Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3851520, 11 pages
http://dx.doi.org/10.1155/2016/3851520

Research Article

Feasible Initial Population with Genetic Diversity for a Population-Based Algorithm Applied to the Vehicle Routing Problem with Time Windows

Marco Antonio Cruz-Chávez and Alina Martínez-Oropeza

Research Center in Engineering and Applied Sciences, Autonomous University of Morelos State, Avenida Universidad 1001, Colonia Chamilpa, 62209 Cuernavaca, MOR, Mexico

Correspondence should be addressed to Marco Antonio Cruz-Chávez; mcruz@uaem.mx

Received 23 August 2015; Accepted 17 January 2016

Academic Editor: Panos Liatsis

A stochastic algorithm for obtaining feasible initial populations to the Vehicle Routing Problem with Time Windows is presented. The theoretical formulation for the Vehicle Routing Problem with Time Windows is explained. The proposed method is primarily divided into a clustering algorithm and a two-phase algorithm. The first step is the application of a modified k-means clustering algorithm which is proposed in this paper. The two-phase algorithm evaluates a partial solution to transform it into a feasible individual. The two-phase algorithm consists of a hybridization of four kinds of insertions which interact randomly to obtain feasible individuals. It has been proven that different kinds of insertions impact the diversity among individuals in initial populations, which is crucial for population-based algorithm behavior. A modification to the Hamming distance method is applied to the populations generated for the Vehicle Routing Problem with Time Windows to evaluate their diversity. Experimental tests were performed based on the Solomon benchmarking. Experimental results show that the proposed method facilitates generation of highly diverse populations, which vary according to the type and distribution of the instances.

1. Introduction

The Routing Problem is one of the most important and widely studied combinatorial problems focused on distribution, logistics, and transportation systems. It is important to note that there is great variation in real problems. Each company has a different problem with specific features that makes it unique. According to its importance and hardness, several variants of the Routing Problem have been proposed to provide approaches for more realistic problems. Variants of the Routing Problem are defined by theoretical mathematical models that focus on general problems with specific features. Due to its high complexity and similarities with real problems, the Vehicle Routing Problem with Time Windows (VRPTW) is one of the most studied models.

The VRPTW can be described as a set of identical vehicles that have to serve a set of customers. Each customer has a fixed distance, a defined service time, a specific demand, and a time window during which it must be served. In addition, each vehicle has a limited capacity. A *route* is defined as the set of ordered customers served by a single vehicle, which starts and finishes at the depot. The objective function finds a solution that minimizes the total travel cost. The cost of going from customer i to j is obtained by calculating the Euclidean distance.

The VRPTW is classified by the Complexity Theory as NP-Complete [1–3]. There is no known deterministic algorithm bound in polynomial time that solves this problem. It is necessary to implement heuristic methods for undertaking problems in this class.

Although several heuristics and metaheuristics have been applied to the VRPTW in attempt to develop efficient distribution strategies according to the constraints, this work is focused on the most widely used population-based methods, evolutionary algorithms, [4] and genetic algorithms [5, 6]. Population-based algorithms sometimes suffer from premature convergence due to loss of diversity during the evolutionary process; they become rapidly trapped in local optima. The initial population diversity and the selection of genetic operators are crucial for the optimal performance of these kinds of algorithms [7, 8] because the proper selection improves the exploration and exploitation of the solution space.

Several methods exist for generating feasible solutions to the VRPTW. The most frequently used ones are clustering methods and insertion heuristics [2, 9, 10]. Both of these techniques are taken into account in this work. Some of the recent work related to this problem is presented in [11], where a basic insertion heuristic is implemented to solve the VRPTW with Multiple Routes per Vehicle in order to generate a feasible solution. At first, an unrouted customer is selected to be inserted between two customers i and j, if the hard problem constraints are fulfilled. Once a customer is inserted, the new route is updated. This process is iteratively repeated until all unrouted customers have been inserted into a route. In [12], a sequential insertion heuristic is applied to three variants of the Routing Problem. Variants used in this research are the Multiple Time Windows Vehicle Routing Problem, Heterogeneous Fleet Vehicle Routing Problem, and Double Scheduling Vehicle Routing Problem. The efficiency of the insertion heuristic is evaluated based on Solomon's and Gehring and Homberger's benchmarks. This method is very useful for heuristics that require a high-quality initial solution, such as Taboo Search.

According to the literature, a crucial factor in obtaining high-quality solutions in a population-based algorithm is the diversity. There are several methods for obtaining equilibrium between exploration and exploitation of the solution space to get the proper convergence and avoid stagnation behavior. It is well-known that the initial population plays an important role in the convergence of population-based algorithms. High-quality solutions will be obtained with the combination of a sufficiently diverse population and the application of the proper genetic operators that allow diversity to be maintained [13–15]. In addition, several methods exist that inject diversity into the population. One of the most relevant of these methods is the DDCGA (Dynamic Diversity Control Genetic Algorithms) proposed by Chang et al. [16]. This method is a chromosome control mechanism that generates a set of artificial chromosomes with high diversity and injects them into the population to effectively maintain the diversity during the evolutionary process. Anselmo and Pinheiro [17] presented an interesting work which is not applied to Routing Problems but is focused on the construction of phylogenetic trees based on the evaluation of Hamming distance among DNA sequences. They proposed an Unweighted Pair Group Method with Arithmetic Mean (UPGMA) and neighbor joining (NJ). This method is applied to two binary data sets based on sequences of mitochondrial DNA from humans, chimpanzees, gorillas, and orangutans. The statistical properties of the DNA sequences are compared using Hamming distance to construct the phylogenetic tree based on their differences. In addition, this method has the capacity to efficiently deal with multiple and different numbers of sequences per group. The relevance of this work to the current paper is that the UPGMA examines several binary sequences. These DNA sequences are studied to identify which of them are related among themselves, in order to construct a phylogenetic tree. The idea of using Hamming distance to identify relationships among compound data sets with several subgroups is taken in this paper. The evaluation of dissimilarities allows for identification of the need to increase the diversity in the nonbinary initial population. The similarities help identify identical individuals, which could negatively affect the algorithm's behavior. More recently, Yang and Li [18] presented another relevant method called the Autoenhanced Population Diversity (AEPD). This method identifies when the population should be rediversified based on its convergence or stagnation and reinitializes individuals to enhance population diversity.

