Set Partitioning Based Heuristics for Interactive Routing

Frank H. Cullen, John J. Jarvis, and H. Donald Ratliff
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332

The set partitioning model is used as the basis for an interactive approach for solving a broad class of routing problems. A pricing mechanism is developed which can be used with a variety of methods in generating improving solutions. A version of the approach for delivery problems has been implemented via a colorgraphics display. The human aided optimization procedure was tested on the standard 50-point, 75-point, and 100-point test problems of Eilon, Watson-Gandy, and Christofides [6]. In the case of the first two test problems, the procedure was able to generate the best known solutions. In the 100-point problem, a better solution was generated than the current best known solution.

I. INTRODUCTION

We will consider here a set partitioning based approach for solving a broad class of routing problems. The approach is designed to take advantage of a high level of human interaction; the current implementation is interactive via a colorgraphics display. However, many of the concepts discussed here could be easily implemented in an automatic system.

The routing problem which motivated much of this work is what is called the static or subscriber dial-a-ride problem. This problem will be discussed in detail in later sections. It is one of the more complex members of the class of routing problems which are amenable to the approach presented here. This class also includes many practical delivery problems.

In order to introduce the underlying methodology which provides the basis for the approach, consider a very simple delivery example. Assume that a depot is located at the square box labeled D in Figure 1. From this depot a single delivery is to be made to each of the points represented by numbered circles. The numbers on arcs connecting the circles represents the travel distance between delivery points. Assume also that each vehicle (e.g., truck) can deliver to a maximum of two points on a single trip. The objective is to determine which vehicle should deliver to each point and the routing for the vehicles which minimizes the total distance traveled.

Each column in the matrix of Table I represents one possible vehicle route. For

NETWORKS, Vol. 11 (1981) 125-143

CCC 0028-3045/81/020125-19$01.90

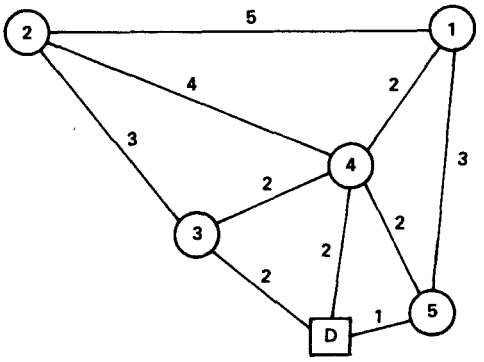


FIG. 1. Delivery example network.

example, column one represents a vehicle traveling from the depot to delivery point (1) and returning. The c_j row indicates the distance traveled for each trip. For example, column six represents a vehicle proceeding from the depot to delivery point (1), on to delivery point (2) and from there, back to the depot. The value of c_6 is 14, the total distance traveled for this trip. By enumerating each of the possibilities, as has been done for this matrix, the problem becomes one of selecting a set of columns such that every row is represented in exactly one column and the sum of the costs of the columns selected is the smallest possible. This integer program is called a "set partitioning model."

The set partitioning model was originally proposed for a similar class of routing problems by Charnes and Miller [3] and for this specific problem by Balinski and Quandt [2]. The model is very powerful in the sense that many realistic route constraints and cost functions can be handled easily in the column enumeration process. The obvious shortcoming of the model is that there are typically a very large number of columns to be enumerated and the resulting integer program is very large. The approach presented here is heuristic in the sense that we generate only a subset of the possible columns or routes and in general we do not solve the set partitioning model to optimality.

The set partitioning model has two very desirable features for interactive optimization. The first is that any route generated can be included as a column in the model. This allows the human interactor to utilize his/her intuition and spatial perception as well as a wide spectrum of mathematical techniques to generate new routes. The

TABLE I. Matrix corresponding to routes in the delivery example of Figure 1.

Route	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_j	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5
	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0
	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0
	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1
	0	0	0	0	1	0	0	0	1	0	0	1	0	1	1

second feature is that, unlike more general integer programs, a feasible solution to the set partitioning model provides the basis for pricing information which can be used to generate new candidate columns.

We will restrict the class of routing problems considered here to be those for which any subroute of a feasible route is also a feasible route with cost less than or equal to the cost of the original route. By imposing this restriction, as long as there is at least one 1 in every row of the partial set partitioning model that we have enumerated, we can easily generate feasible partitions. The only other restriction that we put on the class of routing problems is that we be able to pose them in a natural way as set partitioning problems. However, it should be noted that if there is not a nice spatial representation of the routing problem, the human interactor is much more restricted in his/her contribution.

II. SET PARTITIONING AND ROW PRICES

The set partitioning problem can be stated as

$$\text{minimize} \qquad Z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \qquad \sum_{j=1}^{n} a_{ij} x_j = 1 \qquad \text{for } i = 1, 2, \ldots, m \tag{2}$$

$$x_j = 0 \text{ or } 1 \qquad \text{for } j = 1, 2, \ldots, n. \tag{3}$$

For the delivery example in Table I, the set partitioning model has the second row as the value of the c_j and rows three through seven as the values of the a_{ij}. The variables which are set to one in a solution to the set partitioning problem will be called a "partition." We will denote a partition as $J^k = \{j | x_j^k = 1\}$. Balas and Padberg [1] provide a recent survey of results related to set partitioning problems.

A fundamental idea underlying much of the work presented here is the concept of "row prices."

Definition. $P^1 = (p_1^1, p_2^1, \ldots, p_m^1)$ is a set of feasible row prices corresponding to the partition J^1 if

$$\sum_{i=1}^{m} p_i^1 a_{ij} = c_j \qquad \text{for } j \in J^1. \tag{4}$$

It will be useful to interpret the price p_i^1 as an estimate of the cost to satisfy constraint i using solution X^1. For the delivery problem, p_i^1 is then an estimate of the cost of satisfying the requirement of delivery point i using the route corresponding to partition J^1.

