USING SIMULATED ANNEALING TO MINIMIZE THE COST OF CENTRALIZED TELECOMMUNICATIONS NETWORKS[1]

JEAN-MARIE BOURJOLLY, DANIEL TOMIUK

Concordia University, 1455, de Maisonneuve Blvd. W., Montreal PQ H3G 1M8, Canada

GENE H.M. KAPANTOW

University of Queensland, Australia.

ABSTRACT

The network design problems studied in this paper are typically known in the Telecommunications literature as the Concentrator Location, Terminal Assignment and Terminal Layout Problems. These are versions of well-known Operations Research models such as Capacitated Location, Capacitated Assignment, Capacitated Minimum Spanning Tree, and Vehicle Routing. We describe two Simulated Annealing algorithms for solving them and analyze the results obtained through computational testing.

Keywords: Telecommunications networks, Location, Assignment, Network Topology, Capacitated Minimum Spanning Tree, Vehicle Routing, Simulated Annealing.

RÉSUMÉ

Dans cet article, nous étudions des problèmes connus dans le domaine des télécommunications sous les noms de localisation de concentrateurs, d'allocation de terminaux et d'agencement optimal de terminaux. Ce sont des versions de modèles très connus de la recherche opérationnelle: localisation, allocation et arbre de recouvrement minimum en présence de capacités, et problème de tournées de vehicules. Nous décrivons deux algorithmes de recuit simulé pour résoudre ces problèmes et analysons les résultats obtenus.

1. INTRODUCTION.

Centralized telecommunications networks can be viewed as three-level hierarchical structures. At the first and lowest level we find input or output devices dispersed geographically at known locations. Although these devices are typically referred to as 'terminals' they do not solely include office and home computers. Rather, a multitude of devices are considered as terminal equipment. These include automatic teller machines for banking transactions, touch-tone telephones for bill payment and student registration, and even sensor devices such as certain alarm systems that are able to notify a security company located kilometers away.

At the second level in the hierarchy are line-concentrating devices with a limited number of possible sites for locating them. These sites may or may not coincide with the terminal sites. These intermediate line-concentrating devices are generically referred to as 'concentrators' and are connected to the central computer via high-speed lines such as fiber optic cables. When designers can identify a cluster containing terminals that are in close proximity to one another but relatively far away from the central computer, the use of a line-concentrating device may be justified. These devices consolidate low-speed lines into higher-speed lines, taking advantage of the economies of scale of cost versus capacity' ([21], p.179). When such a device can be placed in the vicinity of the cluster, terminals can be connected to it rather than span the distances needed to link each and every terminal to the central computer. Economies of scale dictate that in some instances the savings such devices generate outweigh the additional costs (acquisition,

[1]Recd. May 1998; Revd. Jan. 1999

installation, and connection) incurred. Finally, at the highest level of the hierarchy we find a single element referred to as the central computer or simply the centre. Once the number and locations of concentrators are known, and each terminal has been assigned exclusively to one of the concentrators or to the central computer, we must then determine how to lay out our lines so that every terminal will be connected into the network. Both terminals and concentrators can either be connected in a 'point-to-point' or in a 'multipoint' fashion. '*Point-to-point*' simply means that a terminal is directly linked either to a concentrator or the central computer, and therefore does not share a line with any other terminal in the network. For a concentrator, a 'point-to-point' connection implies a direct link to the central computer Alternatively, '*multipoint*' lines, also known as '*multidrop*' lines, consist of many terminals or concentrators linked together onto one line in order to further lower costs.

A limit is typically imposed on the number of components that may share a communication medium. This limit may be a directly imposed one due to reliability and efficiency concerns. In fact, when the number of terminals on a line increases for example, not only does each terminal's average access to the line decrease, but also, the potential number of 'downed' terminals in the system rises in case of a link failure. Furthermore, the number of components on a line is affected by the amount of data exchanged via the line. This exchange of information consists of sending and receiving messages over various types of medium intermittently with a transmission time usually lasting no more than a few milliseconds. By knowing the average amount of data exchanged *(traffic or weight)* between each terminal and the central computer for example, it becomes possible to interconnect these components at a minimal cost given the type of network configuration desired and data transmission capabilities of the communication medium. When the data communication medium linking the network components is a line (a twisted-pair wire or a coaxial cable as opposed to radiated media such as satellite or microwave technology), the cost of the network is a function of the aggregate geographical distance the connection links must span in order for each terminal to be able to exchange data with the central site.