Based on the previously explained work, some modifications of the classical k-means algorithm are implemented in this research. Those features are described in Section 3. The contribution of this paper is focused on the generation of feasible solutions from the partially feasible solutions obtained by the clustering method, where no time constraints have been evaluated yet. Therefore, a two-phase algorithm is proposed to evaluate the time windows and generate highly diverse initial populations. The first phase corresponds to an initial evaluation of time constraints, discarding those customers with time violations. The second phase implements a hybrid insertion heuristic to ensure the solutions feasibility. According to experimental results, the hybridization performed in this phase favors the diversity among solutions.

According to literature, one of the crucial features of a population is the diversity, because the convergence of a population-based algorithm depends on it. There are various methods for calculating diversity. The most widely used are Levenshtein distance [19] and Hamming distance [20].

The Hamming distance is the most common diversity measure for comparisons of categorical sequences and binary sequences [21]. Although there are several ways to apply this method to optimization problems, in this work a modification was made. The Hamming distance was applied, but it was necessary to adapt the method to the solution structure that the VRPTW represents. A computational method is proposed based on this methodology.

The present work is organized as follows. Section 2 provides a description of the Vehicle Routing Problem with Time Windows. Section 3 describes the clustering problem and the implemented clustering algorithm. Section 4 explains the two-phase algorithm proposed in this work. Section 5 introduces the classical Hamming distance method. It also presents the modifications, methodology, and algorithm proposed for calculating the diversity among solutions in a population for the undertaken problem. Section 6 shows the obtained experimental results based on the well-known Solomon benchmarking of 100 customers. Finally, the conclusions and future research direction are presented.

2. Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) is one of the most studied problems due to its wide application in industry and its economic significance. It is important because it involves the main features of the classical problems of transportation, distribution, and logistics. Therefore, the focus of this paper is on the model proposed by Paolo and Daniele [2]. The VRPTW is considered to be a hard problem and is classified by Complexity Theory as NP-Complete [1–3].

The VRPTW can be formally described as a directed graph $G = (V, E)$ with a set $V = \{0, 1, 2, \ldots, n\}$ and an edge set E. The set V consists of $1, 2, \ldots, n$ customers distributed into a two-dimensional space (x_i, y_i), where each customer i has a defined service time S_i, a specific demand d_i, and a time window $[a_i, b_i]$ in which the customer must be served. Each customer must be served at most by a single vehicle. Each vehicle belongs to a homogeneous fleet of K identical vehicles with limited capacity C. The ordered sequence of customers visited by a vehicle k is known as a *route*; a single route is exclusive to a single vehicle. Each route must start and finish at the depot (identified as 0 at the beginning and $n + 1$ at the end) within the time window of the depot $[E, L]$.

The objective is to minimize the number of routes and the total travel cost c_{ij} of the solution. The cost of a route is given by the sum of the Euclidean distance of each edge between customers (i, j) served by a single vehicle, where $i \neq j$. Therefore, the total travel cost corresponds to the sum of the route's costs involved in a solution. According to this description, the integer linear programming model of the VRPTW [2] can be formulated as shown below.

Integer Linear Programming Model of the VRPTW [2]. Consider

$$\min \quad f = \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ijk} \tag{1}$$

$$\text{Subject to:} \quad \sum_{k\in K} \sum_{j\in\Delta^+(i)} x_{ijk} = 1 \quad \forall i \in N, \tag{2}$$

$$\sum_{j\in\Delta^+(0)} x_{0jk} = 1 \quad \forall k \in K, \tag{3}$$

$$\sum_{i\in\Delta^-(j)} x_{ijk} - \sum_{i\in\Delta^+(j)} x_{ijk} = 0 \qquad \forall k \in K,\ j \in N, \tag{4}$$

$$\sum_{i\in\Delta^-(n+1)} x_{i,n+1,k} = 1 \quad \forall k \in K, \tag{5}$$

$$w_{ik} + s_i + t_{ij} - w_{jk} \leq \left(1 - x_{ijk}\right) M_{ij} \qquad \forall k \in K,\ (i, j) \in A, \tag{6}$$

$$a_i \sum_{j\in\Delta^+(i)} x_{ijk} \leq w_{ik} < b_i \sum_{j\in\Delta^+(i)} x_{ijk} \qquad \forall k \in N,\ i \in N, \tag{7}$$

$$E \leq w_{ik} < L \quad \forall k \in K,\ i \in \{0, n+1\}, \tag{8}$$

$$\sum_{i\in N} d_i \sum_{j\in\Delta^+(i)} x_{ijk} \leq C \quad \forall k \in K, \tag{9}$$

$$x_{ijk} \geq 0 \quad \forall k \in K,\ (i, j) \in A, \tag{10}$$

$$x_{ijk} \in \{0, 1\} \quad \forall k \in K,\ (i, j) \in A. \tag{11}$$

The goal of the objective function (1) is to minimize the total cost of the solution by attempting to reduce the number of routes. The constraint set in (2) makes certain that each customer is visited by a single vehicle at most. The constraint set in (3) ensures that only one customer connects directly to the depot at the beginning of a route. The inequality set in (4) safeguards the idea that the number of vehicles arriving to a customer is the same as that leaving it. The constraint set in (5) verifies that only one customer connects to the depot at the end of a route. The constraint set in (6) guarantees the feasibility of a solution. The constraint set in (7) ensures that the time constraints of each customer are met. The constraint set in (8) guarantees that all of the routes are performed within the time window of the depot. The constraint set in (9) ensures that the sum of customer demands assigned to a specific vehicle does not exceed its maximum capacity. The constraint sets in (10) and (11) guarantee the nonnegativity of variables x and define the formulation as a binary integer linear programming model.

