bar{c}_{\min}$

Desrochers, Desrosiers, and Solomon [14] devised a branch-and-bound algorithm to solve the column-generation problem and thus the linear programming relaxation of the set-covering model. They considered the VRPTW, but we describe here how this method can be applied to the CVRP. They generated a lower bound on $\bar{c}_{\min}$ using dynamic programming. Thus each calculation of this bound requires only pseudo-polynomial time. Further details on VRPTW may be found in Chapter 7.

To be able to solve CG using dynamic programming (with a state space of manageable size), we modify problem P to allow routes that visit the same customer more than once. The benefits of including this modification will be clear in a moment. Unfortunately, this method has the disadvantage of expanding the set of feasible routes. The model, call it problem P_m (where m stands for the "modified" formulation), is defined as follows. Enumerate all feasible routes, satisfying the capacity constraint, that may visit the same customer a number of times; each such visit increases the total load by the demand of that customer. Let the number of routes (columns) be $\mathcal{R}_m$, and let c_r be the total distance traveled in route r. For each customer $i \in V$ and route $r = 1, 2, \ldots, \mathcal{R}_m$, let

$$\xi_{ir} = \text{number of times customer } i \text{ is visited in route } r.$$

Also, for each $r = 1, 2, \ldots, \mathcal{R}_m$, define

$$y_r = \begin{cases} 1 & \text{if route } r \text{ is in the optimal solution,} \\ 0 & \text{otherwise.} \end{cases}$$

The CVRP can be formulated as

$$(\mathrm{P}_m) \quad \min \sum_{r=1}^{\mathcal{R}_m} c_r y_r$$

subject to

$$\sum_{r=1}^{\mathcal{R}_m} \xi_{ir} y_r \geq 1 \qquad \forall\, i \in V, \tag{4.14}$$

$$\sum_{r=1}^{\mathcal{R}_m} y_r \leq K, \tag{4.15}$$

$$y_r \in \{0, 1\} \qquad \forall\, r = 1, 2, \ldots, \mathcal{R}_m. \tag{4.16}$$

This is the set-covering problem solved by Desrochers, Desrosiers, and Solomon [14] in the context of the VRPTW. Clearly, the optimal integer solution to problem P_m is the optimal solution to the CVRP. However, the optimal solution values of the linear relaxations of problem P_m and problem P may be different. Of course, the linear relaxation of problem P_m provides a lower bound on the linear relaxation of problem P.

To solve the linear programming relaxation of problem P_m we use the method described above (for solving the linear programming relaxation of problem P). We enumerate a partial set $\mathcal{R}'_m$ of routes; solve problem P'_m, which is the linear relaxation of problem P_m defined only on this partial set of routes; and use the dual variables to see whether there exists a column not in the partial set with $\sum_{i=1}^{n} \xi_{ir} \bar{\pi}_i > c_r + \bar{\theta}$. If there exists such a column(s), we add it (them) to the formulation and solve the resulting linear program again. Otherwise, we have the optimal solution to the linear programming relaxation of problem P_m.

The modification we have made makes the column-generation step (the solution of CG) computationally easier, at the cost of only generating a lower bound. This can be done in pseudo-polynomial time using dynamic programming, as described next.

We need the following definitions. Given a path $\Pi = \{0, u_1, u_2, \ldots, u_\ell\}$, where it is possible that $u_i = u_j$ for $i \neq j$, the total load of this path is defined as $\sum_{i=1}^{\ell} d_{u_i}$. That is, the total load of the path is the sum, over all customers in Π, of the demand of a customer multiplied by the number of times that the customer appears in Π. Let $\{c_{ij} : i, j \in V^0\}$ denote a general distance measure between all pairs of nodes in V^0. In what follows we use $c_{ij} = t_{ij} - \bar{\pi}_j$ for all $i, j, \in V^0$, where $\bar{\pi}_0 \equiv 0$. Let $f_i(q)$ be the cost (evaluated using a distance measure $\{c_{ij}\}$) of the least-cost path that starts at the depot and terminates at customer i with a total load of q (this is called a q-path). This can be calculated using the following recursion:

$$f_i(q) = \min_{j \neq i} \left\{ f_j(q - d_i) + c_{ji} \right\} \tag{4.17}$$

with the initial conditions

$$f_i(q) = \begin{cases} c_{0i} & \text{if } q = d_i, \\ +\infty & \text{otherwise.} \end{cases}$$

Finally, let $f_i^0(q) = f_i(q) + c_{i0}$. Thus, $f_i^0(q)$ is the minimum reduced cost of a tour that starts at the depot, visits a subset of the customers, of which customer i is the last to be visited, and terminates at the depot with total load q. Observe that finding $f_i^0(q)$ for every q, $1 \le q \le C$, every i, $i \in V$, requires $O(n^2C)$ calculations.

