Control and Cybernetics

vol. **25** (1996) No. 1

The modeling and solution of a class of dial-a-ride problems using simulated annealing

by

Stephen M. Hart

A Priori Solutions, Inc.
3338 Woodsfield Drive Suite C, Marietta, GA 30062, USA

Abstract: This paper presents a new mixed integer linear programming mathematical model of the dial-a-ride problem (DAP) which facilitates incorporation of multiple criteria and accommodates various forms of the basic DAP model. This new mathematical model encapsulates into a single model various instances of the DAP previously available only in separate models.

A simulated annealing based solution heuristic is developed for the model. Extensive analysis of the effects of the annealing parameters on the efficiency and accuracy of results is performed. The robustness of the algorithm is explored through solution of various instances of the problem. The intent is to explore the applicability of simulated annealing as a solution methodology for the DAP, and to examine the design issues associated with implementation of same.

The results obtained in this paper are encouraging for small to medium sized instances of the DAP with efficiences (with respect to objective value) exceeding 95%.

1. Introduction

Although the requirements of the dial-a-ride problem (DAP) model are readily described, efficient solutions to existing models are evasive due to its NP-hard computational complexity. Several mathematical models and solution methodologies exist for different classes of the DAP; however, there is no single mathematical model which can encompass them all. Development of such a robust model will allow researchers and practitioners to rely on a single model and its corresponding solution methodology rather than searching for or developing a dedicated model for their specific application. Moreover, since most real-world problems typically involve more than one, usually conflicting objective or criteria, the versatility of the DAP model will be enhanced through incorporation of multiple criteria; however, very little research has been conducted in this area. Finally, since the DAP is NP-hard, little emphasis has been placed on development of exact algorithms, and although several heuristic algorithms exist in

the literature, none have explored the application of simulated annealing to the problem. Moreover, the existing heuristics have concentrated on solving a specific instance of the DAP rather than developing a robust solution methodology applicable to multiple instances of the problem.

The objective of this research was to develop a robust mathematical model of the DAP which can encompass various instances of the problem and facilitate the inclusion of multiple criteria. Additionally, since the complexity of the mathematical model would render its exact solution intractable, the utility of a simulated annealing-based heuristic solution methodology was explored.

This paper is divided into five sections. Section 2 provides background into the DAP and its associated vehicle routing problem. Additionally, an overview of simulated annealing is provided. In Section 3, the mathematical model of the DAP problem is formulated, and in Section 4, the simulated annealing solution algorithm is presented. This section includes an extensive analysis of the performance of the algorithm for various control parameters. The paper is concluded in Section 5.

2. Background

2.1. Dial-a-Ride problem

The dial-a-ride problem (DAP) is a subset of the general vehicle routing problem (VRP), Christofides et al. (1979). In general, the VRP involves the routing (i.e., the order in which customers are serviced) of one or more vehicles to and from a central depot to multiple customer demand centers, and the scheduling (i.e., the times at which customers are picked up and/or delivered) of the arrival and departure times of vehicles to/from the customer demand centers and depot.

The dial-a-ride problem encompasses a class of problems in which customers located at various points in the service area call a central dispatcher who collects information on the customer's location, his desired destination, and desired pickup and delivery times. The dispatcher then determines which vehicle will service the demand, determines the route the vehicle will travel, and determines the vehicle's departure and arrival times (i.e., the vehicle's schedule). As with the general VRP, dial-a-ride problems can be subdivided according to the assumptions of the model:

1. Number of pickup and delivery points. In this case, customers request pickup from several distinct points with delivery to a single, common point (many-to-one) or delivery to many distinct points (many-to-many).
2. Request type or time of reservation. Customer reservations for transportation can be made in advance (static) or as the service is needed (dynamic or immediate request).
3. Number of vehicles. The fleet can consist of a single or multiple vehicles.
4. Timing constraints. Customers may specify a fixed time for pickup and/or delivery or may specify an interval of time during which pickup and/or

delivery may occur (timing windows).
5. Vehicle capacities. Constraints on the maximum number of customers per vehicle may be imposed.

The objectives of the models are also problem dependent and include:

1. Minimization of the number of vehicles or fleet size.
2. Minimization of customer wait times, travel times, or the summation of both (i.e., the system time).
3. Minimization of vehicle travel distances.

The literature on the VRP and its extensions is extensive as is evident in the review articles by Bodin et al. (1983) and Solomon and Desrosier (1988). Some specific examples follow.

Psaraftis (1980) studied the single vehicle, many-to-many, immediate request problem. The single vehicle, many-to-many, advanced reservation problem with delivery time windows was studied by Sexton and Bodin (1985a,1985b). Kikuchi and Rhee (1989) consider a multiple vehicle, many-to-many, advance reservation problem with pickup and delivery time windows employing an insertion heuristic similar to that developed by Jaw et al. (1986). Foster and Ryan (1976) developed a 0-1 integer programming formulation of a vehicle scheduling problem (VSP) in which the primary objective was to minimize the maximum number of vehicles deployed on any given day of a multi-day planning horizon. Their model facilitates inclusion of timing windows, vehicle capacity constraints, and limitations on the durations of routes. Laporte et al. (1985) developed an integer programming formulation for the VRP, and Achuthan and Caccetta (1991) and Hart (1992) developed MILP models.

2.2. Simulated Annealing

Simulated annealing is a computational process which attempts to solve difficult optimization problems through controlled randomization. The procedure was popularized by Kirkpatrick et al. (1983), and is based on work by Metropolis et al. (1953) (the so-called Metropolis algorithm) in statistical mechanics. Simulated annealing emulates the physical process of annealing (hence the name of the heuristic) which attempts to force a system to its lowest energy state through controlled cooling. In general, the annealing process involves the following steps:

1. The temperature of the system is raised to a sufficient level.
2. The temperature of the system is maintained at this level for a prescribed amount of time.
3. The system is allowed to cool under controlled conditions until the desired energy state is attained.

The initial temperature (Step 1), the time the system remains at this temperature (Step 2), and the rate at which the system is cooled (Step 3) are referred to as the *annealing schedule*. If the system is allowed to cool too fast, it may "freeze" at an undesirable, high energy state. With respect to optimization problems, the state of the system corresponds to the value of the objective

function. Similarly, the freezing of a system at an undesirable energy state corresponds to an optimization problem which is "frozen" at a local optima.

In simulated annealing the problem starts at some suboptimal solution or state, and a series of moves or transitions (changes of values of decision variables) are made according to a user-defined annealing schedule until either the optimal solution is attained or the problem becomes frozen at a local optima from which it cannot improve. Transitions between states occur randomly according to a problem-specific mechanism which defines how the algorithm selects a neighborhood of its current state. To avoid freezing at a local optima, the algorithm moves slowly (with respect to the objective value) through the solution space. This controlled improvement of the objective value is accomplished by accepting non-improving moves with a certain probability (based on the resulting change in the objective value and the current temperature) which decreases as the algorithm progresses.

The general procedure for implementing a simulated annealing algorithm follows:

Step 1. Select an initial temperature, t_0, and an initial solution, X_0. Let $f_0 = f(X_0)$ denote the corresponding objective value. Set $i = 0$, $k = 0$, $n = 0$, and go to Step 2.

Step 2. Set $i = i + 1$, and $n = n + 1$. Randomly generate a neighborhood solution, X_i and evaluate $f_i = f(X_i)$.

Step 3. If $f_i < f_{i-1}$, then go to Step 5. Otherwise, accept f_i as the new solution with probability $e^{\frac{-\Delta Z}{t}}$ and go to Step 4.