3. Clustering Algorithm

Clustering is an important and difficult problem classified as NP-Hard [12], with many applications in different areas. In Combinatorial Optimization, it is applied to difficult problems based on the paradigm "*Divide and Conquer*," where the problem is divided into P segments or subsets [22]. Each subset groups together similar data, separating it by feature into different subsets so that a large problem can be approached as smaller independent subproblems. Formally, a clustering problem can be described as a problem divided into $P = s_1, s_2, \ldots, s_P$ subsets. Therefore, a solution is a compound of P data subsets which can be undertaken independently.

Several clustering methods exist [23] that can be applied to Routing Problems, but one of the most well-known and widely used is the *k-means* [24]. This method consists of the evaluation of a set of N data distributed in a Euclidean space, where P disjoint subsets of elements have to be created according to their data features, where P is a known value at the beginning of the process. The first step is to determine the P starting points called *centroids*. A *centroid* is determined as a point in space. Each centroid groups together a set of elements that minimize the mean squared Euclidean distances from each element to the nearest centroid. Despite its popularity, this method has advantages and disadvantages [8], which are listed below.

Advantages are as follows:

(i) It is simple and fast.

(ii) It is relatively flexible and efficient.

(iii) It can process values with good geometrical and statistical meaning.

(iv) It has relatively good results for convex clusters.

Disadvantages are as follows:

(i) It is necessary to know the number of clusters at the beginning.

(ii) There is sensitivity to the starting point (selection of the first centroid).

(iii) There are clusters with similar form and density.

(iv) There is recalculation of centroids and distances.

According to the features of k-means [23] and the known desirable features of clustering methods [25], some modifications were applied according to the features of the clustering method and the proposed mathematical model shown in (1)–(11). The algorithm is shown in Algorithm 1.

The proposed algorithm (Algorithm 1) divides the elements of input into subsets. This algorithm is applied specifically to the VRPTW. Elements are known as customers and the Euclidean distance between two customers i and j is denoted as distance or cost. Each obtained subset corresponds to a route. The modifications of the k-means algorithm are such that each customer has a specific demand which is evaluated before the insertion of a customer into a cluster. In order to generate a semifeasible solution where capacity constraints are taken into account, the segmentation involves the evaluation of the distance between the customers (i, j) and the customers' demand. It is noteworthy that it is not necessary to know the number of clusters at the beginning of the process, because the algorithm finds the correct number of clusters according to the input and constraints.

The benchmarks of Solomon are the algorithm input in this research. The algorithm receives the input in Step (2) (Algorithm 1). Step (3) takes the benchmarking information and focuses on the coordinates of customers in order to calculate the Euclidean distances between them. Step (4) performs the initialization of the taboo list, which is useful to control the customer assignment, using 0 for nonassigned customers and 1 for assigned ones. Step (5) is the iterative process of clustering. The first step is to create a new cluster by the random selection of a centroid. This centroid corresponds to a nonassigned customer. After this selection, the centroid status is changed in the taboo list and nonassigned customers are evaluated according to distance and demand, with an attempt to add as many customers as possible to the current cluster. If the maximum capacity of a vehicle is exceeded, another route is opened, without exceeding the maximum number of vehicles in the fleet. Step (5) is repeated until all the customers are assigned to a route. At the end of this step, it is said that a semifeasible solution is obtained, because capacity constraints are met on each route.

The modifications applied to k-means are as follows:

(i) Number of clusters does not have to be known.

(ii) A centroid is a customer, instead of a point in space.

(iii) Euclidean distances are calculated only once.

```
(1) Initialize $(N, Vh, C)$
(2) Read Input Data
(3) for $i = 0 : i < N$
(4)   for $j = 0 : j < N$   $d[i][j] = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
(5)   end-for
(6) end-for
(7) Initialize structure Solution, vect_taboo, index
(8) Repeat
(9)      centroid = 1 + (rand()%$N$)
(10)     if vect_taboo [centroid] == 1 then
(11)          Repeat centroid = 1 + (rand()%$N$)
(12)          Until vect_taboo [centroid] == 0
(13)     end-if
(14)     vect_taboo [centroid] == 1
(15)     Search_nearest_client_at_centroid
(16)     if $(d_i + \sum_{i \in N} d_k) \leq C$ then
(17)          if vect_taboo [$i$] == 1 then
(18)                 Repeat Search_nearest_client_at_centroid
(19)                 Until vect_taboo [$i$] == 0
(20)                 if $(d_i + \sum_{i \in N} d_k) > C$ then $v$++ end-if
(21)          end-if
(22)     else
(23)          vect_taboo [$i$] = 1
(24)          Solution [index] = $i$
(25)     end-if
(26) Until index == $N - 1$
```

Algorithm 1: Clustering algorithm applied to VRPTW.

(iv) It is not necessary to recalculate centroids.

(v) Customers demand and maximum vehicle capacity are taken into account.

The solution obtained by the clustering algorithm is taken as the input for the two-phase algorithm to turn a semifeasible solution into a feasible one.

4. Two-Phase Algorithm

The input for this algorithm is a semifeasible solution, where capacity constraints of the VRPTW are met. However, some constraints have not been evaluated yet. The proposed two-phase algorithm was developed to focus on the evaluation of time constraints with the aim of turning each semifeasible solution into a feasible one.

The two-phase algorithm is implemented in two distinct phases which are outlined here. The first phase performs a first debugging, leaving each cluster with only the customers whose time windows are met and marking those with violations of time constraints as nonassigned on the taboo list. The second phase corresponds to an insertion heuristic, which takes the nonassigned customers and tries to insert them into the existent routes based on the problem constraints. If any customer cannot be inserted, another route is created. The whole process of the proposed methodology is described in the following subsections.

4.1. First Phase. This phase receives the semifeasible solution generated by the clustering algorithm described in Section 3. The algorithm then performs an evaluation of time constraints.

The time constraints are known for being the hardest constraints in the VRPTW, because each customer has a different and well-defined time window during which it must be served by a single vehicle. If the vehicle arrives before the time window opens, it has to wait, and no delays are permitted. If a vehicle arrives after the time window closes, the solution is considered unfeasible.