In the following sections, the Centralized Telecommunications Network design problems will be addressed in further detail:

> *Concentrator Location* — Deciding how many concentrators to use, if any, and where to place them.
>
> *Terminal Assignment* — Determining what terminals will be serviced by each concentrator, given that a concentrator is limited in the number of terminals and amount of traffic that it can accommodate and assuming that the concentrator locations are known.
>
> *Terminal Layout* — Selecting a cost effective way to connect the terminals to their assigned concentrators with possible configurations being the star, the tree, the bus, or the loop.

All three problems can be linked to other well-known Operations Research models.

In this paper, we present a software package that uses Simulated Annealing in order to solve the above problems. Our goal was to produce the prototype of a Decision Support System that has graphical capabilities, that is flexible and user-friendly, and at the same time, that provides solutions that are better than heuristics commonly used in telecommunications, such as the ADD, the DROP, and the Esaü-Williams [13] algorithms. In Figure 9, an example of the graphical capabilities of the package is presented. Our prototype is designed to be a general-purpose application in which accuracy has to be somewhat sacrificed for speed. But nevertheless, from a comparison

with benchmark results, it does appear to be competitive with some of the best currently available solution methods.

2. THE EVOLUTION OF CENTRALIZED NETWORKS AND THE EMERGENCE OF DESIGN STRATEGIES.

2.1. Line-Concentrating Devices

In their infancy, centralized telecommunications networks consisted of at most a few hundred users located at close proximity to the central computer. Therefore, simply connecting each terminal to the central computer in a point-to-point fashion could effectively provide telecommunication services. In such configurations, each terminal completely monopolized the low-capacity line connecting it to the centre. Today, a centralized network arranged in such a fashion is commonly said to have a '*star*' topology. As centralized networks evolved and expanded, the distances and the number of terminals increased, making the star topology very costly. The introduction of line-sharing techniques (*multiplexing* and *concentration*) allowed designers to consolidate several low-speed lines onto one higher-speed line. With such technology, cost savings were made possible through economies of scale.

Multiplexing techniques divide a communication link into segments, each of which can carry information coming from a separate source. Each segment is separated from other segments by either *time-division multiplexing* (where each data source is allocated a time slot) or *frequency-division multiplexing* (where data sources are transmitted over different frequencies). One multiplexer at the 'remote site' allows multiple signals to be transmitted over a single link by merging all incoming terminal lines into one line. The combined data is then transmitted over this single link to the central computer, also known as the host. At the host end, a second multiplexer separates the data and distributes it among the outgoing terminal lines. *Concentrators* are also line sharing devices and their primary function is the same as that of multiplexers. But unlike multiplexers, concentrators are computers, and therefore, may have auxiliary storage for use in support of an application, although in recent years, additional functions have been added to multiplexers that have narrowed the differences between multiplexers and concentrators. With today's technology, the most noticeable difference that remains between the two is that concentrators are used one at the time while multiplexers are used in pairs. No matter what technique is chosen, both make line sharing transparent to the users; in essence they create the illusion and feel of a point-to-point connection. Throughout this paper, the word 'concentrator' will be used as a generic term for consistency reasons.

A point was reached where terminals could be connected to a device relatively close-by, and their low-speed lines consolidated onto one high-speed line that stretched from the device to the central computer. Today's literature typically refers to these types of centralized networks as 'Star-Star', since not only is each terminal directly connected to its respective concentrator, but also, each concentrator monopolizes the high-capacity line linking it to the central computer. An illustration is given in Figure 1.