Theorem 1. Given a set of feasible row prices $(p_1^1, p_2^1, \ldots, p_m^1)$ corresponding to partition J^1 with value Z^1, any other partition J^2 has value

$$Z^2 = Z^1 - \sum_{j \in J^2} \sum_{i=1}^{m} (p_i^1 a_{ij} - c_j). \tag{5}$$

Proof:

$$\sum_{j \in J^2} \left(\sum_{i=1}^{m} p_i^1 a_{ij} - c_j \right) = \sum_{i=1}^{m} p_i^1 \sum_{j \in J^2} a_{ij} - Z^2.$$

Since J^2 is a partition, $\Sigma_{j \in J^2} a_{ij} = 1$ for each $i = 1, 2, \ldots, m$. Also, since $(p_1^1, p_2^1, \ldots, p_m^1)$ are feasible row prices corresponding to X^1 we have $\Sigma_{i=1}^{m} p_i^1 = Z^1$. Hence the result follows.

Corollary 1. For any set of feasible row prices P^1 corresponding to a partition J^1 if

$$\sum_{i=1}^{m} (p_i^1 a_{ij} - c_j) \leqslant 0, \tag{6}$$

for $j = 1, 2, \ldots, n$ then X^1 is optimum.

It can also be shown, using linear programming duality, that a set of feasible row prices P^1 satisfying Corollary 1 exists if and only if X^1 is an optimum solution with the constraints $x_j = 0$ or 1 replaced by $x_j \geqslant 0$ for $j = 1, 2, \ldots, n$.

The quantity $\Sigma_{i=1}^{m} p_i^1 a_{ij} - c_j$ will be interpreted as the "potential" savings over the value of Z^1 which can result from constructing a partition that includes column j. Note from Theorem 1 that the potential savings can actually be achieved only if a partition can be constructed from columns with nonnegative potential savings.

III. POTENTIAL SAVINGS HEURISTIC

Given a partition J^1 and a corresponding set of feasible row prices P^1, an attractive heuristic for attempting to generate a better partition is the following:

Step 0: Let $J^2 = \phi$ (J^2 will be the indices of columns in the new partition) and $N = \{1, 2, \ldots, n\}$, (N will be the indices of columns which are candidates for inclusion in J^2)

Step 1: Calculate the potential savings $\Sigma_{i=1}^{m} p_i^1 a_{ij} - c_j$ for $j = 1, 2, \ldots, n$.

Step 2: Pick the column k in N with the largest potential savings, $\Sigma_{i=1}^{m} p_i^1 a_{ik} - c_k$.

Step 3: For $i = 1, 2, \ldots, n$, if $a_{ik} = 1$ set $a_{ij} = 0$ for all $j \neq k$. (Note from the assumption of Sect. I that any subroute of a feasible route is also a feasible route, the new columns are legitimate.)

Step 4: Let $J^2 = J^2 U \{k\}$ (i.e., put column k in the new partition) and $N = N - \{k\}$.

Step 5: Delete from N all j for which $a_{ij} = 0$ for all $i = 1, 2, \ldots, m$.

Step 6: If $N = \phi$ stop. Otherwise go to step 2.

Note that under the assumption that any subset of a route is also a feasible route (discussed in Sec. I) this procedure will always terminate with J^2 as a partition al-

TABLE II. An example illustrating the potential saving heuristic with $J^1 = \{1, 2, 3, 4, 5,\}$, $J^2 = \{6, 13, 5\}$, and $J^3 = \{8, 10, 5\}$.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
c_j	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	P^1	P^2	P^3	
	1					1	1	1	1							8.0	6.2	5.3	
		1				1				1	1	1				10.0	7.8	7.1	
			1				1			1			1	1		4.0	3.0	2.9	
				1				1			1		1		1	4.0	3.0	2.7	
					1				1			1		1	1	2.0	2.0	2.0	
$\sum_{i=1}^{m} p_i^1 a_{ij} - c_j$	0	0	0	0	0	4	2	4	2	4	3	0	2	0	0	$x_6 = 1$			
	–	–	0	0	0	–	-6	-4	-6	-6	-7	-10	2	0	1	$x_{13} = 1$			
	–	–	–	–	0	–	–	–	-6	–	–	-10	–	-4	-3	$x_5 = 1$			
$\sum_{i=1}^{m} p_i^2 a_{ij} - c_j$	-1.8	-2.2	-1	-1	0	0	-0.8	1.2	0.2	0.8	-0.2	-2.2	0	-1	0	$x_8 = 1$			
	–	-2.2	-1	–	0	-6.2	-7	–	-6	0.8	-3.2	-2.2	-3	-1	-3	$x_{10} = 1$			
	–	–	–	–	0	–	–	–	-6	–	–	-10	–	-4	-3	$x_5 = 1$			
$\sum_{i=1}^{m} p_i^3 a_{ij} - c_j$	-2.7	-2.9	-1.1	-1.3	0	-1.6	-1.8	0	-0.7	0	-1.2	-2.9	-0.4	-1.1	-0.3				

though not necessarily a better partition than J^1. If convenient, c_j should be recomputed whenever step 3 affects a change in column j, as the actual route may change. Computation to date indicates that an optimum or near optimum solution to the set partitioning problem is determined very quickly by repeated application of the potential savings heuristic (note that the a_{ij} should be reset to their original values after each repetition). The heuristic is repeated until either optimality is proven (i.e., for some X^k all $\Sigma_{i=1}^m p_i^k a_{ij} - c_j \leqslant 0$) or until some specified number of partitions has been generated.

To illustrate the potential savings heuristic, consider again the delivery example depicted in Figure 1 and Table I. Suppose that we select $J^1 = \{1, 2, \ldots, 5\}$ as an initial partition. A set of feasible row prices P^1 is given in Table II. (The question of how to generate "good" feasible row prices will be addressed in the next section.) The corresponding potential savings $\Sigma_{i=1}^m p_i^1 a_{ij} - c_j$ are also given in Table II. Applying the potential savings heuristic and breaking ties by selecting the column with the lowest index yields the new partition $J^2 = \{6, 13, 5\}$. The new partition has a cost of $Z^2 = 22$ as compared to a cost $Z^1 = 28$ for the initial solution.

Using the row prices P^2 and potential savings $\Sigma_{i=1}^m p_i^2 a_{ij} - c_j$ of Table II and reapplying the potential savings heuristic yields the partition $J^3 = \{8, 10, 5\}$ which has a cost of $Z^3 = 20$. Again, from Table II we find that using the row prices P^3 gives potential savings $\Sigma_{i=1}^m p_i^3 a_{ij} - c_j \leqslant 0$ for $j = 1, 2, \ldots, n$. Hence, from Corollary 1 the partition J^3 is optimum.