Based on this concept, the received solution is evaluated, leaving the customers with time violations out of the route and changing their status on the taboo list as nonassigned. In this research, this new solution is called partially feasible because all the constraints are met by the assigned customers, but there are still nonassigned customers.

4.2. Second Phase. This phase receives the partially feasible solution generated by the first phase explained in Section 4.1. An insertion heuristic is applied in order to convert the partially feasible solution into a feasible one. The insertion heuristic proposed in this research is a hybridization of four randomly applied straightforward insertion methods. The methods involved in this hybridization are explained as follows.

(i) *Nearest Route (NR).* This method randomly selects a nonassigned customer, then looks for the nearest route, and evaluates capacity and time constraints. If constraints are met, the customer is assigned. Otherwise, another route is selected under the same evaluation criteria. If the customer cannot be assigned to an existent route, another one is created.

(ii) *Direct/Swap Insertion (D/S).* After selection of a nonassigned customer, this method tries to insert the customer directly into the nearest route according to problem constraints. If the constraints are not met, the customer is inserted and another route is randomly selected. Immediately, an exchange of one customer belonging to each of the two routes is performed. If capacity and time constraints are met in both routes, the operation is considered successful and the customer is marked as inserted. Otherwise, both routes return to their original sequences. Another route is then selected and the same procedure is repeated. It is repeated t iterations or until the customer has been inserted. In case it cannot be inserted into any route, another one is created.

(iii) *Swap Insertion (S).* A nonassigned customer and a route are randomly selected. The customer is inserted into the route independent of whether the capacity and time constraints are met or not. Consequently, another route is randomly selected. After that, one customer from each of the selected routes is randomly selected to perform an exchange. In this step, capacity and time constraints are evaluated in both routes. If constraints are met, routes are permanently modified and the customer is consider to be inserted. Otherwise, the process is repeated t times or until the customer is inserted. If the customer cannot be inserted into any existing routes, another route is created.

(iv) *Swaps 1 by 1 and 2 by 2 (1-1 and 2-2).* In this method, the procedure attempts to force customer insertion using two different exchange movements. First, a nonassigned customer is selected, and the process explained in the *swap insertion* is applied. If the customer was not inserted during this process after t iterations, instead of creating a new route, the method performs exchanges of two customers per route under the same conditions as the swap insertion. At the end of this process, if the customer was not inserted, a new route is created.

These random procedures are iteratively performed until all the customers have been scheduled. Once all the customers have been inserted, it is said that a feasible solution to the VRPTW has been obtained.

This research is focused on a population-based algorithm so it is necessary to create a population of m feasible individuals (solutions). Thus, the whole process of creating a feasible solution has to be repeated m times.

5. Hamming Distance

The term *Hamming distance* is based on the *Hamming codes* proposed by Hamming Richard in 1950 [26]. The Hamming distance is a diversity measure used to understand how different two sequences are. The Hamming distance D_H between two binary sequences, $\{X_i\}$ and $\{X_j\}$, with the same length, $i, j = 1, \ldots, N$, is represented in (12). N corresponds to the quantity of elements in each sequence and D_H refers to the difference between the two sequences. Therefore, a small value of D_H indicates that the sequences are very similar. As the value of D_H approaches N, the difference between the sequences increases, indicating that more movements are required to make a transformation from one sequence to the other [27]:

$$D_H = \sum_{n=1}^{N} \left(X_{in} \neq X_{jn} \right). \tag{12}$$

An example of a Hamming distance calculation for sequences of $N = 3$ is shown in Figure 1.

Analogously, Hamming distance can be seen as the minimal number of edges connecting two vertices in an N-*dimensional* representation [28], where N is the number of elements in each sequence. With this type of representation, the example in Figure 1 is seen as follows (Figure 2).

With this representation, the Hamming distance allows the concept of the *nearest neighbor* to be defined. As fewer changes take place to go from one sequence to another, sequences become more similar.

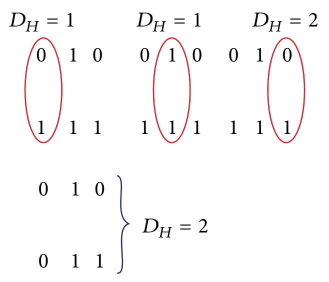


FIGURE 1: Example of the calculation of Hamming distance applied to binary sequences of three elements.

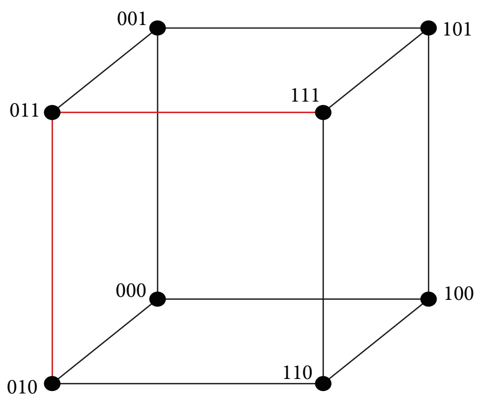


FIGURE 2: Tridimensional representation of Hamming distance between two sequences of three binary elements [28].

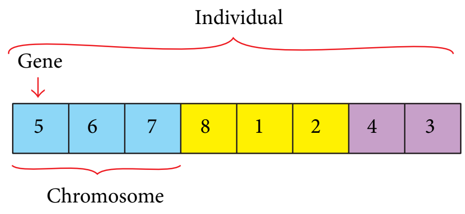


FIGURE 3: Representation of an individual of the VRPTW.

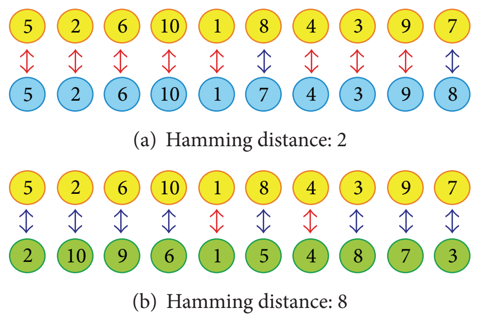


FIGURE 4: (a) Hamming distance calculation for two similar sequences, (b) Hamming distance between very different sequences.