2.2. Multipoint Lines.

As centralized networks grew increasingly larger, even the Star-Star's configuration cost became prohibitively high. Besides the costs associated with multiple spanning lines across long distances, other considerations include the fact that human beings have significantly slower response times when compared to computers, typically resulting in a very low utilization level of point-to-point connections. By grouping terminals into a multipoint configuration, it became possible to take advantage of the low line

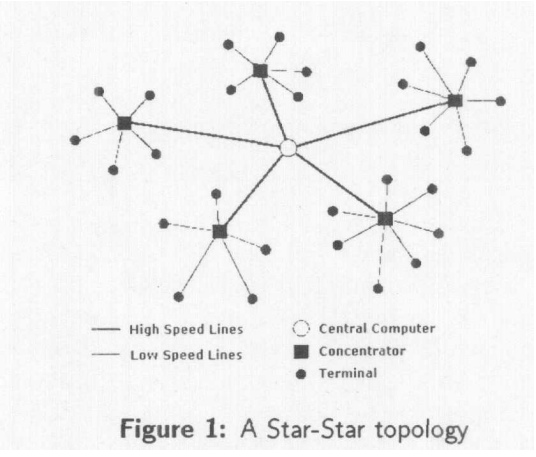


Figure 1: A Star-Star topology

utilization that normally occurs in the star topology. Moreover, since the cost of a network is directly related to the distance that all the lines must span in order to interconnect all of its devices, allowing several devices to share a line could inevitably lower costs. Furthermore, additional savings could be obtained from the decrease in hardware that would be required. In fact, if modems were used, fewer of them would be needed. Note that for each point-to-point connection one pair of modems is necessary, while in multipoint configurations one modem per terminal plus one for each line at the host (either the central computer or a concentrator) would suffice. And if the terminals are sufficiently close in proximity, their individual modems can be replaced by a *cluster controller modem* that supports several terminals at once. Generally, if the amount of data exchanged between a terminal and the central site is quite large, or if the need for security is high such as in certain government institutions, the terminal and the host are typically connected in a point-to-point fashion, where the terminal completely monopolizes the transmission line connecting it to the central site. However, if this is not the case, economies of scale dictate that in order to take advantage of commercially available lines whose maximum capacities are typically gauged with discrete values such as 14400, 28800, 57600 bits per second, one connects several computers onto a 'high-capacity' line, where the line is shared by several terminals at once.

As we have seen, designing a centralized computer network breaks down into figuring out where to place the concentrators, what terminals to assign to them, and finally how to arrange each cluster of assigned terminals onto multipoint lines. However, it must be noted that the strategy of laying out multipoint lines does not only apply to terminals assigned to a concentrator or the central computer. Concentrators themselves may be considered as terminals that exchange very large amounts of information. In such a case, the solution methods for the Terminal Layout Problem can also accommodate designers who wish to place concentrators on multipoint high-speed lines that would then be connected to the central computer, a strategy that would allow even further cost reductions.

3. CENTRALIZED TELECOMMUNICATION NETWORK DESIGN PROBLEMS.

3.1. Location — Allocation Problems.

The decision of where to place concentrators and how many of them to use depends on the number of potential locations and the amount of traffic any concentrator can handle. The limitation in the number of potential sites is typically due to security,

accessibility, and environmental factors that restrict availability sometimes even when the potential site coincides with an existing terminal location. Furthermore, assessing the savings obtained by placing concentrators amid terminals both depends on the concentrator locations and the terminal-concentrator assignments. Therefore, designers must attempt to solve these problems jointly.

3.1.1. The Concentrator Location Problem.

The concentrator location problem is a version of the '*Capacitated Plant Location Problem*' in Operations Research. Given are network points (typically warehouses or markets) having demands for a certain commodity and several sites for possible plant locations. There exists a fixed cost of opening a plant with a certain capacity at a given site. The shipping costs from plant locations to demand points for each unit of the commodity are known. The problem is to determine at which sites to open plants so as to minimize the total cost consisting of building and commodity shipment costs. For more information on the Capacitated Plant Location Problem, we refer the reader to [25]. Similarly, in the concentrator-location problem there is a limited number of terminals that can be accommodated by each concentrator. Boorstyn and Frank [6] identify this limit as a function of 'the limitations in buffer space, input ports, addressing structure, to the finite capacity of the line from concentrator to central site which restricts the amount of traffic from all the assigned terminals that can be handled by each concentrator, and to the share of resources used by the polling scheme' [p.31]. Because of the complicated nature in which these factors combine to estimate the number of terminals a concentrator can handle, research in this area usually assumes that the concentrator's capacity is already known. In the simple model, either one concentrator 'size' is considered where all concentrators are assumed to have the same capacity constraint or even more simply capacities can be utterly overlooked. If a capacity constraint is imposed on the concentrators then the sum of the weights (amount of data) associated with the terminals must be smaller than or equal to the maximum transmitted data capacity that the concentrator placed at a certain location can accept. A further restriction can be imposed on the number of terminals that each concentrator can handle. Well-known algorithms used for locating concentrators are the COM (Centre of Mass), ADD, and DROP algorithms (see [21], e.g.).