The recursion chooses the predecessor of i to be a node $j \neq i$. This requires repeat visits to the same customer to be separated by at least one visit to another customer. In fact, expanding the state space of this recursion can eliminate 2-*loops*: loops of the type $\ldots i, j, i \ldots$. This forces repeat visits to the same customer to be separated by visits to at least two other customers. According to the approach proposed by Christofides, Mingozzi, and Toth [11], this is done as follows. Let $p_i(q)$ denote the predecessor of i in the path of cost $f_i(q)$ for $i \in V$ and $1 \le q \le C$. Then define $g_i(q)$ as the cost of the least-cost path from the depot to customer $i \in V$ with a total load of q and not having $p_i(q)$ as the last customer visited before i. Then, we have (for all $i \in V$ and $1 \le q \le C$)

$$f_i(q) = \min_{j \neq i} \left\{ \left[f_j(q - d_i) + c_{ji} : i \neq p_j(q - d_i) \right], \left[g_j(q - d_i) + c_{ji} : i = p_j(q) \right] \right\}$$

and

$$g_i(q) = \min_{j \neq i} \left\{ \left[f_j(q - d_i) + c_{ji} : i \neq p_j(q - d_i) \text{ and } j \neq p_i(q) \right], \right.$$
$$\left. \left[g_j(q - d_i) + c_{ji} : i = p_j(q - d_i) \text{ and } j \neq p_i(q) \right] \right\}.$$

Finally, let $f_i^0(q) = f_i(q) + c_{i0}$ and $g_i^0(q) = g_i(q) + c_{i0}$. Note that $f_i(q) \le g_i(q)$ for all $i \in V$ and $1 \le q \le C$. This dynamic program can lead to a stronger relaxation of the set-covering model with little extra computational effort. For a more detailed discussion of this recursion and methods of efficient implementation, see Christofides, Mingozzi, and Toth [11, 12] or Desrochers, Desrosiers, and Solomon [14].

The algorithm proceeds as follows. If there exists a q, $1 \le q \le C$, and an $i \in V$ with $f_i^0(q) < 0$, then we add the corresponding column to the set of columns in problem P'_m. If, on the other hand, $f_i^0(q) \ge 0$ for every q and i, then the current solution is the optimal linear programming solution to P_m.

4.3.4 Solving P_D via Dual-Ascent and Branch-and-Bound

A different approach to this same set-covering model was developed by Hadjiconstantinou, Christofides, and Mingozzi [18]. Instead of attacking the primal problem P, they devised a branch-and-bound algorithm to solve the dual problem P_D. The problem is solved using a dynamic programming heuristic in conjunction with a Lagrangian ascent procedure. This produces strong lower bounds on the optimal solution value of the CVRP (the primal problem) which are used in a branch-and-bound framework.

We first describe the lower bounding procedure. Let $\mathcal{R}_i \subset \mathcal{R}$ denote all feasible routes that visit customer $i \in V$. For all $r \in \mathcal{R}$, let $C_r = \sum_{i \in S_r} d_i$ denote the total load on route r.

THEOREM 4.3.1. *Let $\underline{c}_r$ denote a lower bound on route $r \in \mathcal{R}$. Then a lower bound on the optimal solution value to the CVRP is given by*

$$LB = \sum_{i \in V} d_i \min_{r \in \mathcal{R}_i} \left\lceil \frac{\underline{c}_r}{C_r} \right\rceil.$$

Proof. We define the following feasible dual solution $(u_1, u_2, \ldots, u_n, v)$ where $\{u_i\}_{i \in V}$ are the dual variable associated with constraints (4.3) and v is the dual variable associated with constraint (4.4).

For each $i \in V$, let

$$u_i = d_i \min_{r \in \mathcal{R}_i} \left\lceil \frac{\underline{c}_r}{C_r} \right\rceil$$

and let $v = 0$. We show that $(u_1, u_2, \ldots, u_n, v)$ is feasible for P_D. This is clear since, for any route $r \in \mathcal{R}$, we have

$$c_r = \sum_{i \in S_r} d_i \frac{c_r}{C_r} \geq \sum_{i \in S_r} d_i \min_{\ell \in \mathcal{R}_i} \frac{c_\ell}{C_\ell} \geq \sum_{i \in S_r} d_i \min_{\ell \in \mathcal{R}_i} \frac{\underline{c}_\ell}{C_\ell} = \sum_{i \in S_r} u_i + v. \qquad \square$$

For each $i \in V$ and each $d_i \leq q \leq C$ define the set $\mathcal{R}_i(q)$ as those routes of $\mathcal{R}_i$ that have total load exactly q. Then the set $\mathcal{R}_i$ can be decomposed as follows:

$$\mathcal{R}_i = \mathcal{R}_i(d_i) \cup \mathcal{R}_i(d_i + 1) \cup \cdots \cup \mathcal{R}_i(C).$$

It is clear that

$$\min_{r \in \mathcal{R}_i} \left\lceil \frac{\underline{c}_r}{C_r} \right\rceil = \min_{d_i \leq q \leq C} \left\{ \min_{r \in \mathcal{R}_i(q)} \left\lceil \frac{\underline{c}_r}{q} \right\rceil \right\}.$$

If we denote $\underline{c}_{iq} = \min_{r \in \mathcal{R}_i(q)} \{\underline{c}_r\}$, then the lower bound can be rewritten

$$LB = \sum_{i \in V} d_i \min_{d_i \leq q \leq C} \left\lceil \frac{\underline{c}_{iq}}{q} \right\rceil. \tag{4.18}$$

4.3.4.1 Computation of Bounds

Several lower bounds $(\underline{c}_{iq})$ can be computed to evaluate (4.18).

The bounds calculated in section 4.3.3, namely, $f_i(q)$ (evaluated using $c_{ij} = t_{ij}$, $i, j \in V^0$) for each $i \in V$ and $d_i \leq q \leq C$ can be used in (4.18) since $f_i(q) \leq c_r$ for all $r \in \mathcal{R}_i(q)$.