IV. ROW PRICING

For a given partition X^k a set of feasible row prices is obtained by allocating the column cost c_j for each $j \in J^k$ among the rows having $a_{ij} = 1$. For the delivery example, this corresponds to allocating the trip cost among the delivery points of the trip. When a column $j \in J^k$ contains only one $a_{ij} = 1$, the row price is $p_i^k = c_j$. However, when a column j contains more than one $a_{ij} = 1$, there are an infinite number of possible sets of prices. As an example consider the partition $J^3 = \{8, 10, 5\}$ for the problem in Table II. Since column 5 has only $a_{5,5} = 1$ the value $p_5^3 = 2$ is unique. Column 8 has both $a_{1,8} = 1$ and $a_{4,8} = 1$, hence the cost $c_8 = 8$ could be allocated between rows 1 and 4 in an infinite number of ways. Similarly, column 10 has both $a_{2,10} = 1$ and $a_{3,10} = 1$, hence $c_{10} = 10$ could be allocated between rows 2 and 3 in an infinite number of ways. If we allocate c_8 as $p_1^3 = 4$ and $p_4^3 = 4$ and allocate c_{10} as $p_2^3 = 5$ and $p_3^3 = 5$, the resulting $\Sigma_{i=1}^m p_i^3 a_{ij} - c_j$ do not indicate that J^3 is an optimum partition. Hence the set of prices P^3 given in Table II are clearly better since they do indicate that J^3 is on optimum partition.

Ideally, we would like a set of prices which would drive the potential savings heuristic toward an improving solution and would indicate optimality when no improving solution is possible (i.e., We would like the prices to be analogous to dual variables in linear programming). Unfortunately, it is easy to construct cases for which no such prices exist (i.e., any problem for which the integer and continuous solution differ). For the delivery problem and the more complex dial-a-ride problem (to be discussed later), allocating column cost in proportion to the cost of serving the delivery points one-at-a-time is intuitively appealing and seems to work very well. As an illustration consider

again the partition J^3 in Table II. Column 8 has $a_{1,8} = 1$ and $a_{4,8} = 1$. The cost of serving delivery point 1 if it is the only point in a trip is $c_1 = 1$. The cost of serving delivery point 4 if it is the only point in a trip is $c_4 = 4$. The prices for rows 1 and 4 were determined as

$$p_1^3 = \frac{c_1 c_8}{c_1 + c_4} = 5.3 \quad \text{and} \quad p_4^3 = \frac{c_4 c_8}{c_1 + c_4} = 2.7.$$

The other prices in Table II were determined similarly.

When there are substantial differences in the amounts of vehicle capacity required by the delivery points, it seems reasonable to allocate column cost in proportion to the "weighted" cost of serving points one-at-a-time. Here the weights are the delivery sizes. For example if w_i is the vehicle capacity required by delivery point i, we specify the price of row 1 with respect to J^3 above as

$$P_1 = \frac{c_1 w_1 c_8}{c_1 w_1 + c_4 w_4}$$

These and other pricing alternatives are currently being tested for delivery problems under various types of constraints.

There are a variety of other mechanisms for allocating the prices to the demand points. Some allocation procedures may work well in some situations while others may be suited to other problems. The human (or automatic algorithm) may wish to employ different pricing procedures as the algorithm progresses.

V. COLUMN GENERATION

For large scheduling and routing problems it is generally not practical to generate all columns of the corresponding set partitioning model. The remainder of this paper will be concerned with using information gleaned from one solution via Theorem 1 to generate a new and hopefully better solution. This is accomplished by either generating new columns, adding them to the current set partitioning model, and then resolving the model or by using the information from Theorem 1 directly to generate a new solution. In the latter case it is not necessary to retain the columns of the set partitioning model. However, if the columns are retained, it is possible to further improve the solution by periodically solving the set partitioning model.

We should note that for the class of scheduling and routing problems being considered here, it is very easy to generate an initial solution. For our examples we use the identity solution (e.g., in the delivery problem this is the solution which has each vehicle making a single delivery) as our initial solution. However, any feasible solution could be used as the initial solution.

Clearly there is a broad spectrum of possible approaches that one might use to generate new columns and/or new solutions to the set partitioning model. In fact, variations of many of the heuristics which have been applied to delivery problems can be used very effectively in conjunction with Theorem 1. In the next section we will discuss the use of the Clarke and Wright [3] savings procedure in conjunction with the

delivery problem. In later sections we will discuss more complex clustering and chaining heuristics as we have applied them to the dial-a-ride problem.

VI. CLARKE AND WRIGHT PROCEDURE WITH PRICING

The "savings" heuristic of Clarke and Wright [3] is the most widely known of the heuristics developed to date for delivery problems. The algorithm proceeds by calculating a savings for each pair of delivery points i and j defined as

$$s_0(i,j) = 2d_{0i} + 2d_{j0} - (d_{0i} + d_{ij} + d_{j0}) = d_{0i} + d_{j0} - d_{ij}$$

which is the savings in mileage of supplying delivery points i and j on the same route as opposed to supplying them individually directly from the depot (d_{0i} is the distance from the depot to delivery point i). Routes are then constructed either one-at-a-time or in parallel by considering pairs of points in order of decreasing savings and including them in the same route if such a route is feasible.

Suppose that we consider once more the delivery example in Figure 1 and the covering solution information in Table II. Note that the Clarke and Wright savings values are exactly the values of $\Sigma_{i=1}^{m} p_i^1 a_{ij} - c_j$ in Table II since $d_{0i} = p_i^1/2$, for $i = 1, 2, \ldots, m$. Applying the C-W savings algorithm yields the same routing configuration as $J^2 = \{6, 13, 5\}$. At this point, the C-W algorithm would terminate. However, suppose that we set $d_{0i} = p_i^2/2$, for $i = 1, 2, \ldots, m$. The new savings are then exactly the $\Sigma_{i=1} p_i^2 a_{ij} - c_j$ in Table II. Applying C-W algorithm yields the same routing configuration as $J^3 = \{8, 10, 5\}$ which as noted previoulsy is the optimum solution to this delivery example. Hence, for this example at least the C-W algorithm without pricing did not yield an optimum solution while the same algorithm when combined with pricing did yield the optimum solution.