6. Methodology for Hamming Distance Calculation for the VRPTW

The procedure explained in Section 5 was modified in this research, according to the solution structure for the VRPTW of the model shown in (1)–(11). Taking into account that a solution is an individual within the population, the structure of an individual is presented in Figure 3. A *chromosome* corresponds to a route in a solution and a *gene* is a customer assigned to a specific route.

In the literature, the Hamming distance method has been applied to simple sequences which do not have subsequences, as shown in Figures 4(a) and 4(b).

Under these conditions, the Hamming distance is a very simple method, but its complexity increases when it is applied to sequences (individuals) with structural differences, such as those presented in Figure 3. Each individual is composed of several subsequences (chromosomes); each of these is considered a compound route of a set of ordered customers. Based on the features of individuals, a methodology for applying the Hamming distance to VRPTW was developed. The proposed methodology is represented in Figure 5.

This methodology is developed based on chromosomes (routes). The order of chromosomes within an individual is not relevant since two individuals can have the same chromosomes in different orders and still be identical. Therefore, comparisons are made according to positions of chromosomes. For instance, in Figure 5, the customer in position *1,* belonging to the route 1 of *ind 1,* is compared with the customer in position *1* of every route in *ind 2.* The comparisons continue until all the customers in positions *1* have been compared or the same customer is found at the same position in another route, such as the example shown in Figure 5(a). Then, using the same procedure, the next position is evaluated. This process is repeated until all the positions of each route of *ind 1* have been compared with all the routes of *ind 2* (Figures 5(b), 5(c), 5(d), and 5(e)). The value of the Hamming distance remains the same when a customer is found in the same position in any route in the second individual. Otherwise, it increases in value. Thus, a high Hamming distance value means that the individuals being compared are very different. On the contrary, a small Hamming distance value means that individuals are very similar.

6.1. Algorithm for Hamming Distance Calculation. In the literature there are several computational methods for calculating the Hamming distance, but in this work an algorithm based on the methodology described in Section 6 was developed. This algorithm computes the diversity in a population for the proposed algorithm for the model presented in (1)–(11). The algorithm was developed on Visual C 2008. The pseudocode of the proposed algorithm is shown in Algorithm 2.

The algorithm (Algorithm 2) shows an exhaustive comparison process, where each individual is compared with all of the individuals in the population. This process is useful to identify individuals with identical genotype, which is crucial because having identical individuals in a population reduces the diversity. It has been proven that population diversity directly affects the convergence of the algorithm.

According to the algorithm shown in Algorithm 2, the nested loops in Steps (3) and (4) perform a comparison

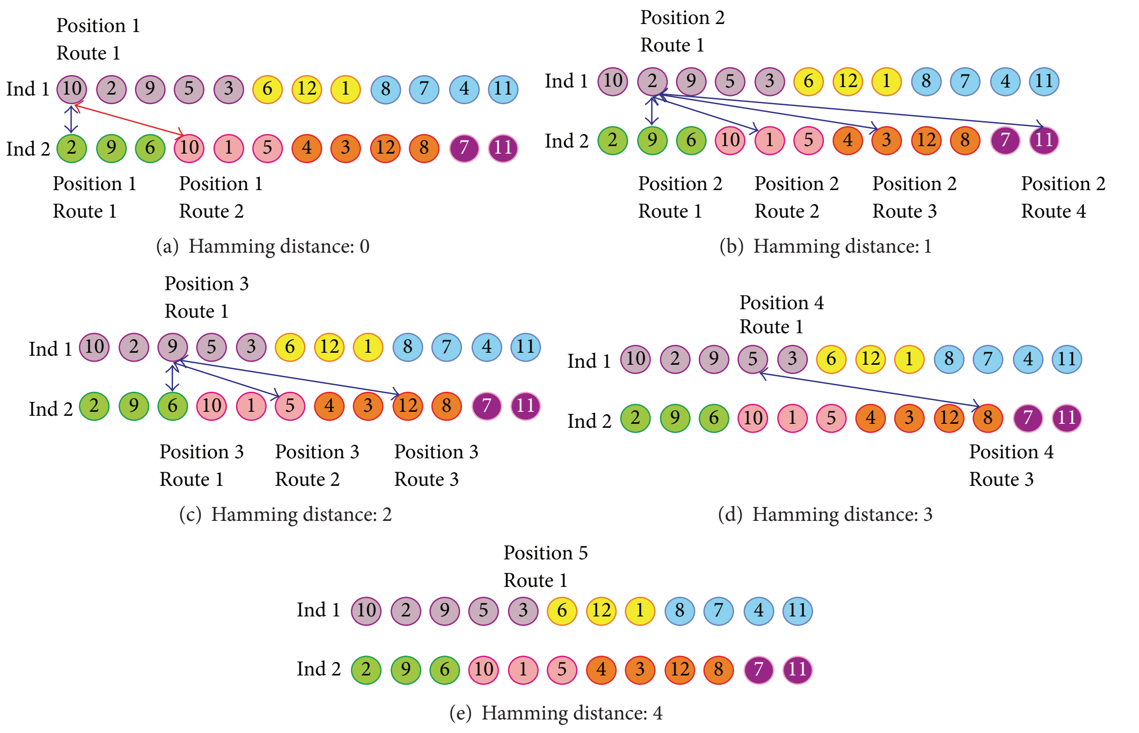


FIGURE 5: Proposed methodology for calculation of Hamming distance between two feasible individuals for the VRPTW.

```
(1) **Initialize** ind, aux, Hamming, cont, $D_H$
(2) Read_Population (InitialPopulation)
(3) for ind = 0 : ind < InitialPopulation
(4)     for aux = ind + 1 : aux < InitialPopulation
(5)         Hamming = cont = 0
(6)         for vh = 0 : vh < ChromosomesInd1
(7)             flag = 0
(8)             for vhInd2 = 0 : vhInd2 < ChromosomesInd2
(9)                 if $I$ == IniChromInd1    cont = 0
(10)                end-if
(11)                if $j \leq$ FinChromInd2 && $j \leq$ FinChromInd1
(12)                    if gens of ind and aux are identical
(13)                        flag = 1
(14)                        vhInd2 = ChromosomesInd2
(15)                    end-if
(16)                end-if
(17)            end-for
(18)            if flag == 0
(19)                Hamming++
(20)            end-if
(21)            cont++
(22)        end-for
(23)    end-for
(24)    $D_H$ = Hamming
(25) end-for
```