Step 4. If f_i was rejected as the new solution in Step 3, set $f_i = f_{i-1}$. Go to Step 5.

Step 5. If $n = L_k$ (i.e., the number of trials per temperature level), set $k = k + 1$, set $n = 0$, and set the temperature, t_k, according to the annealing schedule. If $t_k > t_{\min}$, then go to Step 2. Else, stop.

Although some theoretical guidelines exist for selection of SA parameters (see van Laarhoven and Aarts (1989) for selected references), most applied research employs experimental design to determine the most appropriate parameters for a given application. The parameter design method presented by Johnson et al. (1989) has been successfully applied and extended by Hart and Chen (1994) and Chen, Hart, and Tham (1995). Consequently, this method will be employed herein.

Many researchers have reported encouraging results from the application of simulated annealing to the solution of computationally complex problems. These applications include the mapping problem (Hart and Chen, 1994), general integer linear programming (Chen, Hart, and Tham, 1995), the graph partitioning problem (Johnson et al., 1989a), the traveling salesman problem (Cerny, 1985; and Kirkpatrick, 1984), global wiring (Vecchi and Kirkpatrick, 1983), and nonlinear optimization (Vanderbilt and Louie, 1984; Bohachevsky et al., 1986). A review of some of these applications can be found, for example, in Eglese

(1990).

3. Development of the model

3.1. Model formulation

This section describes the development of the multiple criteria mixed integer linear programming (MCMILP) model of the DAP. Formulation of the model's constraint set is described based upon a network representation of the DAP.

Let n denote the number of customers requiring transport. To each of these customers assign a unique node, v_i (hereafter, *pick up node*). Further, associate with each pick up node in the network a unique node, v_{i+n}, corresponding to the i^{th} customer's destination (hereafter, *destination node*). Define a network, $N = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n, v_{n+1}, \ldots, v_{2n}, v_{2n+1}\}$ and $E = [e_{ij}]_{(2n+2)\times(2n+2)}$. The vertices v_0 and v_{2n+1} denote the start node and terminal node of the network, respectively. Finally, let c_{ij} denote the minimum time required to travel from node i to node j (i.e., the edge cost).

The start node corresponds to the central depot. By this convention, a directed arc; e_{0j}, $j = 1, 2, \ldots, n$; exists corresponding to the travel time from the central depot to each of the customer pickup points. The terminal node corresponds to the post-service position for each vehicle. That is, after a vehicle completes service, it is routed to the terminal node. Similar to the start node, this convention requires that a directed arc; $e_{i,2n+1}$, $i = n+1, n+2, \ldots 2n$; exist corresponding to the travel time from each of the customer destination nodes to the terminal node. Note that customer pick up nodes are not adjacent to the terminal node, and that customer destination nodes are not adjacent to the start node. Otherwise, extraneous arcs and, hence, extraneous decision variables and constraints would be introduced into the MCMILP model.

The parameters and decision variables associated with the MCMILP model are defined below.

Model parameters and decision variables

n An integer constant denoting the number of customers in the system.

k An integer constant denoting the number of available vehicles.

$BC_{\max}$ An integer constant denoting the maximum capacity of each vehicle.

M A very large (i.e., $M \gg 1$) integer constant.

x_{ij} A Boolean decision variable which is 1 if a vehicle is routed from node i to node j, and 0 otherwise.

T_i A real decision variable which denotes the time at which any vehicle reaches node i.

T_{ij} A real decision variable which denotes the cumulative time to travel from node i to node j.

c_{ij} A real constant denoting the travel time from node i to node j.

B_i An integer decision variable denoting the number of customers on the vehicle when it reaches node i.

BC_{ij} An integer decision variable which denotes the remaining capacity of a vehicle as it traverses arc (i, j).

u_i An integer decision variable denoting the number of the vehicle which visited node i.

u_{ij} An integer decision variable denoting the number of the vehicle which traversed edge (i, j).

Since each customer is assigned a unique node, if more than one, say k, customer is waiting at the same pickup node, then the cost of all edges between the k nodes would be zero (In practice, the actual edge cost will be a sufficiently small number, $\epsilon \ll 1$). Similar logic is applied to the destination nodes.

Based upon the various classes of DAPs outlined in Section 2, development of a robust mathematical model capable of facilitating various combinations of said classes requires that the model accommodate, via decision variables and constraints, the following:

1. *Conservation of Flow.* Given that the model accommodates a central depot and postservice point, no vehicle should terminate service at a customer pick up/destination point. This restriction is easily incorporated into the model with conservation of flow constraints. These constraints will utilize the x_{ij} routing variables. The x_{0j} $(1 \leq j \leq n)$ also serve to establish the number of vehicles which were dispatched from the central depot, and to restrict the number dispatched to no more than the maximum number, k.
2. *Timing.* Timing (i.e., the time a vehicle arrives at a pick up or destination node) is crucial in establishment of customer wait times, customer travel times, and ensuring that a customer delivery occurs after said customer has been picked up. In the model, the T_i variables will reflect the time at which a vehicle arrived at a customer pick up/destination node. Because the network model assigns a unique node to every customer's pick up and destination point, T_i and T_{i+n} $(1 \leq i \leq n)$, will reflect the time customer i was picked up and delivered, respectively. Clearly, $T_i \leq T_{i+n}$ will ensure that the time at which a customer is picked up precedes the time at which said customer was delivered.
3. *Vehicle Capacity.* Since limited vehicle capacities are accommodated, the model must ensure that no more than the maximum number of customers are loaded at any time. This restriction is incorporated into the model via the B_i variables. By construction, B_i $(1 \leq i \leq n)$ will reflect the number of customers loaded on the vehicle which visits the i^{th} pick up node. Similarly, B_i $(n+1 \leq i \leq 2n)$ will reflect the number of customers loaded on the vehicle which visits the i^{th} destination node.
4. *Vehicle Number.* Given that there are some number, k, of vehicles in the system, a method of distinguishing between vehicles must be provided. Similar to the timing constraints, the model must ensure that the number of the vehicle which picked up a given customer corresponds to the number of the vehicle which delivered said customer. In the model, u_i will denote

the number of the vehicle which visited the i^{th} node. The model ensures that the vehicle number which picks up a customer matches the vehicle number which delivers said customer by requiring $u_i = u_{i+n}$.

The preceding discussion of the decision variables has assumed that each node in the network (except the start and terminal nodes) will be visited exactly once by one and only one vehicle. That is, a feasible solution to the MCMILP model of the network must correspond to an open-ended m-tour through the DAP network. Lemma 1 justifies this assumption.

LEMMA 3.1 *A path through the DAP network is feasible only if it constitutes an open-ended m-tour.*

Proof.

(a) *Open-ended tour.* All vehicles originate at the start node, node 0. By construction, the terminal node, node $2n + 1$, and the start node are disconnected. Hence, any TSP tour solution must be open-ended.

(b) *TSP tour.* To prove, assume contrary. That is, suppose there exists a feasible path which is not an open-ended tour. This implies one of the following:

(i) A node in the network is not visited.

(ii) A single vehicle visits a given pick up or destination node more than once.

(iii) More than one vehicle visits a given pick up or destination node.

Part (i). If a pick up or destination node is not visited, then a customer is either not picked up or is not delivered, respectively. Clearly, such a solution is infeasible.