Note that when routes are allowed to contain more than two trips, only a small subset of the set partitioning columns would be generated using this procedure. Also, note that the prices and savings can be calculated without ever generating the set partitioning matrix.

When the number of points allowed in a route exceeds two, some interesting questions arise as to exactly how the algorithm should be implemented. As an illustration, suppose that in the example we allow a vehicle to deliver to at most three points rather than two. Now suppose that we start with the solution J^3 in Table II. We see that the two-at-a-time potential savings are all nonpositive. However, if we ignore this fact and proceed with the algorithm, we would put points 1 and 4 in the same route since their potential savings of zero is maximum. If we then consider adding a third point to this route, we find that adding point 5 has a potential savings of 2 which is maximum. Finally, we combine points 2 and 3 into a single route since this has a potential savings of zero. The resulting routes (4, 1, 5) and (2, 3) has a length of 10. Therefore, in this case at least we can get better information by recalculating the potential savings after each augmentation of a route. This recalculation is not done in most implementations of the C-W algorithm.

Clearly, when constructing a route containing more than two points one must decide where in the route to put each additional point. The potential savings can be determined exactly only by solving a travelling salesmen problem over each new point

which is a candidate to be added to the route. This is computationally expensive if a route can contain a large number of points. In most implementations of the C-W algorithm, new points are simply added on to the end of the route being constructed. When the algorithm is implemented interactively using computer graphics, it appears that the human can perform an important role both in selecting candidate points and in inserting them logically into routes.

VII. LOCATION–ALLOCATION WITH PRICING

Another procedure which Krolak and Nelson [5] have found effective in approaching the delivery problem utilizes the location-allocation model of Cooper [4]. This basic concept can also be used in conjunction with Theorem 1 to generate intuitively appealing columns to add to the set partitioning model.

To illustrate the procedure consider the delivery example illustrated in Figure 2. Again the circles represent delivery points and the square represents the depot. The basic idea is to use a surrogate distance rather than the actual distance in determining the points which are assigned to each vehicle. The surrogate distance is obtained by assuming that the vehicle travels from the depot to a specified cluster point, represented in Figure 2 by the dashed circles. It then makes the deliveries, returning after each delivery to the cluster point. After all deliveries have been made, the vehicle returns to the depot. Under this surrogate distance, the problem becomes one of locating the cluster points, one for each vehicle, and then assigning the delivery points to each vehicle.

If (a_0, b_0) represents the coordinates of the depot, (a_i, b_i) represents the coordinates of delivery point i, (x_j, y_j) represents the coordinates of cluster point j, and assuming Euclidean distance, the problem can be modeled as follows:

$$\min_{x,y,z} \sum_{i=0}^{m} \sum_{j=1}^{n} 2[(x_j - a_i)^2 + (y_j - b_i)^2]^{1/2} z_{ij}$$

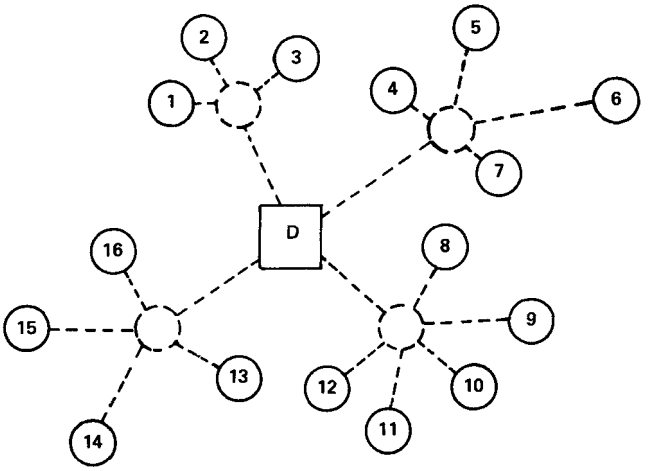


FIG. 2. Delivery example to illustrate the location–allocation model for clustering.

subject to

$$\sum_{i=1}^{m} z_{ij} \leqslant K, \quad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} z_{ij} = 1 \qquad i = 1, 2, \ldots, m$$

$$z_{ij} = 0 \text{ or } 1 \text{ all } i, j$$

$$z_{0j} = 1 \qquad j = 1, 2, \ldots, n,$$

where n is the number of vehicles and K is the vehicle capacity. When $z_{ij} = 1$, delivery point i is assigned to vehicle j and when $z_{ij} = 0$, delivery point i is not assigned to vehicle j.

While there is no method for efficiently solving this problem optimally, the following is an attractive heuristic. Pick a set of locations for the cluster points. With these coordinates fixed, solve the resulting assignment problem. With these values of z_{ij} fixed, solve the resulting location problem. Continue alternating between the assignment and location problems for some specified number of iterations or until there is no further improvement in the objective. Once a cluster has been determined, the vehicle is then routed among the points of the cluster.

A slight modification of this model, together with Theorem 1, allows us to generate attractive new columns for the set partitioning problem. Suppose that we have a solution to the set partitioning problem and a set of row prices $p_1, p_2, \ldots, p_m$. Now consider the model

$$\min_{x,y,z} \sum_{i=0}^{m} \sum_{j=1}^{n} 2[(x_j - a_i)^2 + (y_j - b_i)^2]^{1/2} z_{ij} - \sum_{i=1}^{m} \sum_{j=1}^{n} p_i z_{ij}$$

subject to

$$\sum_{i=1}^{m} z_{ij} \leqslant K \qquad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} z_{ij} \leqslant 1 \qquad i = 1, 2, \ldots, m$$

$$z_{ij} = 0 \text{ or } 1 \text{ all } i, j$$

$$z_{0j} = 1 \qquad j = 1, 2, \ldots, n.$$

Any cluster generated by this model will have a positive potential savings, with respect to the surrogate distance. Hence, it corresponds to an attractive column to add to the set partitioning problem (considering surrogate distances).