ALGORITHM 2: Calculation of the Hamming distance for a population of N individuals.

among all the individuals in the population. The loop in Step (6) takes each of the routes (chromosomes) of each individual. Step (8) takes each of the customers in order which have been assigned to a specific route. Together, Steps (6) and (8) compare them to those contained in another individual. Conditionals in Steps (9) to (16) evaluate whether the customer is in the same position in another individual. If this customer does not exist in the same position, Steps (18) to (21) increase the value of the Hamming distance; otherwise, it remains the same. The temporal function of the algorithm is shown in (13) and has a complexity of $O(n^2m^2s)$:

$$T_{\text{sec}}(n) = 7n^2m^2s + 5n^2ms + n^2m + 82n^2 + 14m^2s + 4nk + 5nm + 10ms + 13n + 2m + 248, \tag{13}$$

where n is size of population; m is number of routes (chromosomes) in an individual; s is number of customers per route, which used to be variable; k is total number of customers to be assigned.

This complexity does not affect the efficiency of the algorithm because it is applied only once, when the feasible initial population is created. Moreover, this code is applied as an external method and is not included in the population-based algorithm.

7. Experimental Results

Experimental tests were performed using the 56 Solomon's benchmarks of 100 customers. The Solomon's benchmarks are divided into two types (according to the hardness of time windows and vehicles capacity) and three kinds of distribution. Type 1 refers to benchmarks with narrow time windows and vehicles with small capacities. Type 2 incorporates larger time windows and larger vehicle capacities. Distribution is divided into three categories: Clustered (C), Random (R), and Clustered-Random (RC). Clustered (C) distribution indicates that customers are geographically grouped. Random (R) distribution uses a random location for all the customers. Finally, the Clustered-Random (RC) distribution is a mix of the two distributions previously explained.

The goal of this research is to develop an algorithm that allows highly diverse populations to be obtained. These are crucial for optimal performance of population-based algorithms because they favor the process of exploration. However, a diverse population is not the only important factor for improving performance. The selection of the genetic operators plays an important role in this process also. With high diversity in the initial population and the application of the proper genetic operators, there is an elevated probability of achieving high performance in population-based algorithms.

After applying the modified clustering method, the two-phase algorithm is implemented to make the solution feasible using one of the following different insertion heuristics: nearest route (NR), direct/swap insertion (D/S), swap insertion (S), insertions 1 by 1 and 2 by 2 (1-1 and 2-2), and the proposed hybrid insertion algorithm. These five insertion heuristics were experimentally tested to determine which showed best performance for the undertaken problem.

Testing was performed on a laptop with a Quad-core 1.73 GHz processor and RAM of 6 GB using Windows 7 OS. All the algorithms in this research were developed using the compiler Visual C 2008. It is noteworthy that the quality of individuals is determined based on the diversity within a single population, because an optimization method has not yet been applied. The averages of experimental results are shown in Table 1 for each of the evaluated benchmarks. Thirty executions per benchmark were performed.

Experimental results shown in Table 1 correspond to the efficacy of the solutions based on the Hamming distance. It is important to mention that efficacy is calculated in terms of diversity, because an optimization method has not yet been applied. The Hamming distance is one of the most frequently used diversity measures because it calculates quantitatively how different two sequences are. The distance was calculated according to the average of 30 executions per benchmark, where each execution generates a population of 500 individuals to show the scalability of the algorithm.

Experimental testing was performed by evaluating each of the five chosen insertion heuristics separately. According to the results shown in Table 1, the insertion heuristic that obtains the largest diversity is the proposed hybrid insertion heuristic. This proposal consists of random selection of the four straightforward insertion heuristics explained in this research, whose movements promote increased diversity. Variations in movement allow the neighborhood size to be modified upon each application of the proposed structure. Moreover, these results can be seen graphically for each benchmark in Figures 6(a)–6(c).

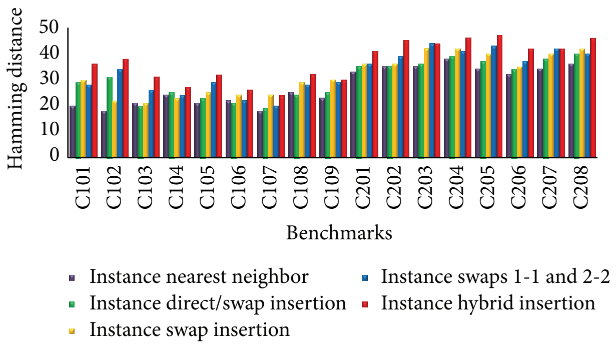
According to the obtained results, which are shown in Figure 6, the proposed hybrid insertion heuristic obtained more diversity than the single straightforward insertion heuristics for nearly all the benchmarks. Based on the type of distribution, *Clustered* (C) *benchmarks* were shown to have more difficulty obtaining high diversity than other distributions because the customers' location plays an important role. This is contrary to *Random* (R) and *Random-Clustered* (RC) *benchmarks*, which are more flexible and therefore able to achieve more diversity due to their distribution features. By analyzing the obtained value for the Hamming distance, it is possible to conclude that the critical parameters for diversity are the distribution of customers and the time windows, which are the hardest constraints.

The population diversity in populations is limited by the sets of hard constraints (6), (7), (8), and (9). The attempt to generate greater diversity implies obtaining unfeasible solutions. The diversity is obtained within a part of the solution space limited by the problem constraints. If there were no sets of constraints, the diversity could be greater, but the model would correspond to a relaxation that does not include time window, capacity, and customer constraints.