Building on the definitions of $f_i(q)$, $g_i(q)$, and $p_i(q)$ in section 4.3.3 (evaluated with $c_{ij} = t_{ij}$ for all $i, j \in V^0$) we can determine another bound. For this purpose, define $\psi_i(q)$ as the cost of the least-cost route without 2-loops, starting at the depot, passing through customer i, and ending at the depot with total load q. Such a route is called a *through q-route*. The route defining $\psi_i(q)$ can be calculated as follows:

$$\psi_i(q) = \min_{d_i \leq q' \leq \frac{q+d_i}{2}} \begin{cases} f_i(q') + f_i(q + d_i - q') & \text{if } p_i(q) \neq p_i(q + d_i - q'), \\ \min\{f_i(q') + g_i(q + d_i - q'), & \\ \quad g_i(q') + f_i(q + d_i - q')\} & \text{otherwise.} \end{cases}$$

Now $\psi_i(q)$ can be used as a lower bound on $\underline{c}_{iq}$ in (4.18). Formally, the lower bound $LB1$ is

$$LB1 = \sum_{i \in V} d_i \cdot \min_{d_i \le q \le C} \left[\frac{\psi_i(q)}{q} \right]. \tag{4.19}$$

4.3.4.2 Ascent Procedure

The lower bound $LB1$ can be improved by observing the following. Consider each of the n routes, one for each $i \in V$, achieving the minimums in (4.19). Define d_i^* so that the route corresponding to $i \in V$ has cost $\psi_i(d_i^*)$. Let this route be r_i^*. If we superimpose these routes, the degree of each node will be even (and at least two). Let δ_i^j denote the degree of customer i with respect to route r_j^*. Then calculate the *weighted degree* of node i as

$$D_i = \sum_{j \in V} \delta_i^j \cdot \frac{d_j}{d_j^*}.$$

If $D_i = 2$ for all $i \in V$, then the solution represented by $\cup_i r_i^*$ is a feasible solution to the CVRP; otherwise, we can apply the following penalty procedure to improve the bound $LB1$. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ denote penalties on the nodes. Redefine the cost matrix so that $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$ for each $i, j \in V^0$. The functions $f_i(q)$, $g_i(q)$, and $\psi_i(q)$ can be recalculated using c'_{ij}, resulting in a new bound $LB1(\lambda)$. The optimal CVRP solution is unaffected by this penalty vector, since it adds only a constant term $2\sum_{i \in V} \lambda_i$ to the objective. Now $LB1(\lambda)$ can be maximized using a standard subgradient optimization procedure. See Hadjiconstantinou, Christofides, and Mingozzi [18] for details. At the conclusion of this subgradient procedure, a feasible dual solution with value close to the optimal is generated; this is a lower bound on the optimal solution value of the CVRP.

Hadjiconstantinou, Christofides, and Mingozzi [18] developed another lower bounding method based on determining the k-shortest paths between the depot and node $i \in V$. These two bounds are used to bound the objective function in a branch-and-bound tree. For details see Hadjiconstantinou, Christofides, and Mingozzi [18].

4.4 Solving the Set-Covering Integer Program

In the previous section we introduced several methods for solving a linear relaxation of the set-covering formulation of the CVRP, problem P, or in the case of section 4.3.4, the dual P_D. The linear relaxation of problem P is likely to be fractional and therefore there is still the problem of finding an integer solution. In this section, we describe a number of ways to use the current set of columns to generate an optimal or a near-optimal integer solution.

We consider two general approaches and then give some additional comments on various computational aspects of this problem. The first, a cutting plane method, does not generate any additional columns from this point on. It therefore solves the integer program defined only on the current set of columns. This method is not guaranteed to generate the optimal solution to P since there may be columns in the optimal integer solution that have not yet been generated. In practice, this solution is probably close to the optimal one, and in any case, any integer solution will come with a worst-case bound on its relative

error (because of the lower bound provided by the optimal solution value of the linear programming relaxation). The second method, branch-and-price, generates new columns during its search and therefore solves problem P exactly (this is the method of Desrochers, Desrosiers, and Solomon [14]). That is, it solves the integer program over the entire set of columns $\mathcal{R}$ without necessarily generating all of the feasible routes in the set. Branch-and-price is generally more complicated since it may require incorporating information about the branch-and-bound search within the column generation problem.

If an upper bound on the optimal integer solution value is known, then the following preprocessing step should be performed. Let Z_{UB} denote the value of the upper bound, and let Z_{LB} denote the value of a lower bound (either Z^{LP} or a lower bound on it). Then any column (route) with reduced cost not smaller than $Z_{\mathrm{UB}} - Z_{\mathrm{LB}}$ can be immediately eliminated from the model.

4.4.1 A Cutting Plane Method

A computationally attractive approach for solving the CVRP using only the current set of columns is a method called the *cutting plane approach*. Given a fractional solution to P, we can generate a set of constraints that will cut off this fractional solution. After adding these constraints to the formulation, we can resolve the linear program, and if it is integer, we have found the optimal integer solution (among the columns $\mathcal{R}'$). If it is still fractional, then we continue generating constraints and resolving the linear program until an integer solution is possibly found. Additionally, one can implement this strategy within a branch-and-bound framework. Typically, this is called branch-and-cut. This method was successfully used by Padberg and Rinaldi [23] to solve the TSP and by Hoffman and Padberg [21] to solve crew-scheduling problems.

Again, the best integer solution found using this method should be close to optimal, and in any case a bound on the relative error is readily obtainable.

Cutting Plane Algorithm

Step 1. Generate an initial set $\mathcal{R}'$ of columns.

Step 2. Solve, using column generation, problem P′ (i.e., the linear programming relaxation of P).