The COM algorithm tries to identify natural clusters of traffic. It begins by considering each terminal as a cluster by itself, and then, iterates to create new clusters by combining existing clusters that are close to one another subject to some given constraints. Typically, these constraints include the desired weight of a cluster, a distance limit between any two clusters to be combined, and the desired number of clusters. To illustrate how the COM algorithm works consider the following:

Assume that for each terminal i, we have its coordinates (x_i, y_i) and weight w_i. If terminals i and j are to be combined, the new cluster is represented by their centre of mass. The weight w_k of the newly created cluster k, is the sum of w_i and w_j. After terminals i and j are combined, they are eliminated from future consideration and replaced by terminal k.

The ADD algorithm starts with all terminals connected to the central computer. For every potential concentrator examined, the algorithm computes the savings that can be made if it is added to the solution. The order in which the first and remaining concentrators are selected is dictated by the largest savings obtained by individually considering each concentrator into the solution and repeating the operation until no additional savings can be found. Needless to say that the number of terminals assigned to a concentrator will inevitably depend on the maximum capacity that the concentrator can support. The major limitation of the ADD algorithm comes from the fact that once

a concentrator is accepted it cannot be dropped in a subsequent iteration. The DROP algorithm provides solutions that are comparable in quality to those obtained by the ADD. It starts by considering that all concentrators are in use. The algorithm then evaluates the savings obtained by dropping each concentrator one at a time. In the case of no capacity constraints on the concentrators, each terminal that has become isolated is simply connected to its nearest concentrator. The algorithm iterates until no additional savings can be found.

In the presence of capacity constraints, evaluating the savings obtained from dropping a concentrator also involves solving a terminal assignment problem. This is because a terminal that has been disconnected from the network may not simply be connected to the nearest of the remaining concentrators with the certainty that it will not violate the concentrator's capacity constraint. Therefore, in order to find the best savings from say, n drops, at each iteration, the algorithm will need to execute n terminal assignments, which will result in an extremely large running time for problems of moderate to large size. Consequently, for larger capacitated problems, it is wiser to use the ADD algorithm.

3.1.2. The Terminal Assignment Problem.

When a specific set J of concentrators have already been picked, the concentrator-location problem is reduced to the terminal assignment problem. For example, this situation occurs when a centralized network is already in place but needs to be expanded with the addition of newly purchased terminals. As in the previous problem description, cost is generally a function of the distance separating terminal T_i from concentrator C_j but unlike what happens in the concentrator-location problem, the cost of the concentrators can be ignored since it is actually a sunk cost. In the terminal assignment problem, each terminal has a capacity w_i associated to it and the sum of these capacities must not surpass the allowable maximum capacity W_j of the concentrator they are assigned to. Furthermore, each concentrator may be limited by the maximum number of terminals it can manage. Classical algorithms for solving this problem include the Original Greedy Algorithm, the Modified Greedy Algorithm with trade-off α [2] and the Alternating Chain Algorithm [21].

The original greedy algorithm has the advantages that it is easy to implement and requires little computing time. In the absence of capacity constraints, each terminal is simply assigned to its nearest concentrator. However, when capacity constraints are imposed, the algorithm tends to strand the terminals considered last which often results in a poor quality solution. In fact, the original greedy algorithm has a strong tendency to allocate the last few unassigned terminals to concentrators that are far away. To alleviate this problem, the Modified Greedy Algorithm with trade-off α was developed. The purpose of the modification is to give preference of connection to those terminals that have the greatest negative impact on the final solution of the original algorithm (referred to as *critical terminals*) because they are connected to their nearest concentrator. Instead of relying on connection costs as a criterion for choosing the order of assignments, the algorithm uses a trade-off function that biases the selection process to give preference to the critical terminals. The value of the parameter ($0 \leq \alpha \leq 1$) reflects this preference, where a specified value of '0' indicates the situation where no preference is given to the critical terminals and would therefore provide a solution identical to that of the original greedy algorithm. The Alternating Chain Algorithm classifies as a semi-greedy algorithm. Contrarily to the algorithms discussed above, the Alternating Chain Algorithm has the capability of reconsidering terminals that have already been assigned. If after p terminal assignments the remaining terminals cannot be assigned to one of the nearest concentrators without violating the capacity constraint,