Parts (ii) and (iii). By construction, a customer is assumed picked up or delivered when any vehicle visits a given pick up or destination node, respectively. (Recall that a unique pick up and destination node is assigned to each customer.) Therefore, since the edge cost denotes the minimum travel time between its incident nodes, parts (ii) and (iii) follow directly from the triangle inequality. That is, inclusion of a previously visited node in any vehicle's path will result in an increase in one or more customer's travel time, wait time, or both. ■

The restrictions and limitations of the MCMILP model and the criteria to be addressed in this research are detailed below:

Criteria

1. Minimize number of vehicles (see (18)).
2. Minimize average customer travel time (see (19)).
3. Minimize average customer wait time (see (20)).
4. Minimize maximum customer wait time (see (21)).
5. Minimize maximum customer travel time (see (22)).

Restrictions/Limitations

1. Although the model will facilitate restrictions on maximum vehicle capacity, the maximum capacity of all vehicles must be identical.
2. The model is deterministic.
3. All vehicles initially depart from the same location.

In lieu of Lemma 1, the MCMILP constraint set must be formulated such that every feasible solution corresponds to an open-ended m-tour. To this end, the constraint set for the MCMILP model is depicted in (1) through (14).

MCMILP Model Constraint Set

$$\sum_{j=1}^{n} X_{0j} \leq k \tag{1}$$

$$\sum_{i=p\ i\neq j}^{2n} X_{ij} = 1\ j = 1,2,\ldots,2n;\ p = \begin{cases} 0 & \text{if} \quad j \leq n \\ 1 & \text{if} \quad j > n \end{cases} \tag{2}$$

$$\sum_{j=1\ j\neq i}^{2n+p} X_{ij} = 1\ j = 1,2,\ldots,2n;\ p = \begin{cases} 0 & \text{if} \quad j \leq n \\ 1 & \text{if} \quad j > n \end{cases} \tag{3}$$

$$T_{ij} = c_{ij}X_{ij} + T_i\ i \neq j\ i = 0,1,\ldots,2n;\ j = \begin{cases} 1,2,\ldots,n & \text{if} \quad i = 0 \\ 1,2,\ldots,2n & \text{if} \quad i > 0 \end{cases} \tag{4}$$

$$T_i \geq T_{ij} - < (1 - X_{ij})\ i \neq j\ i = 0,1,\ldots,2n; \tag{5}$$

$$j = \begin{cases} 0,1,\ldots,2n & \text{if} \quad i \leq n \\ 1,2,\ldots,2n & \text{if} \quad i > n \end{cases}$$

$$T_i \leq T_{i+n}\ i = 1,2,\ldots,n \tag{6}$$

$$BC_{ij} = \delta_i X_{ij} + B_i\ i = 0,1,\ldots,2n;\ j = 1,2,\ldots,2n; \tag{7}$$

$$\delta_i = \begin{cases} 1 & \text{if} \quad i \leq n \\ -1 & \text{if} \quad i > n \end{cases}$$

$$B_i \leq BC_{\max}\ i = 1,2,\ldots,2n \tag{8}$$

$$B_i \geq BC_{ji} - M(1 - X_{ij})\ i \neq j;\ i = 1,2,\ldots,2n \tag{9}$$

$$u_i \geq ix_{0i}\ i = 1,2,\ldots,n \tag{10}$$

$$u_{ij} = u_i\ i \neq j;\ i = 1,2,\ldots,2n \tag{11}$$

$$u_{ij} - M(1 - X_{ij}) \leq u_i \leq M(1 - X_{ij}) + u_{ij}\ i \neq j;\ i = 1,2,\ldots,2n \tag{12}$$

$$u_i \leq M(1 - X_{ij}) + ix_{0i}\ i = 1, 2, \ldots, n \tag{13}$$

$$u_{i+n} = u_i\ i = 1, 2, \ldots, n \tag{14}$$

$x_{ij} \in \{0, 1\}$ for all i, j

Following are the short descriptions of the function of each of the constraints of the MCMILP model:

(1) This constraint ensures that the number of vehicles dispatched from the central depot (node 0) does not exceed the maximum number available.

(2)-(3) These constraints together ensure that exactly one vehicle arrives at and departs from all pick up and delivery nodes.

(4)-(6) These constraints establish customer pick up and delivery times (equivalently, vehicle arrival and departure times). Assuming that all vehicles depart the central depot at relative time zero (i.e., $T_0 = 0$), then T_{0j} $(1 \leq j \leq n)$ will reflect the time it takes a vehicle to travel from the central depot to the j^{th} pickup node. If a vehicle actually travels this path (i.e., $x_{0j} = 1$), then constraint (5) will establish the vehicle arrival time at the j^{th} node. Note that constraints (4) and (5) are working in conjunction with constraints (2) and (3) ensuring there is no conflict in the establishment of arrival times by constraints (5).

Once a given vehicle's arrival time has been established at the first node it visits (i.e., $j \leq n$), the arrival time at the next node in its path can be established in the same manner as previously described.

As previously mentioned, constraints (6) will ensure that a customer's delivery time never precedes his pick up time.

(7)-(9) These constraints collectively track the number of customers on each vehicle throughout their respective tours. Assuming that all vehicles depart the central depot with zero customers (i.e., $B_0 = 0$), then BC_{0j} $(1 \leq j \leq n)$ will reflect the addition of one customer to the vehicle which travels to the j^{th} node. If a vehicle actually travels this path (i.e., $x_{0j} = 1$), then constraint (9) will account for the addition of one customer to the vehicle. The constant, δ_i $(0 \leq i \leq 2n)$, in (7) controls whether a customer will be loaded or unloaded at the j^{th} node. Therefore, similar to the timing constraints, constraints (2) and (3) are working in conjunction with (7) and (9) to track the number of customers on each vehicle throughout their respective tours while (8) ensures no vehicle exceeds its maximum capacity.

(10)-(13) These constraints track the number of the vehicle which visits each node in the network. Constraints (10) and (13) work together to ensure that a unique number is assigned to every vehicle which departs the central depot. By convention, each vehicle departing the central depot is assigned a number corresponding to the first pick up node it visits. This scheme ensures a unique number for each vehicle. Constraints (11) and (12) establish the number(s) of the vehicle(s) arriving at all destination

nodes and at all pick up nodes from a node other than the central depot. Finally, constraints (14) ensure that the number of the vehicle which picks up a given customer corresponds to the number of the vehicle which delivers said customer.

Other It should be noted that timing windows are easily incorporated into the constraint set through addition of constraints of the form

$$T_{\min} \leq T_i \leq T_{\max}$$

where $T_{\min}$ denotes the earliest relative pick up or delivery time and $T_{\max}$ denotes the latest relative pick up or delivery time.

Finally, note that satisfaction of the Boolean restriction on the x_{ij} variables necessarily ensures the integrality of the B_i and u_{ij} variables; hence, the integral restriction on these decision variables can be relaxed.

Lemma 1 established that a feasible solution to the DAP must constitute an open-ended m-tour. It remains to be shown that every solution which satisfies the MCMILP constraint set of (1) through (14) corresponds to an open-ended m-tour.

LEMMA 3.2 *Every solution which satisfies the MCMILP constraint set corresponds to an open-ended m-tour.*

Proof. To prove, assume contrary. That is, suppose there exists a solution satisfying (1) through (14) which is not an open-ended m-tour. This implies one or more of the following:

(i) One or more nodes are not visited.
(ii) One or more nodes (excluding the terminal node) are visited more than once.
(iii) One or more subtours exist in the solution.
(iv) One or more paths exists in the solution which do not contain either the start node (node 0) or the terminal node (node $2n+1$) or both.

Parts (i) and (ii). The constraints of (2) ensure that every pick up and destination node is visited once and only once.