Step 3. If the optimal solution to problem P′ is integer, stop.
Else, generate cutting planes separating this fractional solution.
Add these cutting planes to the linear program P′.

Step 4. Solve the linear program P′. Goto Step 3.

The key to success of this method is to be able to efficiently generate constraints that will separate a fractional solution from all integer solutions (Step 3). We describe two specific kinds of constraints and demonstrate how they can be efficiently identified. Let $\mathcal{R}'$ be the set of routes at the end of the column generation procedure. To generate constraints, construct the *intersection graph* G. The graph G has a node for each column in $\mathcal{R}'$. Two nodes in G are connected by an undirected edge if the corresponding columns have at least one customer in common. Observe that a solution to the CVRP where no customer is visited more than once can be represented by an *independent set* in this graph. An independent set is a collection of nodes of G such that no two nodes are connected by an edge.

This observation gives rise to two simple inequalities that can be added to the formulation. In what follows, let $\bar{y}$ denote the (fractional) optimal solution to the current formulation.

4.4.1.1 Clique Constraints

Select a subset of the nodes of G, say, H, such that every pair of nodes i, $j \in H$ is connected by an edge of G. Each set H, called a clique, must satisfy the following condition:

$$\sum_{r \in H} \bar{y}_r \leq 1. \tag{4.20}$$

Clearly, if there is a node $j \notin H$ such that j is adjacent to every $i \in H$, then we can replace H with $H \cup \{j\}$ in inequality (4.20) to strengthen it (this is called *lifting*). In that sense, we would like to use inequality (4.20) when the set of nodes H is maximal.

Hoffman and Padberg [21] suggested several procedures for clique identification, one of which is based on the fact that small-size cliques can be found quickly by enumeration. For this purpose, select v to be the node with minimum degree among all nodes of G. Clearly, every clique of G containing v is a subset of the neighbors of v, denoted by $T(v)$. Thus, starting with v as the current clique, that is, $H = \{v\}$, we add an arbitrary node w from $T(v)$ to H. We now delete from $T(v)$ all nodes that are not connected to w. Continue adding nodes in this manner from the current set $T(v)$ to H until either there is no node in $T(v)$ connected to all nodes in H, or $T(v) = \emptyset$. In the end, H will be a maximal clique. We then can calculate the *weight* of this clique, that is, the sum of the values $\bar{y}_r$ of the columns in the clique. If the weight is more than 1, then the corresponding clique inequality is violated. If not, then we continue the procedure with a new starting node. The method can be improved computationally by, for example, always choosing the heaviest node (the one where $\bar{y}_r$ is the largest) among those nodes eligible to enter the clique.

4.4.1.2 Odd Hole Constraints

Define a cycle $H = \{u_1, u_2, \ldots, u_\ell\}$ in G, such that node u_i is adjacent to u_{i+1}, for each $i = 1, 2, \ldots, \ell - 1$, and node u_ℓ is adjacent to node u_1. A cycle H is called an *odd cycle* if the number of nodes in H, $|H| = \ell$, is odd. An odd cycle is called an *odd hole* if there is no edge connecting two nodes of the cycle except the ℓ edges defining the cycle. It is easy to see that in any optimal solution to the CVRP each odd hole must satisfy the following property:

$$\sum_{r \in H} \bar{y}_r \leq \frac{|H| - 1}{2}. \tag{4.21}$$

Hoffman and Padberg used the following procedure to identify violated odd hole constraints. Starting from an arbitrary node $v \in G$, construct a layered graph $G_\ell(v)$ as follows. The node set of $G_\ell(v)$ is the same as the node set of G. Every neighbor of v in G is connected to v by an edge in $G_\ell(v)$. We refer to v as the root, or level 0 node, and we refer to the neighbors of v as level 1 nodes. Similarly, nodes at level $k \geq 2$ are those nodes in G that are connected (in G) to a level $k - 1$ node but are not connected to any node at level $<$

$k-1$. Finally, each edge (u_i, u_j) in $G_\ell(v)$ is assigned a length of $1 - \bar{y}_{u_i} - \bar{y}_{u_j} \geq 0$. Now pick a node u in $G_\ell(v)$ at level $k \geq 2$ and find the shortest path from u to v in $G_\ell(v)$. Delete all nodes at levels i $(1 \leq i < k)$ that are either on the shortest path or adjacent to nodes along this shortest path (other than nodes that are adjacent to v). Now pick another node w that is adjacent (in G) to u in level k. Find the shortest path from w to v in the current graph $G_\ell(v)$. Combining these two paths with the edge (u, w) creates an odd hole. If the total length of this cycle is less than 1, then we have found a violated odd hole inequality. If not, we continue with another neighbor of u and repeat the process. We can then choose a node different from u at level k. If no violated odd hole inequality is found at level k, we proceed to level $k+1$. This subroutine can be repeated for different starting nodes (v) as well.

4.4.1.3 Branching Strategies

Another method with which to find the best integer solution among the set of columns $\mathcal{R}'$ is the branch-and-cut method. This method consists of splitting the problem into easier subproblems by fixing the value of a certain branching variable. In this case, a suitable choice is $\bar{y}_r$ for some route r. The branching variable is set to 1 in one branch of the tree and 0 in the other. To each of these subproblems, independently, are added a series of constraints (cuts) strengthening the linear programming formulation. These constraints can be along the same lines as those discussed in section 4.4.1.