Since the second constraint has been changed to an inequality, not all delivery points will be assigned to cluster points. This simply means that the current row price for

the delivery point is more attractive than the cost of serving the delivery point in alternative clusters considered by the model. The model can be solved using the same heuristic discussed for the earlier location-allocation model.

VIII. DIAL-A-RIDE PROBLEM

The dial-a-ride problem is a much more complex routing problem than the delivery problem. In the dial-a-ride problem we are given an origin-destination trip matrix and an underlying network on which the trips are to be made. There is a single item (people, goods, etc.) (demand for service) at each origin that needs to be transported to its specified destination. The items are transported from origins to destinations on vehicles each having capacity K. We wish to satisfy the trip requirements while travelling the minimum distance.

This is a "static" version of the dial-a-ride problem since time is not considered. There are a number of more complex versions of this problem, but this version is sufficient to demonstrate the basic ideas of our approach.

The set partitioning model for the dial-a-ride problem is analogus to that of the delivery problem, but here rows represent trips rather than simple delivery points. The vehicle capacity constraints are handled by generating only routes which satisfy them.

IX. DECOMPOSITION

In delivery problems, representative of the "one-ended" class of routing and scheduling problems, we need only be concerned with a single stop for a vehicle to satisfy a particular demand for service. In contrast, the "two-ended" class, which includes dial-a-ride problems, requires two stops in a specific order (sequence) to satisfy a specific demand for service. It is this requirement for sequencing of "pickup" and "dropoff" point pairs which adds greatly to the difficulty of handling the two-ended class of vehicle routing problems.

When one considers examples of two-ended problems, it becomes immediately apparent that the sequencing requirements greatly inhibit the complex pattern processing abilities of the human. Figure 3 illustrates an example of a 25 trip dial-a-ride

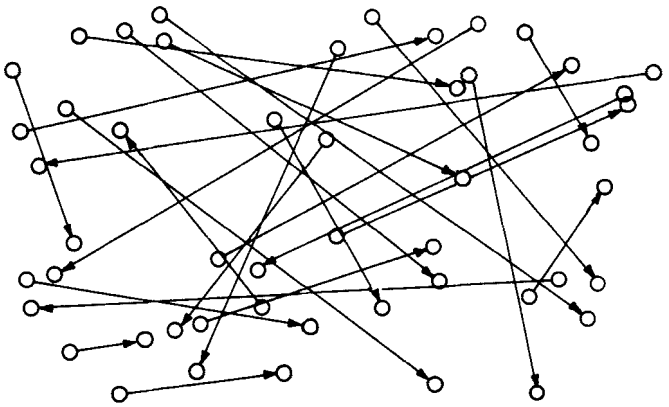


FIG. 3. An example of a 25 trip dial-a-ride problem.

problem. Rather than displaying order and structure, the problem resembles so much spaghetti. It is clear that for such problems the human interactor needs more help in generating good columns for the set partitioning model.

Unfortunately, it is also more difficult to apply straightforward methods such as the savings approach discussed earlier for the delivery example. In generating a dial-a-ride route one must be concerned with where both the origin and the destination occur in the sequence in order to calculate the potential savings. In addition, the capacity constraint may negate what otherwise appears to be good positions for the origin and destination in the sequence.

Because of this complexity, it is helpful to "decompose" the problem into two levels which we call "clustering" and "chaining." In essence, we consider a route to be made up of two components. Clusters correspond to trips which can all be on a vehicle at one time, while chains correspond to movement from the end of one cluster to the beginning of the next. Figure 4 provides an example of five good clusters, while Figure 5 illustrates one way in which these five clusters might be linked into two chains.

The partitioning model and pricing concepts can be effectively exploited to aid in generating improving clusters and chains in a column generation approach to solving two-ended vehicle routing problems. One partitioning model can be ultilized in generating improving clusters while another partitioning model helps identify better chains.

In the partitioning model for clustering, the columns represent clusters and the rows represent the individual trips (demand for service). We shall demonstrate, in the next section, just how the pricing information from the partitioning problem can be used to generate additional clusters.

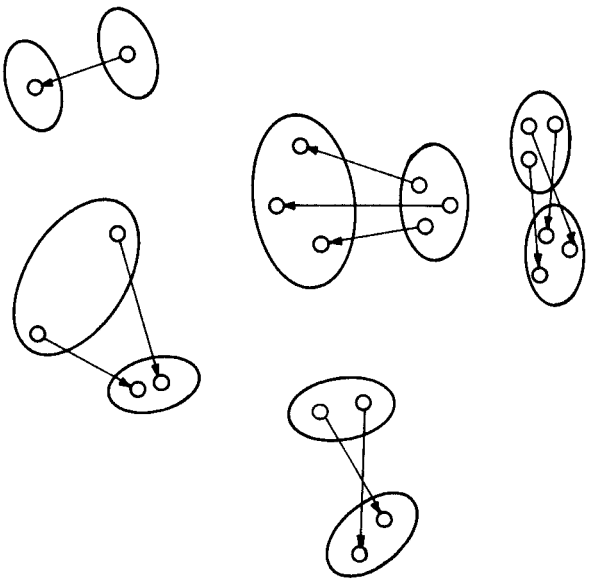


FIG. 4. Example of five clusters.

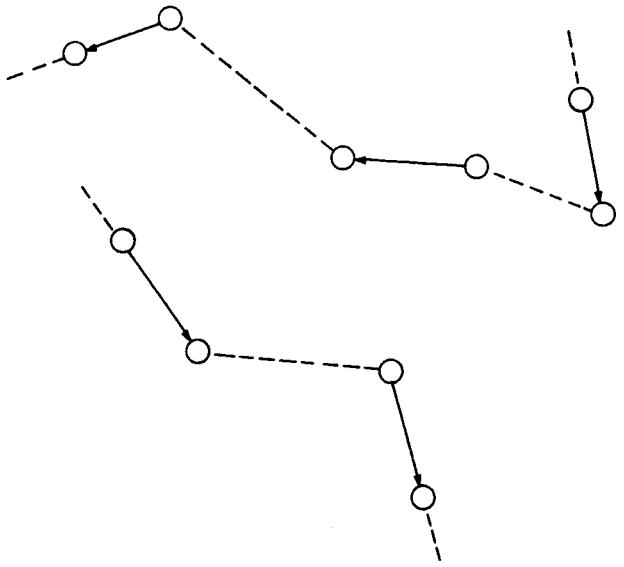


FIG. 5. Example of two chains linking the five clusters of Figure 4.