Part (iii). Since all arcs incident with the start node and terminal node are unidirectional, any subtours must occur within the pick up and destination nodes. Hence, a subtour could consist of a cycle containing only pickup nodes, a cycle containing only destination nodes, or a cycle containing both pick up and destination nodes. Without loss of generality, let the cycle (i.e., subtour) be denoted by the set of arcs $S = \{(1,2),(2,3),\ldots,(n-1,n),(n,1)\}$. Given that constraints (2) and (3) are satisfied, such a cycle would require:

$$T_{12} = c_{12} + T_1 \text{ (by (4))}$$

$$T_1 \geq T_{n1} = c_{n1} + T_{n-1} \text{ (by (5))} \tag{15}$$

$$T_2 \geq T_{12} = c_{12} + T_1 \text{ (by (5))} \tag{16}$$

Substituting (15) into (16) we have:

$$T_2 \geq c_{12} + T_{n-1} + c_{n1} \text{ or } T_{n-1} \leq T_2 - c_{n1} - c_{12} \tag{17}$$

But, since $c_{ij} > 0$ for every edge $(i,j) \in S$, we have $T_l > T_m$ for all $l > m$; hence, (17) is contradicted, and a subtour of any length is infeasible.

Part (iv). The constraints of (3) ensure that a path away from every pick up and delivery node exists. Given that no solutions satisfying (1) through (14) contain subtours, it follows that every feasible path must contain both the start and terminal nodes. ■

It is clear that both the number of constraints and the number of variables in the MCMILP model are a polynomial function of the number of customers in the system. While this is a desirable feature, it is clear that the model would become intractable when solved by typical integer programming techniques as the number of customers in the system increased. (e.g., When $n = 100$, there are 201,100 constraints and 160,200 decision variables.)

At this point, formulation of the five objective functions corresponding to the aforementioned criteria are presented.

$$\min \sum_{j=1}^{n} X_{0j} \text{ (Number of vehicles)} \tag{18}$$

$$\min \frac{1}{n} \sum_{i=1}^{n} (T_{i+n} - T_i) \text{ (Average customer travel time)} \tag{19}$$

$$\min \frac{1}{n} \sum_{i=1}^{n} T_i \text{ (Average customer wait time)} \tag{20}$$

$$\min \max_{1 \leq i \leq n} \{T_i\} \text{ (Maximum customer wait time)} \tag{21}$$

$$\min \max_{1 \leq i \leq n} \{(T_{i+n} - T_i)\} \text{ (Maximum customer travel time)} \tag{22}$$

Objective function (21) is facilitated in the MCMILP model through the introduction of n additional constraints of the form:

$$z_1 \geq T_i \; i = 1, 2, \ldots, n$$

and (21) expressed as:

$$\min z_1$$

Similarly, objective function (22) is facilitated through the introduction of n constraints of the form:

$$z_2 \geq T_{i+n} - T_i \; i = 1, 2, \ldots, n$$

and (22) expressed as:

$$\min z_2$$

Finally, it should be noted that minimization of the total vehicle travel time (or distance) can be accommodated through introduction of an objective function of the following form:

$$\min \sum_{i=n+1}^{2n} (X_{i,2n+1})T_i$$

This nonlinear objective function can be facilitated in the model through the introduction of n additional constraints of the form:

$$z_{i-n} + (1 - X_{i,2n+1})M \geq T_i \; i = n+1, n+2, \ldots, 2n; \; M \gg 1$$

and corresponding objective function:

$$\min \sum_{i=1}^{n} Z_i$$

The complete goal program is depicted in (23) through (31).

$$\min \sum_{i=1}^{4} p_i^+ d_i^+ + \sum_{i=1}^{n} X_{oi} \tag{23}$$

subject to

$$\sum_{i=1}^{n} (T_{i+n} - T_i) - d_1^+ + d_1^- = g_1 \tag{24}$$

$$\sum_{i=1}^{n} T_i - d_2^+ + d_2^- = g_2 \tag{25}$$

$$z_1 - d_3^+ + d_3^- = g_3 \tag{26}$$

$$z_2 - d_4^+ + d_4^- = g_4 \tag{27}$$

$$z_1 \geq T_i \; i = 1, 2, \ldots, n \tag{28}$$

$$z_2 \geq T_{i+n} - T_i \; i = 1, 2, \ldots, n \tag{29}$$

$$T_{\min_i} \leq T_i \leq T_{\max_i} \text{ for all } i = 1, 2, \ldots, 2n \tag{30}$$

such that T_i has timing windows

$$\text{Constraints (1) through (14)} \tag{31}$$

In the goal programming formulation, the p_i^+ variables are user-specified constants corresponding to the weight associated with each deviational or goal variable. The g_j constants are the user-specified goals for each of the decision criteria. For reference, the correspondence between the preceding goal constraints (i.e., (24) through (29)) and the objective criteria (i.e., (18) through (22)) is as follows:

- (18) appears in the objective function
- (24) corresponds to (19)
- (25) corresponds to (20)
- (26) and (28) correspond to (21)
- (27) and (29) correspond to (22)

3.2. Implementation issues

In the MILP formulation, the number of vehicles in the system is established via the constraints of (1). However, the solution algorithm developed in Section 4 employs a route construction technique based upon the model's network formulation (i.e., the MILP model is not solved directly.) Therefore, to facilitate multiple vehicles in the solution, it will be convenient to add N (where N denotes the number of customers in the system) additional nodes to the network presented in Section 3. These nodes will designate the introduction of an additional vehicle into the solution.

For example, consider a DAP with $N = 2$. Then there are $3N + 2 = 8$ nodes in the network formulation (see Figure 1).

Nodes 1 and 2 denote customer pick up nodes, nodes 3 and 4 denote customer delivery nodes, and nodes 5 and 6 denote vehicle transition nodes. (As previously discussed, nodes 0 and $3N + 1$ denote the home depot and vehicle termination point, respectively.) Consequently, the route given by solution 0-1-3-2-4-7 denotes a single-vehicle solution; whereas, the route given by solution 0-1-3-5-2-4-7 denotes a two-vehicle solution.

Since any path which includes a vehicle transition node which is adjacent to a customer pick up node denotes the introduction of an additional vehicle into the solution, the arc costs from the transition nodes to the customer pick up nodes correspond to the arc costs from the central depot (node 0) to the customer pick up nodes. This convention is illustrated in Figure 1. That is, no cost is incurred in the solution through introduction of a given transition node until said node is connected to a customer pick up node. Consequently, as shown in Figure 1, the arc costs from the customer delivery nodes to the transition nodes as well as the arc costs between transition nodes is zero. (It should be noted that inclusion of a transitional node in any given route increases the cost of said route exactly by that amount associated with the introduction of the corresponding vehicle into the solution; therefore, introduction of the transistional nodes represents an equivalent network formulation and does not affect the results of Lemma 1.) Given the aforementioned network structure, it