Exact branching and cutting strategies together with a column generation scheme are easily defined if the CVRP (or VRPTW) is first modeled as a multicommodity flow problem and then decomposed using the Dantzig–Wolfe approach. Therefore, many decisions can be taken on the flow variables or on a combination of these. See Chapters 3 and 7, where many suggestions are provided.

4.4.2 Branch-and-Price

The methods described in the previous section enable us to solve the CVRP on the restricted set of columns: those generated in the process of solving the linear programming relaxation of P. If true optimality of the integer solution is sought (as opposed to a solution that might be very close to optimal), then solving the integer program over all columns of $\mathcal{R}$ is much more difficult.

In this case, we describe a branch-and-bound procedure where additional columns are generated at each node of the branch-and-bound tree. We describe here the approach of Desrochers, Desrosiers, and Solomon [14] for the VRPTW. The main difficulty with the approach described here is that it must be possible to incorporate information about the node of the branch-and-bound tree in the column-generation procedure. For instance, assume we branch on variables y_r as described in the previous section. It is simple to incorporate the information that $y_r = 1$ for a particular route r (in one branch of the tree) in the column generation procedure. This is done by simply omitting the nodes of S_r from the column generation procedure. However, it is not clear how to incorporate the information that $y_r = 0$ into the dynamic programming procedure. Typically, an approach based on expanding the state space will not be computationally attractive. Therefore Desrochers, Desrosiers, and Solomon do not branch on these variables.

Desrochers, Desrosiers, and Solomon branch on the edge variables, i.e., whether an edge (i, j) is used or not used by some route in the integer optimal solution. If we signify this branching variable by $x_{ij} = 0$ or 1, let us consider how this affects the column generation problem CG in each branch of the tree. The information in the branch with $x_{ij} = 1$ can be incorporated into the column-generation step by setting $c_{ij} = -\infty$ in the dynamic program described in section 4.3.3, forcing the minimum cost route to use edge (i, j). For the other branch ($x_{ij} = 0$) we set $c_{ij} = +\infty$ so that the edge (i, j) is never used in a generated route.

4.4.3 Additional Comments on Computational Approaches

We note that developing a successful algorithm to solve a set-covering problem using column generation requires quite a bit of computational testing. In particular, there are a number of tricks to reduce computational time and manage the computer's memory (e.g., generating new columns, throwing away columns that have not been basic in a number of iterations).

In general, to get good integer solutions, it is more important to generate new columns, even heuristically, at all or several nodes of the branch-and-bound tree, than to spend time designing complex branching and cutting plane strategies. If optimal or near-optimal compatible columns are not present, it is useless to work hard on these strategies (even if the lower bound, computed by using only the columns selected in the optimal basis, is very good).

Finally, we suggest another computational trick to more effectively generate columns that are disjoint, collectively exhaustive, and of minimal cost. Cutting planes are used only temporarily to fix columns at value 1. Each time this happens, already-generated cutting planes are removed and new columns are generated on the residual problem (the problem consisting of the customers not served by routes fixed to 1). The lower bound on this residual problem might improve, but much more important, the "missing" columns may now appear to complete the big puzzle into an integer solution.

4.5 Computational Results

We report here some computational results on each of the approaches we have described. The results for the CVRP of Agarwal, Mathur, and Salkin [1], Bixby [5], and Hadjiconstantinou, Christofides, and Mingozzi [18] are on the standard test problems of Christofides, Mingozzi, and Toth [11]. The results of Desrochers, Desrosiers, and Solomon [14] on the VRPTW are on the standard test problems of Solomon [26]. (For further details and results on VRPTW, see Chapter 7.)

In Tables 4.1, 4.2, and 4.3, we list the problem name and number of customers, the value of the lower bound (Z^{LP}), and the value of the upper bound Z^{UB} provided by the particular method used in the paper. The optimal integer solution to the routing problem is denoted Z^* where applicable. The effectiveness of the lower bound is therefore defined as $100(Z^{\mathrm{LB}}/Z^{\mathrm{UB}})$ or $100(Z^{\mathrm{LB}}/Z^*)$, depending on whether an optimal solution to the problem is known.

As one can see, almost uniformly across all cases, the lower bound provided by the linear programming relaxation of the set-covering formulation is "very strong." In addition,

Table 4.1. *Results of Agarwal, Mathur, and Salkin* [1].

Problem	n	Z^{LP}	Z^{UB}	Effectiveness of lower bound
E016-03m	15	268	276	97.1%
E016-05m	15	326	332	98.2%
E021-04m	20	351	358	98.0%
E021-06m	20	430	430	100.0%
E022-04g	21	374	375	99.7%
E022-06m	21	479	494	97.0%
E026-08m	25	606	607	99.8%

Table 4.2. *Results of Bixby* [5].

Problem	n	Z^{LP}	Z^{UB}	Effectiveness of lower bound
S007-02a	6	114	114	100.0%
S013-04d	12	279	290	96.2%
E021-06m	20	430	430	100.0%
E022-04g	21	375	375	100.0%
E023-03g	22	566	569	99.5%
E030-04s	29	503	503	100.0%
E051-05e	50	518	521	99.4%

Table 4.3. *Results of Hadjiconstantinou, Christofides, and Mingozzi* [18].

Problem	n	Z^{LB}	Best available upper bound	Effectiveness of lower bound
E016-05m	15	326.92	334.96	97.6%
E021-06m	20	430.88	430.88	100.0%
E026-08m	25	621.73	621.73	100.0%
E031-09h	30	597.18	610.10	97.9%
E036-11h	35	694.89	698.60	99.5%
E041-14h	40	852.24	861.79	98.9%
E051-05e	50	516.51	524.61	98.5%
E076-10e	75	815.31	835.26	97.6%
E101-08e	100	792.42	826.14	95.9%
E151-12c	150	1000.07	1028.42	97.2%

Note: The lower bound here is not Z^{LP}.