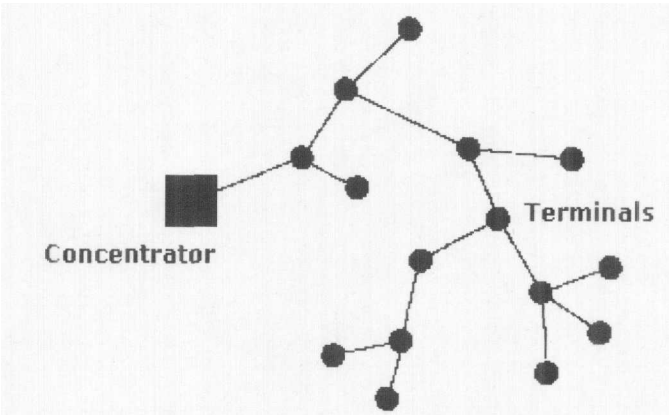


Figure 2: A multidrop line configured as a tree.

the algorithm will search for the least costly alternative for assigning the $(p-1)^{st}$ terminal to a nearby concentrator by relocating one or more of the terminals already assigned to that concentrator. In other words, a number of previous assignment decisions are retracted in order to 'make room' for the terminal under consideration. These alternatives are referred to as 'alternating chains' or 'augmented paths'. Each represents a set of relocations that must be performed to fit the terminal being considered. The alternating chain with the least cost is the one chosen.

3.1.3 Related Problems. When capacity constraints are relaxed, the problem becomes the *uncapacitated facility location problem.* The latter problem can further be modified by assuming that the network designers know in advance the number of facilities to be located. The problem then becomes the *p-facility location problem.* Moreover, if $d_j = 0$ for all j, where d_j is the cost of locating a facility as site j, we then have the *p-median problem.* All of these problems are members of a family of location problems [23].

3.2. The Terminal Layout Problem.

The solutions to both problems stated above generate clusters of terminals associated with each concentrator placed at a certain location. The next step in designing the network is to find the manner in which the terminals in these clusters must be connected to their assigned concentrators via some type of line configuration. Depending on the topology desired, designing multidrop lines give rise to well-known combinatorial optimization problems. The most common topologies are the *tree* (also referred to as *hierarchical*), the *bus*, and the loop (also known as the ring). In section 2.1, we described how concentrators and terminals were interconnected to create Star-Star centralized networks. The reader should note that with the use of multipoint line strategies, today's centralized networks have evolved and include combinations of concentrator-terminal topologies such as the Star-Loop, the Tree-Tree, etc.

3.2.1 The Tree Topology. In a tree topology, each terminal node may have several cascaded terminals attached to it. The tree network is illustrated in Figure 2.

Each node *may have* a certain number of child nodes ranging from 0 to a predetermined maximum. Each node in the subtree except for the root node *must have* a parent node. Trees are typically inexpensive structures, but have the obvious disadvantage that the failure of a single link may disconnect a considerable part of the network. The problem of configuring a multipoint line as a tree topology is better known as the Capacitated Minimum Spanning Tree (CMST) problem. The capacity constraint re-

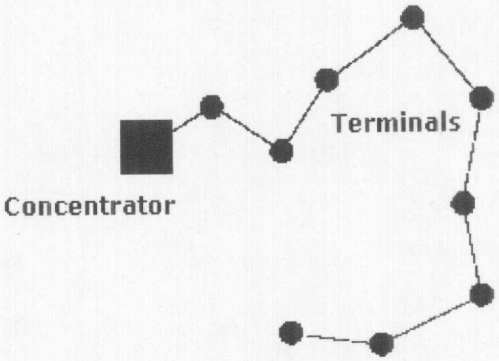


Figure 3: A Multidrop Line Using a Bus Topology

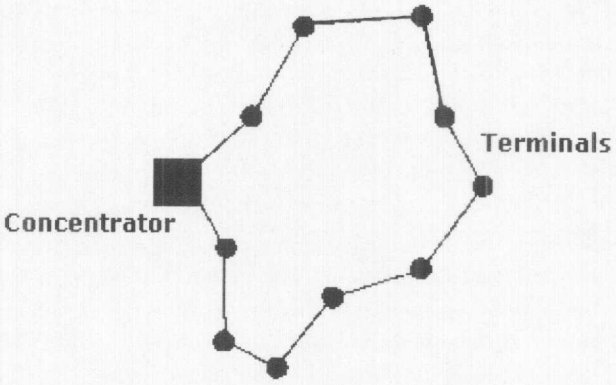


Figure 4: A Multidrop line using a Loop Configuration.