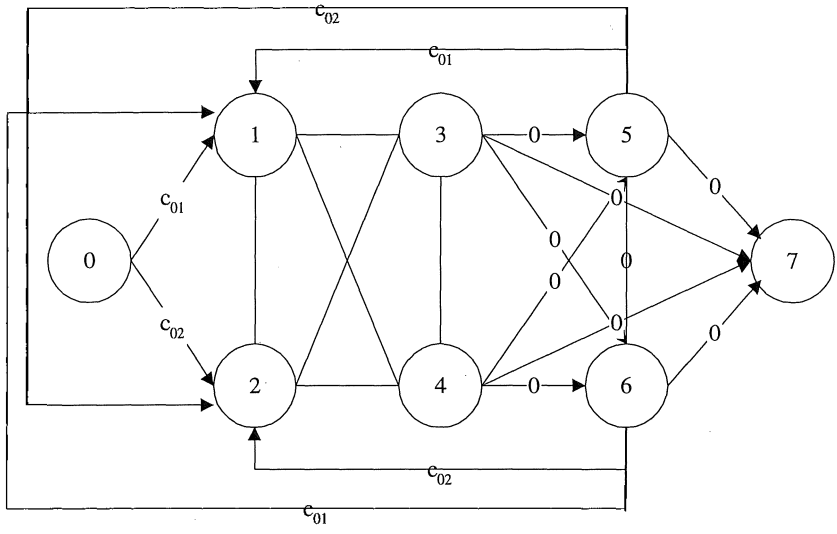

Figure 1. SADAPA network formulation example

follows that the two-vehicle solutions corresponding to the paths listed below:

- 0-1-3-5-2-4-6-7
- 0-1-3-6-2-4-5-7
- 0-1-3-5-6-2-4-7
- 0-1-3-6-5-2-4-7

are equivalent with respect to optimal solution and corresponding optimal objective value. Specifically, all four solutions denote the following vehicle routes:

Vehicle 1: 0-1-3-7 (Depart home depot. Pick up customer 1. Deliver customer 1. Terminate route.)

Vehicle 2: 0-2-4-7 (Depart home depot. Pick up customer 2. Deliver customer 2. Terminate route.)

4. The simulated annealing algorithm

The SA algorithm employed in this research followed the general outline presented in Section 2. The algorithm is initiated with a temperature of $t =$ TINIT. The temperature is lowered at each phase of the algorithm through multiplication with TFAC after every ITER iterations of the algorithm. The algorithm is terminated when one of the following occurs:

1. The temperature falls below the user-specified threshold, TSTOP.

2. The percentage of accepted moves at any temperature level is below the user-specified threshold, MINACCEPT, for a user-specified number, FREEZEMAX, of consecutive temperature levels.

A description of the algorithm's parameters is provided below.

Input Parameters

TINIT A real value denoting the initial or starting temperature.

TFAC A real value ($0 < TFAC < 1$) denoting the temperature multiplication factor.

TSTOP A real value denoting the terminating temperature level.

MINACCEPT A real value denoting the threshold or minimum number of allowable accepted moves at a given temperature level.

SIZEFACT A real value used to establish the number of trials per temperature level.

Program Parameters

NSIZE The number of neighborhood solutions to a given trial solution.

ITER The number of trial solutions per temperature level.

REJECTED The number of rejected trial solutions at a given temperature level.

ACCEPTED The number of accepted trial solutions at a given temperature level.

FAILCOUNT The number of consecutive temperature levels for which the minimum number of accepted moves was less than MINACCEPT.

t The current temperature level.

n The number of remaining trial solutions for a given temperature level.

The design of the SA algorithm is based upon a separation of the feasibility requirements expressed in the MCMILP model into two components: (1) *TSP Feasible*: Every feasible solution to the MCMILP model constitutes an open-ended m-tour. (2) *Scheduling, Routing, and Timing (SRT) Feasible*: Every feasible solution satisfies the timing and routing constraints in (4) through (14).

Determination of a TSP/SRT feasible solution is based on a series of rejection rules. That is, if for a given incumbent node any of the following rules are satisfied, then the corresponding trial solution is infeasible.

Rejection Rules

Rule 1. The incumbent node would result in termination of the algorithm before all pick up and delivery nodes had been visited. (Associated with constraints (2)-(3))

Rule 2. The incumbent node would result in a subtour. (Associated with constraints (2)-(3) and (4)-(6))

Rule 3. The incumbent node would result in a violation of the maximum vehicle capacity. (Associated with constraints (7)-(9))

Rule 4. The incumbent node would result in a vehicle visiting a delivery node prior to vistiting the associated pick up node. (Associated with constraints

(4)-(6))

Rule 5. The incumbent node would result in the incorrect vehicle number visiting a destination node. (Associated with constraints (10)-(14))

Rule 6. The incumbent node results in a nonempty vehicle visiting a transition node.

The first TSP feasible trial solution is generated by building an open-ended TSP tour consisting of the following ordered set of edges: (Throughout the discussion in this section, unless otherwise stated, N will denote the number of customers in the system.)

$$\{(0,1),(1,N+1),(N+1,2),\ldots,(2N-1,N),(N,2N),$$
$$(2N,2N+1),(2N+1,2N+2),\ldots,(3N,3N+1)\}.$$

Note that this trial solution satisfies all SRT feasibility constraints in (1)-(14) and is therefore a feasible solution to the MCMILP model. Subsequent trial solutions are generated through pairwise interchange of two nodes in the network. This interchange is a general pairwise interchange procedure. That is, the interchange of two nodes in the network, (i,j), is performed as: $(i,j) = (1,2),(1,3),\ldots,(1,3N),(2,1),(2,3),\ldots,(2,3N),\ldots,(3N-1,3N)$. This procedure was selected based upon experimentation with various neighborhood selection schemes including pairwise interchange of randomly selected nodes and adjacent pairwise interchange. It should be noted that Potts and van Wassenhove (1991) reached the same conclusion with respect to selection of neighborhood solutions in their study of simulated annealing solutions for the single machine tardiness sequencing problem.

The simulated annealing solution algorithm for the MCMILP is as presented below.

Simulated Annealing DAP Algorithm (SADAPA)

Step 1. Set $t =$ TINIT. Generate the initial trial solution and compute the corresponding objective value, f_0. Set BESTOBJ $= f_0$. Set ITER $=$ (NSIZE)(SIZEFACT) then set $n =$ ITER.

Step 2. Set $i = i+1$. Generate a new trial solution using the general pairwise interchange procedure and, if feasible, calculate corresponding objective value, f_i. Else set $f_i = 1.5$BESTOBJ.

Step 3. If $f_i < f_{i-1}$, then set FAILCOUNT $= 0$, BESTOBJ $= f_i$, and go to Step 5. Otherwise, accept f_i as the new solution with probability $e^{\frac{-\Delta z}{t}}$ and go to Step 4.

Step 4. If f_i was rejected in Step 3, set $f_i = f_{i-1}$ and set REJECTED $=$ REJECTED+1. If f_i was accepted in Step 3, set ACCEPTED $=$ ACCEPTED $+ 1$. Go to Step 5.

Step 5. Set $n = n-1$. If $n > 0$, then go to Step 2. Else set $t = (t)(TFAC)$. If $t < TSTOP$, then stop. Else if $\left(\frac{\text{ACCEPTED}}{\text{ACCEPTED+REJECTED}}\right) \leq$ MANACCEPT, then set FAILCOUNT $=$ FAILCOUNT+1. If FAILCOUNT $=$

FREEZEMAX, then stop; otherwise, set n = ITER, set REJECTED = 0, reinstall best solution, reset interchange variables and go to Step 2.

The algorithm was implemented in the C programming language using the Borland Turbo C version 2.0 compiler, and all experiments were conducted on an IBM-compatible 486DX2 50 MHz system.

In Step 2 of the algorithm, infeasible trial solutions are assigned a penalty cost of 1.5BESTOBJ, where BESTOBJ is the best feasible objective value so far obtained. The selection of the multiplication factor of 1.5 was based on experimentation with values ranging from 1.1 to 2.0. Experiments were also conducted in which all infeasible trial solutions were rejected. However, this procedure performed poorly. The poor performance of unconditional rejection of infeasible solutions follows since the general pairwise interchange procedure is implemented as a pair of nested loops. That is, in many cases movement from one feasible trial solution to another via the implemented interchange procedure requires "passing through" an infeasible trial solution. This can be easily verified through manual solution of a small test problem.