Table 4.4. *A Sample of Results from Desrochers, Desrosiers, and Solomon* [14].

Problem	n	Z^{LP}	Z^*	Effectiveness of lower bound
R103	25	454.6	454.6	100.0%
R107	50	703.2	711.1	98.9%
R108	25	396.2	397.2	99.8%
R110	50	692.4	697.0	99.3%
C101	100	827.3	827.3	100.0%
C103	50	361.4	361.4	100.0%
C106	100	827.3	827.3	100.0%
C107	100	827.3	827.3	100.0%
RC103	25	332.1	332.8	99.8%
RC104	25	305.9	306.6	99.8%
RC105	25	411.0	411.3	99.9%
RC108	25	280.3	294.5	95.2%

the upper bounds are also very close to the lower bound and therefore very close to the optimal value.

The problems given in Table 4.4 are for a randomly selected sample of the problems that were solved to optimality by Desrochers, Desrosiers, and Solomon [14]. This list therefore represents those problems that are more likely to have an effective lower bound. However, it is clear that the lower bound is most likely very strong for a large class of problems. In the next section, we consider the theoretical question of why the bound is so effective.

4.6 Effectiveness of the Set-Covering Formulation

We now analyze the strength of the linear programming relaxation of problem P. The effectiveness of the above approaches depends critically on the so-called *integrality gap*: the difference between the values of the optimal integer solution and the optimal solution to the linear relaxation of problem P. If the lower bound provided by the linear programming relaxation is not very tight (i.e., the gap is large), then the methods described most likely will not be computationally effective. On the other hand, when the gap is small, the procedures are likely to be effective.

Fortunately, many researchers have reported that the linear relaxation of the set-covering problem P provides an optimal solution value very close to the optimal integer solution value. Evidence of this can be found in Desrochers, Desrosiers, and Solomon [14] for the case of the VRPTW and Hoffman and Padberg [21] for crew-scheduling problems. That is, the solution to the linear relaxation of problem P provides a very tight lower bound on the integer programming solution value. For instance, in their paper, Desrochers, Desrosiers, and Solomon reported an average relative gap between the optimal solution value to the linear relaxation and the optimal integer solution value of only 0.733%.

We cite results concerning the gap's size measured by using both worst-case and average-case criteria. In particular, the average-case analysis shows that, asymptotically,

the relative error between the optimal solution value to the linear relaxation of P and the optimal integer solution value tends to zero as the number of customers increases.

4.6.1 Worst-Case Analysis

It is interesting to characterize the largest possible value of the ratio Z^*/Z^{LP}. A simple bound can be constructed using the Iterated Tour Partitioning (ITP) heuristic (see Haimovich and Rinnooy Kan [19] or Altinkemer and Gavish [2]). For the equal demand case ($d_i = 1$ for all $i \in V$), we get (see Altinkemer and Gavish [2])

$$Z^* \leq Z^{\mathrm{ITP}(1)} \leq \frac{2}{C}\sum_{i\in V} t_{0i} + \Big(1 - \frac{1}{C}\Big)L^*(V^0),$$

where $L^*(V^0)$ is the length of the optimal traveling salesman tour through V and the depot. It is easy to show that $Z^{\mathrm{LP}} \geq 2\sum_{i\in V} t_{0i}/C$. Using Held and Karp's lower bound [20], one can show that

$$L^*(V^0) \leq \frac{3}{2}Z^{\mathrm{LP}},$$

thus giving the 2.5 bound. For the general demand case, following this same line of reasoning it is possible to show that $Z^*/Z^{\mathrm{LP}} \leq 3.5$. It is not known if these bounds are tight.

Worst-case analyses of the set-covering model for a special case of the CVRP were performed in Chan, Simchi-Levi, and Bramel [10]. They looked at the Bin Packing Problem (BPP), which can be viewed as a CVRP where all the customers are at the same location at a fixed (nonzero) distance from the depot. Chan, Simchi-Levi, and Bramel showed that for the BPP, $Z^*/\lceil Z^{\mathrm{LP}}\rceil \leq 4/3$, and they provided an example achieving this bound. Therefore, if this special case is any indication, the lower bound provided by the optimal solution to the linear programming relaxation of the set-covering problem is strong indeed. That is, the lower bound is at least 75% of the value of the optimal integer solution.

4.6.2 Average-Case Analysis

We now present a probabilistic analysis of this model. A similar analysis was performed for the VRPTW, resulting in similar conclusions, by Bramel and Simchi-Levi [8]. Here we perform this same analysis for the CVRP.

To present the analysis, we assume the customers are dispersed in the Euclidean plane, specifically, customer $i \in V$ is located at $x_i \in \mathbb{R}^2$. We assume, without loss of generality, that the depot is at the origin, and we denote by $\|x\|$ the Euclidean distance between point $x \in \mathbb{R}^2$ and the depot. We also scale the vehicle capacity to 1 and therefore assume $d_i \in (0, 1]$ for each $i \in V$. We assume, for the purposes of simplifying the analysis, that the fleet size is not limited.