In the partitioning model for chaining, the columns represent chains and the rows represent the individual trips. The function of the chaining partitioning model is to combine the clusters into good vehicle routes.

The overall solution procedure consists of linking the clustering models and chaining models together in an interactive manner. The clustering models, partitioning matrix and pricing information are used to generate good clusters. These clusters are then passed to the chaining phase where they are linked together via the chaining models, partitioning matrix and associated pricing information. After chaining, it is possible to return to the clustering phase to identify additional clusters.

The next two sections discuss the specifics of clustering and chaining.

X. CLUSTERING

In the previous section we introduced the clustering concept. In this section we shall provide more details of the concept as well as the structure and operation of various clustering models. Figure 6 depicts a typical cluster (in this case, three trips).

Clustering makes sense if the origins are reasonably close together and the destinations are also close together. One way to develop an evaluation of such circumstance is to locate the centroid of the origins, the centroid of the destinations and accumulate the resulting distances from the original trips. In Figure 6 we could evaluate the distances represented by $(a + b + c) + (d + e + f)$. If this sum is small then it would make sense to cluster the trips.

By utilizing surrogate distances we lose the actual route distance evaluation; however, we gain the ability to evaluate large numbers of cluster possibilities conveniently and simultaneously. In Figure 6 we might employ Euclidean distances. In this case we could compare the sum of the row prices $p_1 + p_2 + p_2$ generated in the covering

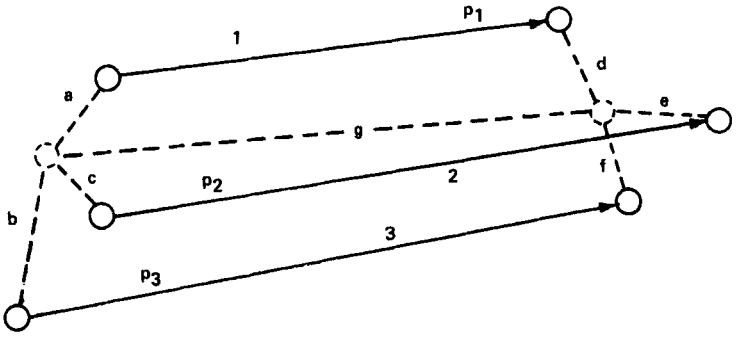


FIG. 6. Surrogate distances for a typical cluster.

model for clustering to the quantity $2(a+b+c)+2(d+e+f)+g$ to determine whether clustering is appropriate. We can think of the latter quantity as a surrogate for the vehicle routing distance.

We can develop a straight forward extension of the location-allocation model discussed in Sec. VII to identify good clusters for the dial-a-ride problem. In place of the quantity

$$\sum_i \sum_j 2[(x_j - a_i)^2 + (y_j - b_i)^2]^{1/2} z_{ij}$$

we employ a qunatity

$$\sum_i \sum_j 2[(\bar{x}_j - \bar{a}_i)^2 + (\bar{y}_j - \bar{b}_i)^2] z_{ij}$$

$$+ \sum_i \sum_j 2[(\hat{x}_j - \hat{a}_i)^2 + (\hat{y}_j - \hat{b}_i)^2]^{1/2} z_{ij}$$

$$+ \sum_j [(\bar{x}_j - \hat{x}_j)^2 + (\bar{y}_j - \hat{y}_j)^2]^{1/2},$$

where $\bar{x}_j, \bar{y}_j, \bar{a}_i$ and $\bar{b}_i$ correspond to origins and $\hat{x}_j, \hat{y}_j, \hat{a}_i$ and $\hat{b}_i$ correspond to destinations of clusters and trips.

We can also develop a savings approach to clustering in a similar manner to that described earlier for the delivery problem. As in the delivery problem, the clusters generated by the location-allocation models by the savings approaches or by the human interaction can be achieved in a set partitioning model. This model can then be solved to determine a "best" set of clusters.

XI. CHAINING

Upon termination of the clustering process, a number of reasonably good clusters are available. This set includes not only the "best" cluster sets as selected by the cov-