After completion of each temperature level, the best solution so far obtained is reinstalled. Additionally, after the completion of each temperature level, the node interchange variables are reset to their initial value. That is, the interchange of two nodes in the network, (i, j), is performed as $(i, j) = (1, 2), (1, 3), \ldots (1, 3N), (2, 1), (2, 3), \ldots$ from the initiation of each new temperature level.

The experiments conducted to determine the "ideal" parameter settings for the annealing algorithm are described. The test case employed in analysis of the effects of the annealing algorithm's parameter set was designed so that the optimal solution was known. The parameters were analyzed using a 20 customer test problem, and the robustness of the chosen parameters was studied through solution of 30 and 40 customer problems.

The experimental procedures followed were adapted from the work of Johnson et al. (1989) and Hart and Chen (1994). It should be noted that the number of parameters, the range of values each can assume, and their inherent interactions prohibit a complete enumeration of all possible combinations. However, the experiments conducted herein provide a systematic approach which can be duplicated by other researchers. Moreover, the approach has been shown by other researchers, as reported in the literature, to adequately explore the relationships between the various parameters.

The parameters of interest are: TINIT, TFAC, MINACCEPT, SIZEFACT, and NSIZE. An additional parameter of interest is INITPROB. This is a dependent variable representing the probability of accepting a trial solution during the first temperature level. In SADAPA, INITPROB is expressed as the proportion of nonimproving moves accepted by the algorithm during a given temperature level (see Step 5 of the algorithm). Clearly, this parameter is dependent upon TINIT, NSIZE, and SIZEFACT.

The variable, NSIZE, corresponds to the number of neighborhood solutions associated with a given trial solution. Ideally, NSIZE is set to the actual neighborhood size and is used in conjunction with SIZEFACT to control the number of trials per temperature level. Since SADAPA employs a general pairwise interchange procedure, NSIZE is defined as follows:

$$\text{NSIZE} = 3N(3N-1)$$

The first experiment required determination of the relationship between TINIT and the acceptance probability, INITPROB. In the spirit of Johnson et al. (1989) and Hart and Chen (1994), multiple runs of 20 replications each were conducted for various values of TINIT. The test case consisted of a DAP with 20 customers, a fleet size of 20 vehicles, and a vehicle capacity of 10 customers. The criteria selected was the minimization of the total vehicle travel distance, L.

The test case was designed so that the optimal solution, L^*, was known. Specifically, the travel distance from the home depot, node 0, to each of the customer pickup nodes, $1, 2, \ldots, N$, was set to 1. Since a solution including a node number larger than $2N$ denotes introduction of an additional vehicle into the solution, the travel distances from each of the nodes $2N+1, 2N+2, \ldots 3N$ to the customer pickup nodes was also set to 1, and the distance between any pair of nodes $2N+1, 2N+2, \ldots, 3N$ was set to zero.

For every pair of pickup nodes (i, j) such that $2 \leq (i, j) \leq N$ and $i \neq j$, the distance from node i to node j, and hence from j to i, was set to $|i-j|$. For example, the distance from node 4 to node 10 would be 6. The distance from node 1 to node N was set to 1. The same scheme was employed for the delivery nodes, $N+1, N+2, \ldots, 2N$.

The travel distance from a given pickup node to its corresponding delivery node was set to $60N$ (the choice of 60 was arbitrary), and the distance between all other pairs of pickup/delivery nodes was set at their Euclidean distance. Finally, all infeasible arcs (e.g., self-loops) were set to -1.

Given the aforementioned network structure, the optimal solution is given by

$$L^* = \left\lfloor \frac{N}{BMAX} \right\rfloor (2(BMAX) - 1 + 60N) + \left(\left\lceil \frac{N}{BMAX} \right\rceil - \left\lfloor \frac{N}{BMAX} \right\rfloor \right) \left(2\left(N - \left\lfloor \frac{N}{BMAX} \right\rfloor BMAX \right) - 1 + 60N \right)$$

where $BMAX$ denotes the maximum vehicle capacity. (These values are: 2438, 5457, and 9676 for 20, 30, and 40 customer problems with $BMAX = 10$, respectively.) A formal proof of the validity of L^* is omitted; however, it should be clear that the magnitude of the travel distance from any pick up node $i \leq N$ to any delivery node $N+1 \leq j \leq 2N$ forces an optimal solution to minimize the number of vehicles dispatched. That is, in the optimal solution, every vehicle

TINIT	—	1	10	170	210	350	525	800	1300
INITPROB	.1	.2	.3	.4	.5	.6	.7	.8	.9

Table 1. Relationship between initial temperature (TINIT) and the acceptance probability (INITPROB)

	TINIT/INITPROB							
	1/.2	10/.3	170/.4	210/.5	350/.6	525/.7	800/.8	1300/.9
Average Objective Value	3599	2827	2467	2530	2461	2458	2529	5579
Standard Deviation	471	556	23.3	284	9.5	7.6	284	5995
Average CPU Time (secs)	8.3	124.2	408.5	387.5	452.4	477.9	554.2	612.6
Best Objective Value	2462	2458	2452	2452	2445	2446	2443	2443

Table 2. Effects of starting temperature (TINIT) and the acceptance probability (INITPROB)

dispatched will pick up and deliver the maximum feasible number of customers. (To prove, assume contrary: Assume the optimal solution consists of fewer vehicles than that given by L^* which yields either an infeasible solution due to the capacity limit $BMAX$ or a solution with a higher objective value due to the requirement that one or more vehicles make multiple pick up and deliver runs. Alternatively, assume the optimal solution consists of more vehicles than that given by L^*. Again, a contradiction is reached due to vehicle travel distances between pick up and delivery nodes.)

For each replication, the algorithm was terminated after one full iteration (equivalently, one temperature level) which was fixed at NSIZE iterations (i.e., SIZEFACT = 1.0). The percentage of accepted moves was calculated and averaged over the 20 replications, and a correlation between TINIT and the acceptance probability, INITPROB, was established (see Table 1).

To determine the effect of INITPROB on the algorithm's efficiency, 20 replications were performed on the DAP with a vehicle capacity of 10 customers for each of the TINIT values in Table 2.

For these experiments, SIZEFACT, MINACCEPT, TFAC, FREEZEMAX, and TSTOP were fixed at 1.0, .2, .95, 1, and .001, respectively. (Hereafter the aforementioned set of parameters will be referred to as the *standard parameters.*) The results of this experiment are depicted in Table 2.

In the table, the average final objective value and average execution time obtained over the 20 replications is depicted. From the table, the best performance with respect to average objective value and variability of solutions obtained occurs when INITPROB = .7.

Customers	TINIT	Best Objective Value	CPU Time (secs)	Efficiency
30	1200	5457	3563	100%
40	1950	9696	6252	99.8%

Table 3. Results for 30 and 40 customer problems

Although the best overall solution was obtained when INITPROB = .8 and INITPROB = .9, the significant difference in CPU times, average objective values, and variability of solutions obtained supported selection of INITPROB = .7 into the standard parameter set. Further support for this selection was provided by experiments with 30 and 40 customer problems. The results of these experiments are depicted in Table 3.

(NOTE: The TINIT values listed in the table are those required to yield an average INITPROB = .7 across 20 replications of the algorithm.)