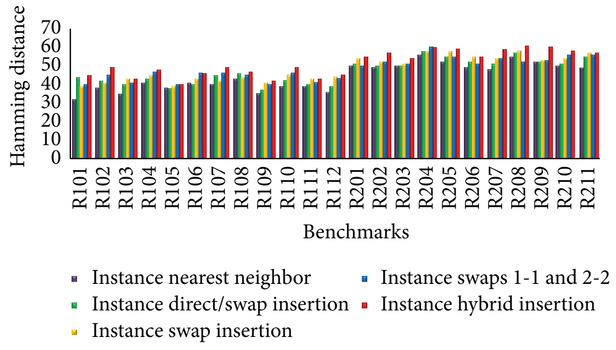
In terms of efficiency, tests were performed under the same conditions as the efficacy tests, using 30 executions per benchmark and obtaining populations of 500 individuals. The obtained results were grouped together to calculate the average by type and distribution. The time average of the obtained results for each of the type/distribution benchmarks is shown in Figure 7.

TABLE 1: Average of 30 executions per benchmark for calculation of Hamming distance for the Solomon's benchmarks.

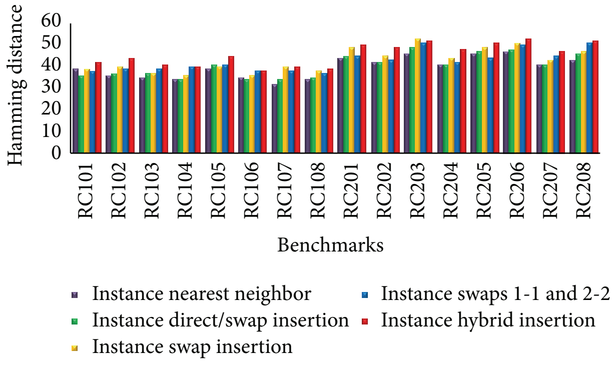
Instance	NN	D/S	S	1-1 and 2-2	Hybrid
Type 1 (Hamming distance)					
C101	20	29	30	28	**36**
C102	18	31	22	34	**38**
C103	21	20	21	26	**31**
C104	24	25	23	24	**27**
C105	21	23	25	29	**32**
C106	22	21	24	22	**26**
C107	18	19	**24**	20	**24**
C108	25	24	29	28	**32**
C109	23	25	**30**	29	**30**
R101	32	44	39	40	**45**
R102	38	42	41	45	**49**
R103	35	40	**43**	41	**43**
R104	41	43	45	47	**48**
R105	38	38	39	**40**	**40**
R106	41	40	43	**46**	**46**
R107	40	45	42	46	**49**
R108	43	46	44	45	**47**
R109	35	37	41	40	**42**
R110	39	42	45	46	**49**
R111	39	40	**43**	41	**43**
R112	36	39	44	43	**45**
RC101	39	36	39	38	**42**
RC102	36	37	40	39	**44**
RC103	35	37	37	39	**41**
RC104	34	34	36	**40**	**40**
RC105	39	41	40	41	**45**
RC106	35	34	36	**38**	**38**
RC107	32	34	**40**	38	**40**
RC108	34	35	38	37	**39**
Type 2 (Hamming distance)					
C201	33	35	36	36	**41**
C202	35	35	36	39	**45**
C203	35	36	42	**44**	**44**
C204	38	39	42	41	**46**
C205	34	37	40	43	**47**
C206	32	34	35	37	**42**
C207	34	38	40	**42**	**42**
C208	36	40	42	40	**46**
R201	50	51	54	50	**55**
R202	49	50	52	52	**57**
R203	50	50	51	51	**54**
R204	56	58	58	**60**	**60**
R205	52	55	58	55	**59**
R206	49	52	**55**	51	**55**
R207	48	51	54	54	**59**
R208	55	57	58	52	**61**
R209	52	52	53	53	**60**
R210	50	51	54	56	**58**
R211	49	55	**57**	56	**57**
RC201	44	45	49	45	**50**
RC202	42	42	45	43	**49**
RC203	46	49	**53**	51	52
RC204	41	41	44	42	**48**
RC205	46	47	49	44	**51**
RC206	47	48	51	50	**53**
RC207	41	41	43	45	**47**
RC208	43	46	47	51	**52**



(a)



(b)



(c)

FIGURE 6: Average value of Hamming distance reached by the proposed algorithm for each of the benchmarks applied to (a) Clustered distribution, (b) Random distribution, and (c) Clustered-Random distribution.

The results in Figure 7 show that the proposed hybrid insertion heuristic, represented with a light blue bar, is competitive in efficiency compared to the other straightforward structures. This behavior is explained because it involves four different insertions, which are selected randomly. Thus, the execution time varies according to the number of times each movement is applied.

The application of the proposed hybrid insertion heuristic within the two-phase algorithm was decided according to the analysis of efficacy and efficiency results, shown in Figures 6 and 7.

Once the genetic algorithm is applied to the problem, the genetic operators play an important role in the process of

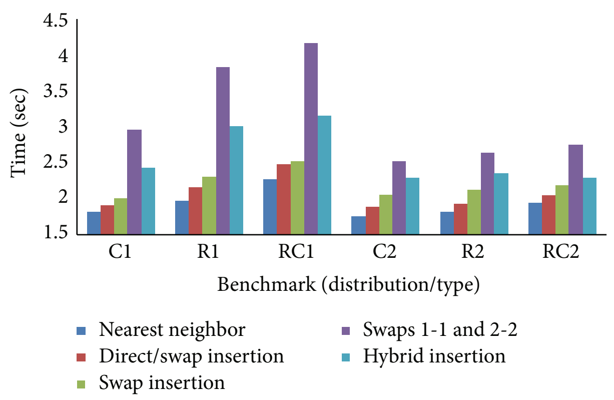


Figure 7: Average execution time to obtain a feasible population of 500 customers.

```
(1) **Begin_generation**
(2) Selection
(3) Crossover
(4) Modified Mutation operator
(5) Population update
(6) **End_generation**
```

Algorithm 3: Reinsertion of diversity in a genetic algorithm.

exploration and exploitation of the solution space. Genetic operators must be selected carefully according to the features of the problem in order to maintain the proper diversity and obtain the best possible solution. Another way to maintain diversity during the optimization process is to adapt the second phase of the proposed two-phase algorithm. A mutation operator can be applied at given intervals, after a certain number of generations, to reinsert diversity within the population and avoid premature convergence, as shown in Algorithm 3.