Consider the n customer locations to be independently distributed according to a distribution μ with compact support in $\mathbb{R}^2$. Let the customer demands $\{d_i : i \in V\}$ be drawn from a distribution Φ with density ϕ which is assumed to be Lipschitz continuous of order $q \geq 1$ on $[0, 1]$. (For $x \notin [0, 1]$, $\phi(x) = 0$.) A function ϕ is Lipschitz continuous of order q on S if there exists an H such that

$$|\phi(x) - \phi(y)| \leq H|x - y|^q, \qquad x, y \in S.$$

This implies in particular the existence of a constant H_0 such that $\phi(x) \leq H_0$ for all $x \in [0, 1]$. Finally, we assume that a customer's location and its load are independent of each other.

THEOREM 4.6.1. *Let the customer locations $x_1, x_2, \ldots, x_n$ be a sequence of independent random variables having a distribution μ with compact support in $\mathbb{R}^2$. Let the customer demands be independently and identically distributed like Φ. Let Z^{LP} be the value of the optimal fractional solution to P, and let Z^* be the value of the optimal integer solution to P, that is, the value of the optimal solution to the CVRP. Then*

$$\lim_{n\to\infty} \frac{1}{n} Z^{\mathrm{LP}} = \lim_{n\to\infty} \frac{1}{n} Z^* \quad \textit{almost surely.}$$

It is interesting to note that the value of these limits is also known. Bramel et al. [7] showed that as the number of customers increases, the quantity Z^*/n tends almost surely to $2\gamma \mathrm{E}[d]$, where $\mathrm{E}[d]$ is simply the customer's expected distance to the depot and γ is the bin packing constant associated with Φ. The bin packing constant is defined as follows. Let b_n^* denote the number of bins required to pack n items drawn from Φ. Then $\gamma = \lim_{n\to\infty} b_n^*/n$, and note that $\gamma \in [0, 1]$. The value $1/\gamma$ can be interpreted as the asymptotic average number of items per bin in an optimal solution.

The result described in Theorem 4.6.1 says that almost surely $\frac{1}{n}(Z^* - Z^{\mathrm{LP}}) \to 0$ as $n \to \infty$. It is also important in results of this type to characterize the rate of convergence of this quantity to zero.

4.6.2.1 Motivation

We do not present a proof of Theorem 4.6.1. For that, we refer the reader to Bramel and Simchi-Levi [9]. However, we do provide a simplified analysis that gives some insight into why Theorem 4.6.1 holds. To do this we consider a simpler discrete vehicle routing model, defined as follows. Define a customer type to be a location $x \in \mathbb{R}^2$ and demand $w \in [0, 1]$. That is, two customers of the same type are located at the same location and have identical customer demands. Consider a *discretized* vehicle routing model in which there is a finite number, W, of possible customer types. In a particular instance, let n_i be the number of customers of type i for $i = 1, 2, \ldots, W$, and let $n = \sum_{i=1}^{W} n_i$ be the total number of customers. Clearly, this discretized CVRP can be solved by formulating it as a set covering problem.

Let a *vehicle assignment* be a vector $(a_1, a_2, \ldots, a_W)$, where $a_i \geq 0$ are integers, such that a single vehicle can feasibly serve a_i customers of type i for each $i = 1, 2, \ldots, W$, without violating the capacity constraint. Index all the possible vehicle assignments $1, 2, \ldots, R'$ and let c_r be the total length of the shortest feasible route serving the customers in vehicle assignment r. (Note R' is independent of n.) The CVRP can be formulated as follows. Let

$$a_{ir} = \text{number of customers of type } i \text{ in vehicle assignment } r$$

for each $i = 1, 2, \ldots, W$ and $r = 1, 2, \ldots, R'$. Let

$$y_r = \text{number of times vehicle assignment } r \text{ is used in the optimal solution.}$$

Then problem P_d is

$$(P_d) \quad \min \sum_{r=1}^{R'} c_r y_r$$

subject to

$$\sum_{r=1}^{R'} a_{ir} y_r \geq n_i, \qquad i = 1, 2, \ldots, W,$$
$$y_r \in \{0, 1\}, \qquad r = 1, 2, \ldots, R'.$$

Let Z_d^* be the optimal solution value of P_d and let Z_d^{LP} be the optimal solution value to its linear relaxation. Clearly, we can also formulate this discrete problem as an instance of problem P. If we compare the solution to P_d and to P we see that problems P and P_d must have the same optimal solution values, i.e., $Z^* = Z_d^*$. Observe that a feasible solution to the linear programming relaxation of P can be used to construct a feasible solution to the linear programming relaxation of P_d, and therefore

$$Z_d^{LP} \leq Z^{LP}. \tag{4.22}$$

Define $\bar{c} = \max_{r=1,2,\ldots,R'}\{c_r\}$, i.e., $\bar{c}$ is the length of the longest route among the R' vehicle assignments. Then, we have the following lemma.