From the table, it is clear that the performance, with respect to efficiency of solution, is robust across various problem sizes.

The effects of TFAC and SIZEFACT were explored by performing 10 replications of the algorithm using the standard parameters and the TFAC, SIZEFACT combinations depicted in Table 4.

TFAC has been chosen such that each increase corresponds to taking the square root of the previous value – a choice that will double the number of temperature levels at which the function is evaluated for a specified range. Similarly, each increase in SIZEFACT doubles the previous value; therefore, since the number of trials per temperature level is the product of SIZEFACT and NSIZE, this procedure will also result in a doubling of the number of trials over a specified range. As noted by Johnson et al. (1989), fixing either parameter and increasing the other to its next value should yield an approximate doubling of the algorithm's execution time.

From Table 4, for a fixed value of SIZEFACT, an approximate doubling of the algorithm's execution time is observed between adjacent values of TFAC in most cases. However, for a fixed value of TFAC, doubling SIZEFACT results in an approximate doubling of the algorithm's execution time only when the increase is from SIZEFACT = .25 to SIZEFACT = .5. In all other cases, the data indicates an approximate three to four fold increase in the algorithm's execution time.

Based on these observations, it can be concluded that doubling the number of temperature levels has no significant effect on the algorithm's acceptance probability at each temperature level; hence, the temperature at which the algorithm is declared frozen remains constant. This result is illustrated in Figure 2 which compares the acceptance probability and best objective value obtained at each temperature level for the SIZEFACT = 1.0, TFAC = .8145

	Average Objective Value Standard Deviation Average CPU Time (secs) Best Objective Value				
	TFAC				
SIZEFACT	.6634	.8145	.9025	.9500	.9747
0.25	3714 8.9 3.5 3696	3351 552 22.6 2535	2882 545 32.7 2525	3243 1167 71.3 2491	2758 491 115.1 2488
0.50	2745 495 23.4 2491	2876 550 42.9 2506	2529 16.6 63.9 2505	2642 361 131.2 2514	3117 1886 283.9 2491
1.00	2827 570 72.5 2455	2575 375 140.2 2450	2457 8.4 253.1 2448	2457 7.25 486.7 2450	2452 7.07 949.9 2443
2.00	2442 5.14 250.5 2438	2442 6.4 393.9 2438	2440 2.78 968.3 2438	2442 7.99 1604 2438	2438 .713 2950 2438
4.00	2446 8.23 538.1 2438	2440 5.0 1009 2438	2443 8.74 1887 2438	2440 693 3710 2438	2438 .02 7343 2438

Table 4. Effects of TFAC and SIZEFACT

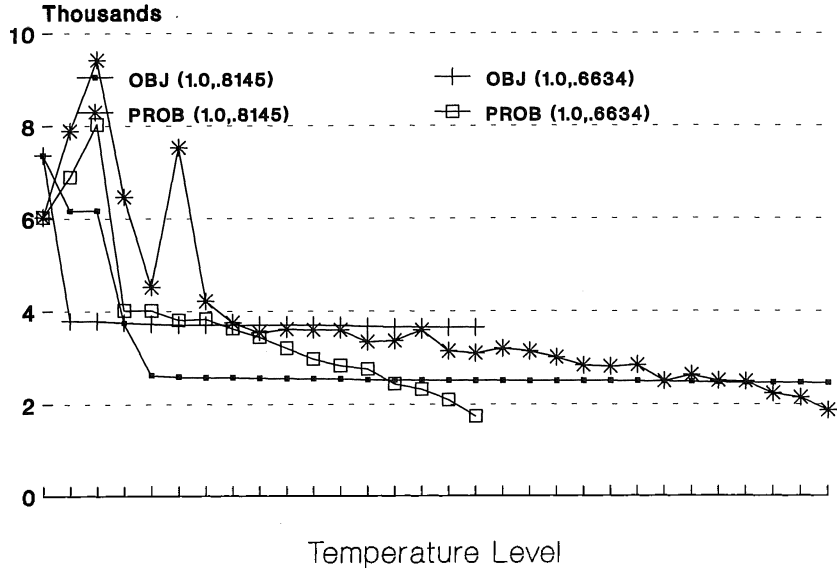

Figure 2. Effects of doubling the number of temperature levels

and SIZEFACT = 1.0, TFAC = .6634 pairs.

On the other hand, doubling the number of trials per temperature level seems to significantly alter the associated acceptance probability and therefore increase the run length or convergence time of the algorithm. To illustrate this, a single replication of the algorithm was performed using the TFAC = .8145, SIZEFACT = 1.0 and TFAC = .8145, SIZEFACT = 2.0 pair. For this experiment, the acceptance probability and best objective value were recorded for each temperature level as was described for the results of Figure 2. (The acceptance probabilities in both Figure 2 and Figure 3 have been scaled by a factor of 1000.) These results are illustrated in Figure 3.

From the figure, it is clear that increasing the number of trials per temperature level has yielded an increase in the acceptance probability at each temperature level and has therefore resulted in an increase in the algorithm's execution time which exceeds the predicted doubling.

Both Figure 2 and Figure 3 serve to illustrate that the majority of progress (with respect to objective value) attained by the algorithm occurs within the first eight temperature levels. Equivalently, the majority of progress is attained when the acceptance probability is greater than approximately .4. This observation is significant in that it indicates that "good" solutions (i.e., those with an efficiency of 95% or greater) can be achieved early in the annealing schedule; therefore, if an efficiency of 95% is acceptable, the annealing schedule can be truncated

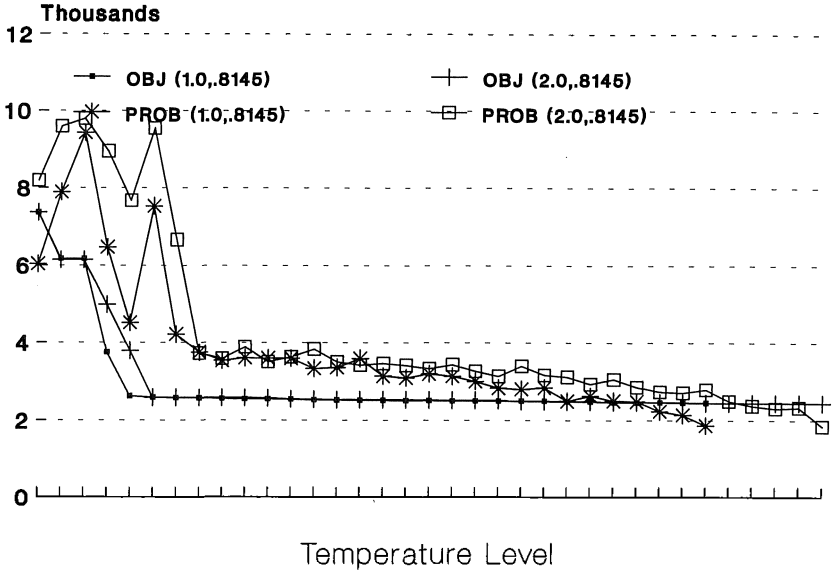

Figure 3. Effects of doubling the trials/temperature level

resulting in a significant decrease in the algorithm's execution time.

Returning to Table 4, the selection of the TFAC, SIZEFACT pair for entry into the standard parameter set should be based on the requirements for a given application. That is, short convergence time with average and best efficiencies exceeding 95% can be obtained using the TFAC = .9025, SIZEFACT = .5 pair. This pair was selected over other pairs yielding comparable efficiencies do to the low variability of solutions obtained. If, on the other hand, optimality is desired, then SIZEFACT should be selected from either SIZEFACT = 2.0 or SIZEFACT = 4.0 since at least one optimal solution was obtained over 10 replications for every associated TFAC value.