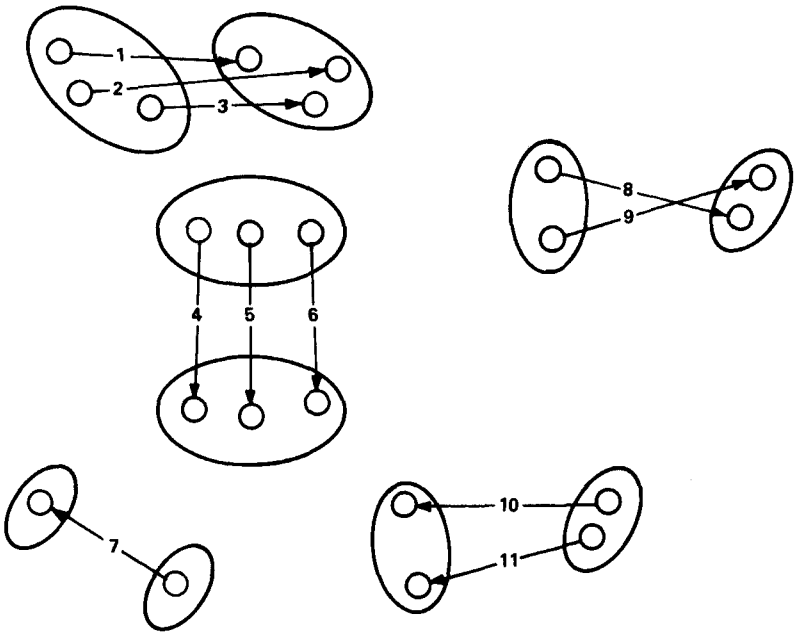


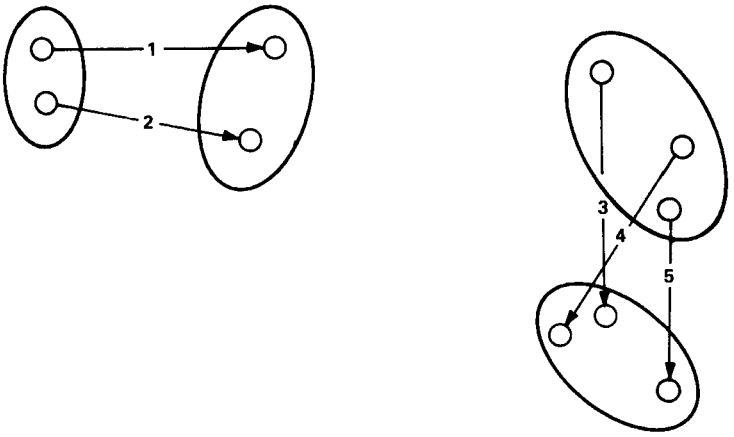
FIG. 7. Example of clusters resulting from the clustering process.

ering model for clustering, but also a number of other clusters which might be nearly as good, but which were not selected in the optimal solution to the covering model. The pricing mechanism can again be utilized to select these additional "good" clusters. Figure 7 illustrates a set of clusters which might result from the clustering process. The clusters in the figure represent only the optimal solution to the clustering process. In addition, we might have other clusters available, e.g., a cluster containing only trips 1 and 2.

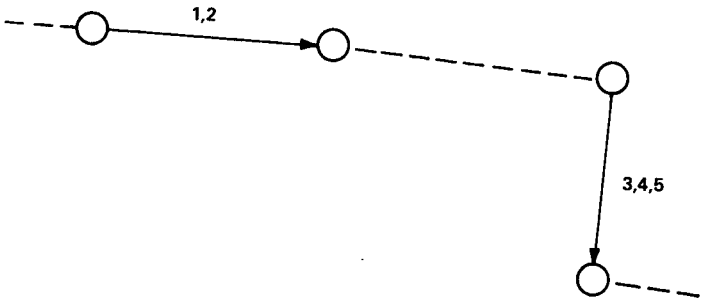
The clusters obtained represent good possibilities for segments (legs) of a vehicle route. The next step in the process is to link ("chain") these clusters into complete vehicles routes. Figure 8 illustrates the chaining concept. Trips 1 and 2 form one cluster, while trips 3, 4, and 5 form another cluster [see Fig. 8(a)]. In Figure 8(b), trips 1 and 2 are replaced by a single pseudo (cluster) trip, as is also the case for trips 3, 4, and 5. These cluster trips are then chained together.

The interpretation of chaining is that a single vehicle will service the first set of trips [in Figure 8(b) these would be trips 1 and 2] and then proceed to service the next set of trips (i.e., trips 3, 4, and 5) in the chain. Figure 9 illustrates the likely vehicle route to service the two clusters.

In the partitioning model for chaining, we associate a column of the partitioning matrix with each feasible chain (vehicle route) and a row for each trip. Column j contains 1 in row i if trip i is serviced by chain (vehicle route) j. Otherwise, it contains a 0 (zero). The zero-one variable associated with the columns provide indications of which chains were selected in the optimal partitioning solution. Table III illustrates a partitioning matrix for several chains in Figure 7.



a. Two Typical Clusters



b. The Associated Cluster Arcs Chained Together

FIG. 8. An example of chaining clusters together.

The column generation process for the chaining problem is similar to that for the delivery problem in Sec. V. However, there are two differences which must be addressed. The first is that in the delivery problem we were routing through "points" while in chaining, we are routing through "lines" (one for each cluster). This requires

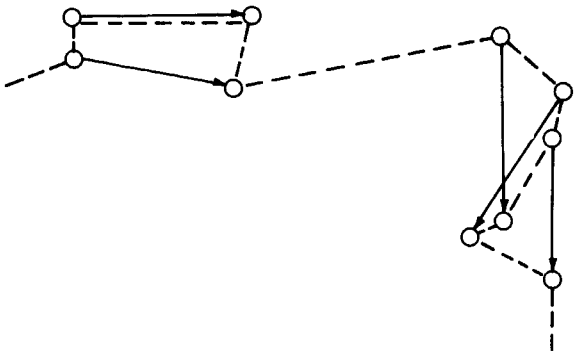


FIG. 9. Probable vehicle route for the chain of Figure 8(b).

TABLE III. An example partitioning model for certain chains in Figure 7.

	Clusters 1 & 2	Clusters 1 & 4	Clusters 2 & 5	Clusters 1, 2, & 5	Clusters 3, 4, & 5	Clusters 1, 2, & 4	Clusters 1, 2, 3, & 5
	1	2	3	4	5	6	7
1	1	1		1		1	1
2	1	1		1		1	1
3	1	1		1		1	1
4	1		1	1		1	1
5	1		1	1		1	1
6	1		1	1		1	1
7					1		1
8		1			1	1	
9		1			1	1	
10			1	1	1		1
11			1	1	1		1

only minor changes in the savings approach; however, the location-allocation is no longer appropriate. The second difficulty occurs because we do not want two clusters in the same chain if they have a trip in common. This is easily avoided in the same manner as we avoid exceeding vehicle capacity in the delivery problem.

XII. DIAL-A-RIDE SOLUTION PROCEDURE

We may put all of the foregoing ideas together into an algorithm for the dial-a-ride problem. Figure 10 provides a flowchart for the algorithm. All of the operations ① thru ⑦, in Figure 10 have been previously discussed except operation ④.

The chaining procedure forms clusters into good vehicle routes. In order to increase the flexibility of the chaining procedure we must provide it with a number of clusters to work with. Thus, in addition to passing the optimal set of clusters to the chaining procedure we should also pass a number of other good clusters. For example, we might pass clusters which priced out near optimal in the partitioning problem for clustering. This is what is suggested in operation ④.