8. Conclusion

In this paper, a modified k-means algorithm was applied to the VRPTW, where only the capacity constraints were taken into account. The experimental tests were performed to prove the efficacy of the clustering algorithm and the proposed two-phase algorithm to obtain feasible solutions to VRPTW.

The two-phase algorithm performs two procedures to generate a feasible solution based on the evaluation of time constraints. The first procedure evaluates the time windows of the customers in an assigned route, leaving out those that violate time constraints. The second procedure applies an insertion heuristic to add the nonassigned customers to a route. The selection of the insertion heuristic is done randomly. It is noteworthy that efficacy was evaluated according to the Hamming distance of the obtained population. Five insertion heuristics were evaluated and the best results were obtained with the proposed hybrid insertion heuristic. This proposed insertion heuristic obtains populations with more diversity and is competitive in efficiency when compared to the other insertion heuristics evaluated.

In this paper, a methodology to calculate the Hamming distance for solutions to the model of VRPTW was described and is shown in (1)–(11). Moreover, a computational method was developed based on this methodology, enabling the calculation of Hamming distance for populations of feasible individuals. This is important, because, in a population, diversity is a critical feature for maintaining maximum performance of the algorithm. Therefore, this characteristic, in conjunction with genetic operators, is very important for avoiding stagnation behavior and becoming trapped in local optima.

Moreover, based on the results of the Hamming distance for benchmarks with different features, it is possible to identify the critical parameters for diversity, which are the hardness of time windows. Under these conditions, experimental results show that it is possible to obtain major diversity for type 2 benchmarks. In the case of distribution, benchmarks with a Random distribution allow higher values of Hamming distance to be obtained. The proposed methodology enables highly diverse populations to be attained based on the application of the clustering method followed by random application of insertion heuristics used in the two-phase algorithm. The neighborhood size is continuously changing which improves the distribution in the solution space.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[2] T. Paolo and V. Daniele, *The Vehicle Routing Problem* , Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2002.

[3] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, Mineola, NY, USA, 1998.

[4] A. Garcia-Najera and J. A. Bullinaria, “An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows,” *Computers & Operations Research*, vol. 38, no. 1, pp. 287–300, 2010.

[5] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, “A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows,” *Computers & Operations Research*, vol. 40, no. 1, pp. 475–489, 2013.

[6] Y. Deng, J. Xiang, and Z. Ou, “Improvement of genetic algorithm for vehicle routing problems with time windows,” in *Proceedings of the 3rd International Conference on Intelligent System Design and Engineering Applications (ISDEA ’13)*, pp. 866–869, IEEE, Hong Kong, January 2013.

[7] M. Lozano, F. Herrera, and J. R. Cano, “Replacement strategies to preserve useful diversity in steady-state genetic algorithms,” *Information Sciences*, vol. 178, no. 23, pp. 4421–4433, 2008.

[8] G. Deepti and G. Shabina, "An overview of methods maintaining diversity in genetic algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 5, 2012.

[9] S. Gavriel, *Handbook of Industrial Engineering. Technology and Operations Management*, Institute of Industrial Engineers, Norcross, GA, USA, 3rd edition, 2001.

[10] V. Pureza, R. Morabito, and M. Reimann, "Vehicle routing with multiple deliverymen: modeling and heuristic approaches for the VRPTW," *European Journal of Operational Research*, vol. 218, no. 3, pp. 636–647, 2012.

[11] A. M. Campbell and M. Savelsbergh, "Efficient insertion heuristics for vehicle routing and scheduling problems," *Transportation Science*, vol. 38, no. 3, pp. 369–378, 2004.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.

[13] P. A. Diaz-Gómez and D. F. Hougen, "Empirical study: initial population diversity and genetic algorithm performance," in *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition (AIPR '07)*, pp. 334–341, July 2007.

[14] P. A. Diaz-Gómez and D. F. Hougen, "Initial population for genetic algorithms: a metric approach," in *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM '07)*, Las Vegas, Nev, USA, June 2007.

[15] P. Annibale, O. Rocco, D. Massimiliano, and D. Andrea, "Improving multi-objective test case selection by injecting diversity in genetic algorithms," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 358–383, 2014.

[16] P.-C. Chang, W.-H. Huang, and C.-J. Ting, "Dynamic diversity control in genetic algorithm for mining unsearched solution space in TSP problems," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1863–1878, 2010.

[17] C. A. F. Anselmo and A. Pinheiro, "Phylogenetic trees via Hamming distance decomposition tests," *Journal of Statistical Computation and Simulation*, vol. 82, no. 9, pp. 1287–1297, 2012.

[18] M. Yang and C. Li, "Differencial evolution with auto-enhanced population diversity," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 302–315, 2014.

[19] E. K. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: an analysis of measures and correlation with fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, 2004.

[20] S. J. Louis and G. J. E. Rawlins, *Predicting Convergence Time for Genetic Algorithms*, Foundations of Genetic Algorithms, 1992.

[21] H. P. Pinheiro, A. de Souza Pinheiro, and P. K. Sen, "Comparison of genomic sequences using the Hamming distance," *Journal of Statistical Planning and Inference*, vol. 130, no. 1-2, pp. 325–339, 2005.

[22] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[23] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[24] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," in *Proceedings of the 18th Annual Symposium on Computational Geometry (SCG '02)*, pp. 10–18, ACM, Barcelona, Spain, June 2002.

[25] H.-V. Edna, *Clustering algorithm based on entropy to discover sets of attributes within mix-type attributes [Master Thesis in Sciences with Option in Computing]*, Department of Electrical Engineering Computing Section, Centro de Investigación y de Estudios Avanzados del IPN, 2006.

[26] W. Hamming Richard, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 26, no. 2, pp. 147–160, 1950.

[27] S. Ignacimuthu, *Basic Bioinformatics*, Alpha Science International, New Delhi, India, 2005.

[28] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867–872, 1988.