Analysis of the effects of the standard parameter set on the efficiency and accuracy of solutions was based on a single criteria: minimization of total vehicle travel distance. Since it is possible that the quality of solutions obtained in these experiments was associated with the structure of the network model employed , additional experiments were conducted on the same network model previously presented using different criteria. The intent was to design experiments for which optimality was attained in different regions of the solution space. (The standard parameters employed in these experiments are listed in Table 5.)

The criteria employed in these experiments were (i) Minimization of total travel distance, L. (Since the standard parameters employed differed from those used in the previous experiments, it was decided, for completeness, to repeat the

INITPROB	TFAC	SIZEFACT	FREEZEMAX	MINACCEPT	TSTOP
.7	.9025	2.0	1	.2	.001

Table 5. Standard parameters for robustness experiments

Criteria	Customers	Best Solution	Optimal Solution
$\min L$	20	2438	2438
	30	5457	5457
	40	9676	9676
$\min AW$	20	1	1
	30	1	1
	40	1	1
$\min k \ s.t. AW \leq 1.5$	20	2	2
	30	3	3
	40	5	4

Table 6. Experimental results for various criteria

same experiments.) (ii) Minimization of average wait time, AW. (iii) Minimization of the number of vehicles dispatched, k, subject to a maximum allowable customer wait time which was set at 1.5 times the number of customers in the system. The experiments were conducted on 20, 30, and 40 customer problems, and as mentioned, used the same network structure previously described. For each problem instance, three replications of the experiment were performed. The results of these experiments appear in Table 6.

In the table, the "Best Solution" column corresponds to the best solution obtained by the algorithm across the three replications; whereas, the "Optimal Solution" column corresponds to the global optimal solution for the given problem. The results reported in Table 6 support the claim of robustness for the simulated annealing based solution heuristic.

5. Summary and conclusions

This research has examined the feasibility of encapsulating into a single mathematical model and corresponding solution methodology the dial-a-ride problem in its various forms. To this end, a mixed integer linear programming model of the DAP was developed which incorporates various constraints and objectives commonly occurring in DAP models. A rigorous presentation and mathematical justification of the model was provided.

The relationship between the mathematical model and its corresponding net-

work formulation was utilized to develop a simulated annealing based solution heuristic. Design and implementation issues associated with the *SA* algorithm were discussed, and the performance of the algorithm was analyzed for various instances of the *DAP*. This analysis indicates that solutions with efficiencies exceeding 95% can be obtained within reasonable computation time for small to medium-sized problems.

In summation, the major contributions of this research are two-fold as outlined below:

1. A mathematical model of the dial-a-ride problem has been developed and justified which encompasses multiple instances of the problem, and which can facilitate various single objectives or multiple objectives expressed as a goal program. Although the model's size is prohibitively large for solution by standard optimization packages, the existence of the model should assist other researchers in the development of efficient solution algorithms.
2. An efficient simulated annealing based solution heuristic was developed based on the mathematical model and its corresponding network representation. The algorithm was shown to provide efficient solutions in reasonable computation time on small to medium-sized instances of the problem. The ability to efficiently solve the problem using the algorithm presented herein is not only significant in and of itself, but the ability to represent and efficiently solve the complex mathematical model of the problem using an equivalent network formulation suggests that other existing heuristic techniques such as tabu search or genetic algorithms may provide a desirable alternative methodology over simulated annealing.

Acknowledgment. The author is grateful to an anonymous referee for suggestions for improvement of an earlier version of this manuscript.

References

ACHUTHAN, N.R. and CACCETTA, L. (1991) Integer linear programming formulation for a vehicle routing problem. *European Journal of Operational Research*, **52**, 86-89.

BOHACHEVSKY, I.O., JOHNSON, M.E. and STEIN, M.L. (1986) Generalized simulated annealing for function optimization. *Technometrics*, **28**, 3, 209-217.

BODIN, L.D., GOLDEN, B.L., ASSAD, A.A. and BALL, M.O. (1983) Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, **10**, 2, 63-122.

CERNY, V. (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, **45**, 1, 41-51.

CHEN, C.S., HART, S.M. and THAM, W.M. (1995) A new simulated annealing approach to integer linear programming: design and implementation.

European Journal of Operational Research, (to appear).

CHRISTOFIDES, N., MINGOZZI, A., TOTH, P. and SANDI, C. (eds.) (1979) *Combinatorial Optimization*, New York, NY: John Wiley & Sons.

EGLESE, R.W. (1990) Simulated Annealing: a tool for operational research. *European Journal of Operational Research*, 46, 271-281.

FOSTER, B.A. and RYAN, D.M. (1976) An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly*, **27**, 2, 367-384.

GOLDEN, B., BODIN, L., DOYLE, T. and STEWART JR., W. (1980) Approximate traveling salesman algorithms. *Operations Research*, **28**, 3, 694-711.

HART, S.M. (1992) In: Transportation and material handling models for a rocket motor production system. Technical Report No. MSSU-EIRS-IE-92-1, Chapter IV, Department of Industrial Engineering, MS State University.

HART, S.M. and CHEN, C.S. (1994) Simulated annealing and the mapping problem: a computational study. *Computers & Operations Research*, **21**, 4, 455-461.

JAW, J., ODONI, A.R., PSARAFTIS, H.N. and WILSON, N.H.M. (1986) A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research*, **20B**, 3, 243-257.

JOHNSON, D.S., ARAGON, C.R., MCGEOCH, L.A. and SCHEVON, C. (1989) Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 6, 865-892.

KIKUCHI, S. and RHEE, J. (1989) Scheduling method for demand-responsive transportation system. *Journal of Transportation Engineering*, **115**, 6, 630-645.

KIRKPATRICK, S. (1984) Optimization by simulated annealing: quantitative studies. *Journal of Statistical Physics*, **34**, 5, 975-986.

KIRKPATRICK, S., GELATT JR., C.D. and VECCHI, M.P. (1983) Optimization by simulated annealing. *Science*, **220**, 4598, 671-680.

LAPORTE, G., NOBERT, Y. and DESROCHERS, M. (1985) Optimal routing under capacity and distance restrictions. *Operations Research*, **33**, 5, 1050-1073.

PSARAFTIS, H.N. (1980) A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, **14**, 2, 130-154.

PSARAFTIS, H.N. (1983) An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, **17**, 3, 351-357.

SEXTON, T.R. and BODIN, L.D. (1985A) Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transportation Science*, **19**, 4, 378-410.

SEXTON, T.R. and BODIN, L.D. (1985B) Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. *Transportation*

Science, **19**, 4, 411-435.

SCHNIEDERJANS, M.J. (1984) *Linear Goal Programming*, Princeton, NJ: Petrocelli Books.

SOLOMON, M.M. and DESROSIER, J. (1988) Time-window constrained routing and scheduling problems. *Transportation Science*, **22**, 1.

VANDERBILT, D. and LOUIE, S.G. (1984) A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, **56**, 259-271.

VAN LAARHOVEN, P.J.M. and AARTS, E.H.L. (1987) *Simulated Annealing: Theory and Applications*, Dordrecht, Netherlands: Kluwer Academic Publishers.

VECCHI, M.P. and KIRKPATRICK, S. (1983) Global wiring by simulated annealing. *IEEE Transactions on Computer-Aided Design*, **CAD-2**, 4, 215-222.