XIII. PRELIMINARY COMPUTATIONAL RESULTS

The delivery algorithm described in this paper has not yet been fully implemented. However, for evaluation purposes a version was tested which employed prices to aid the human in locating improving delivery solutions, but did not include a partitioning problem to locate a best set from the candidate routes generated. The version was tested on the 50-point, 75-point and 100-point problems of Eilon, Watson-Gandy, and Christofides [6]. All of these problems (1) employ a single depot, (2) are Euclidean, and (3) have a limited vehicle capacity.

Near-optimal solution values for each of the three problems selected are given in Russell [8]. They are 524, 854, and 833, respectively. For each problem, one of the authors employed the interactive procedure until no improvement could be made. In every case, Russell's solution value was equaled or bettered. Table IV presents the

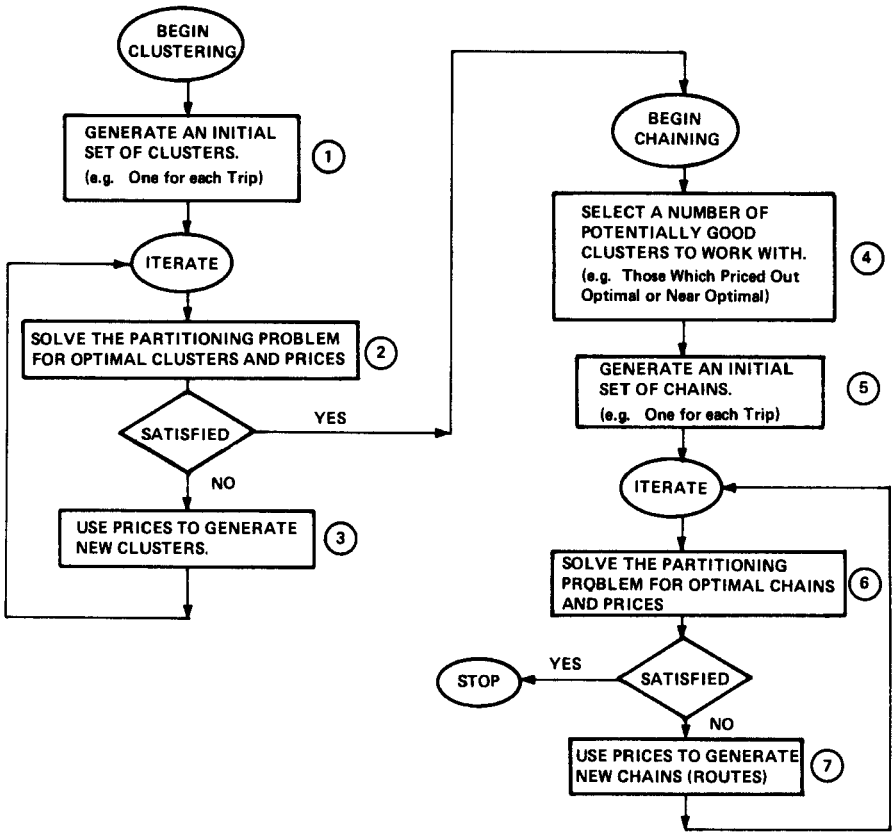


FIG. 10. Flowchart of the dial-a-ride solution procedure.

TABLE IV. Results of the proposed method compared with Russell's solutions.

	50 Point	75 Point	100 Point
Russell's solution	524	854	833
Iteration 1	559	875	896
Iteration 2	541	870	891
Iteration 3	528	857	868
Iteration 4	526	854	860
Iteration 5	524		856
Iteration 6			852
Iteration 7			835
Iteration 8			833
Iteration 9			830
Iteration 10			829
Iteration 11			827

results of the three tests. Any time a new set of routes was generated it was counted as an iteration. In some iterations all new routes were generated while in others some of the routes which seemed good from the previous iteration were retained and new routes generated only for the remaining points. In general, those iterations where the costs were very close from one iteration to the next were cases where most of the routes from the previous iteration were retained. The original set of prices were generated from the solution where each point was in a route by itself. New prices were generated at the end of each iteration based on the solution generated at that iteration.

XIV. CONCLUSIONS

An interactive delivery system and dial-a-ride system based on the concepts presented here have been implemented on a Chromatics colorgraphics terminal interfaced with a CYBER 74 mainframe computer. Preliminary tests indicate that the interactive procedure is able to compete with state-of-the-art automatic procedures for delivery problems without a great degree of human effort. Although the systems are in many ways very rudimentary, they dramatically indicate the potential for this kind of human aided optimization.

The interactive procedures should derive their greatest benefits in more complex problems containing unusual side constaints (e.g., multiple depots, time, etc.) which the human is able to perceive and control. We are in the process of making extensive modifications in the software and are testing a variety of new models and heuristics to aid in clustering and in route generation.

This work was partially supported by the Transportation Systems Center and the Office of Naval Research. David J. Friedman and Robert T. Lewis contributed significantly to the implementation of the interactive dial-a-ride system. The extension of the location–allocation model to the dial-a-ride problem is due to Robert T. Lewis.

References

[1] E. Balas and M. W. Padberg, "Set Partitioning: A Survey," *SIAM Rev.*, 18, 710–760 (1976).

[2] M. L. Balinski and R. E. Quandt, "On An Integer Program for a Delivery Problem," *Oper. Res.*, 12, 300–304 (1964).

[3] A. Charnes and M. H. Miller, "A Model for the Optimal Programming of Railway Freight Train Movements," *Manage. Sci.*, 3, 74–92, (1956).

[4] G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Oper. Res.*, 12, 569–581 (1964).

[5] L. Cooper, "*N*-Dimensional Location–Allocation Used for Cluster Analysis," Report No. C00-1493-23, Washington University, St. Louis, MO, 1969.

[6] S. Eilon, C. D. Watson-Gandy, and N. Christofides, *Distribution Management*, Hafner, New York, 1971, pp. 200–202.

[7] P. D. Krolak and J. H. Nelson, "A Family of Truck Load Clustering (TLC) Heuristics for Solving Vehicle Scheduling Problems," Technical Report 78-2, *Computer Science*, Vanderbilt University, Nashville, TN, 1978.

[8] R. A. Russell, "An Effective Heuristic for the M-Tour Travelling Salesman Problem with Some Side Constraints," *Oper. Res.*, 25, 517–524 (1977).