Lemma 4.6.2.

$$Z^{LP} \leq Z^* \leq Z_d^{LP} + W\bar{c} \leq Z^{LP} + W\bar{c}.$$

Proof. The left-most inequality is trivial while the right-most inequality is due to (4.22). To prove the central inequality, note that in P_d there are W constraints (one for each customer type). Let $\bar{y}_r$ for $r = 1, 2, \ldots, R'$ be an optimal solution to the linear programming relaxation of P_d and observe that there exists such an optimal solution with at most W positive variables, one for each constraint. We construct a feasible solution to P_d by rounding the linear programming solution up; that is, for each $r = 1, 2, \ldots, R'$ with $\bar{y}_r > 0$ we make $y_r = 1$ and for each $r = 1, 2, \ldots, R'$ with $\bar{y}_r = 0$ we make $y_r = 0$. The increase in the objective function is therefore at most W times the largest possible cost of a route, $\bar{c}$. □

Observe that the upper bound on Z^* obtained in Lemma 4.6.2 consists of two terms. The first, Z^{LP}, is a lower bound on Z^*, which clearly grows with the number of customers, n. The second term $(W\bar{c})$ is the product of two numbers that are fixed and independent of n. Therefore, the upper bound on Z^* of Lemma 4.6.2 is dominated by Z^{LP}, and consequently we see that for large n, $Z^* \approx Z^{LP}$, exactly what is implied by Theorem 4.6.1. Indeed, much of the proof of Theorem 4.6.1 is concerned with approximating the distributions μ and Φ with discrete distributions and forcing the number of different customer types to be independent of n.

We now outline the main steps in the proof of Theorem 4.6.1. It is clear that $Z^{LP} \leq Z^*$ and therefore, almost surely, $\underline{\lim}_{n\to\infty} \frac{1}{n}(Z^* - Z^{LP}) \geq 0$. The interesting part is to find an upper bound on Z^* that involves Z^{LP} and shows that $\overline{\lim}_{n\to\infty} \frac{1}{n}(Z^* - Z^{LP}) \leq 0$, almost

surely. We do this in essentially the same way as before. To mimic that approach, we introduce a series of discretizations of the customer parameter distributions. We discretize the customer locations using a grid of squares. Each customer is then moved to the center of the square in which it is located. We do the same with the customer demands: we select a unit and round each customer demand to a multiple of this unit. The proof then proceeds to show the following. For the purposes of this discussion, let $\hat{Z}^{\mathrm{LP}}$ and $\hat{Z}^{*}$ denote the optimal linear relaxation value and the optimal integer solution value, respectively, of the set-covering formulation of the discretized vehicle routing problem. Under specific rounding schemes, as the discretization becomes finer,

- the relative difference between Z^{*} and $\hat{Z}^{*}$ decreases,
- the relative difference between Z^{LP} and $\hat{Z}^{\mathrm{LP}}$ decreases, and
- the relative difference between $\hat{Z}^{*}$ and $\hat{Z}^{\mathrm{LP}}$ decreases (as in the motivation above).

One can see then how the result follows from these points. Proving these results is rather involved and we therefore do not go through the details here (the interested reader can see Bramel and Simchi-Levi [9]). We note that a byproduct of the analysis is a bound on the rate of convergence which is $O(n^{4/5})$. That is, $\mathrm{E}[Z^{*}] = \mathrm{E}[Z^{\mathrm{LP}}] + O(n^{4/5})$.

Acknowledgments

This research was supported in part by ONR contracts N00014-90-J-1649 and N00014-95-1-0232 and NSF contracts DDM-9322828 and DMI-9732795.

Bibliography

[1] Y. Agarwal, K. Mathur, and H.M. Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19:731–749, 1989.

[2] K. Altinkemer and B. Gavish. Heuristics for delivery problems with constant error guarantees. *Transportation Science*, 24:294–297, 1991.

[3] M. Balinski and R. Quandt. On an integer program for a delivery problem. *Operations Research*, 12:300–304, 1964.

[4] P. Bauer. The circuit polytope: Facets. *Methods of Operations Research*, 22:110–145, 1996.

[5] A. Bixby. Polyhedral analysis and effective algorithms for the capacitated vehicle routing problem. Ph.D. dissertation, Northwestern University, Evanston, IL, 1998.

[6] A. Bixby, C. Coullard, and D. Simchi-Levi. The capacitated prize-collecting traveling salesman problem. Working paper, Department of Industrial Engineering and Engineering Management, Northwestern University, Evanston, IL, 1997.

[7] J. Bramel, E.G. Coffman, Jr., P. Shor, and D. Simchi-Levi. Probabilistic analysis of algorithms for the capacitated vehicle routing problem with unsplit demands. *Operations Research*, 40:1095–1106, 1991.

[8] J. Bramel and D. Simchi-Levi. Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows. *Operations Research*, 44:501–509, 1996.

[9] J. Bramel and D. Simchi-Levi. *The Logic of Logistics*. Springer-Verlag, New York, 1998.

[10] L.M.A. Chan, D. Simchi-Levi, and J. Bramel. Worst-case analyses, linear programming and the bin-packing problem. *Mathematical Programming*, 83:213–227, 1995.

[11] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem based on the spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[12] N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11:145–164, 1981.

[13] F. Cullen, J. Jarvis, and D. Ratliff. Set partitioning based heuristics for interactive routing. *Networks*, 11:125–144, 1981.

[14] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[15] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, *Handbooks in Operations Research and Management Science* 8, North-Holland, Amsterdam, 1995, pp. 35–139.

[16] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14:545–565, 1984.

[17] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

[18] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Annals of Operations Research*, 61:21–43, 1995.

[19] M. Haimovich and A.H.G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10:527–542, 1985.

[20] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[21] K.L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39:657–682, 1993.

[22] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, UK, 1990.

[23] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[24] C. Ribeiro and F. Soumis. A column generation approach to the multi-depot vehicle scheduling problem. *Operations Research*, 42:41–52, 1994.

[25] P. Seymour. Sums of circuits. In J. Bondy and U. Murty, editors, *Graph Theory and Related Topics*, Academic Press, New York, 1979, pp. 341–355.

[26] M.M. Solomon. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks*, 16:161–174, 1986.