lates to the fact that the lines carrying the data to and from the terminals have limited transmission capabilities. Expressing this constraint can be done by considering that terminals have a traffic requirement which can be depicted as w_i (weight of terminal i) while a constraint *Wmax* can be placed on the maximum weight any subtree (multipoint line) can carry. This constraint indicates the maximum data transmission capabilities of the commercially available lines being considered and indirectly limits the number of terminals interconnected on any single line. It is possible to further constrain the problem by incorporating a specified limit on the number of terminals any communication channel (i.e., line) can handle. In the telecommunications industry, limiting the number of terminals on a single line is not uncommon practice. In fact, a major disadvantage associated with using multipoint lines is an increase in the user's waiting time when too many terminals share the same medium. In addition, limiting the number of terminals on the lines also minimizes the potentially detrimental effects associated with the failure of a single line within the network. Well-known CMST Heuristics include the Esaü-Williams [13], the Modified Kruskal [6], and Modified Prim [15] algorithms.

Esaü and Williams [13] used the notion of a 'trade-off value' given by the formula $t_{ij} = c_{ij} - c_{0i}$, where c_{0i} denotes the connection cost between terminal i and the central site '0', while c_{ij} represents the connection cost from terminal i to terminal j. Initially, each component i is made up of terminal i and the central site, in other words, terminals

are considered to be directly connected to the central site in a star configuration. For each pair of components (i, j), the trade-off is calculated representing the change in cost due to removing the central link connecting component i to '0' and forming the link $(i \rightarrow j)$. At each iteration, the algorithm finds the best trade-off t_{ij} and merges the component currently containing terminal i with the component currently containing terminal j. Before joining these components into one, the combined traffic is checked to verify that it does not exceed the amount that one line can handle. If the link is accepted and components i and j are merged, if $c_{0i} > c_{0j}$ then $c_{0i} := c_{0j}$ and if $c_{0i} < c_{0j}$ then $c_{0j} := c_{0i}$. The algorithm stops when no additional savings can be obtained by combining components together.

The Modified Kruskal Algorithm [6] begins by sorting all the links (least expensive first). The algorithm then traverses the list, one link at a time, evaluating whether each link can be performed without violating the line capacity constraint. The algorithm offers worse results than the Esaü-Williams algorithm for comparable CPU times. The Modified Prim algorithm [15] begins by assuming that the concentrator is the only node in the tree. It then finds the 'out-of-tree' node that is the nearest to the tree and connects it into the tree. The algorithm then proceeds to update the distance to the tree for all remaining 'out-of-tree' nodes. The algorithm accounts for line capacity constraints when updating the distances. As lines reach maximal capacity, any remaining 'out-of-tree' node's distance to the tree will not represent its shortest path to the tree. Therefore, connections made at the end of the process will be sub-optimal. The Modified Prim Algorithm typically gives worse solutions than those provided by the Esaü-Williams Algorithm.

3.2.2 The Bus Topology. Although the bus architecture resembles a loop (described in the next subsection) in which one of the ends is not connected to the concentrator, it is actually a tree topology where the number of child terminals is constrained to 1. Figure 3 illustrates a bus topology.

3.2.3 The Loop Topology. A loop structure is depicted in Figure 4. A line is a loop when its terminals possess a parent linking it into the network, and only one child. Furthermore, the first and last terminals in the structure must be linked to the central site thus creating a circuit.

The reliability of loops is better than that of trees. Although the cost is usually higher, the reliability issue often warrants the extra expenditure needed in wiring. 'All traffic ordinarily travels in one direction around the loop, say, clockwise. If, however, a link breaks, the terminals may have the capability of recognizing this and of temporarily using the remaining portion of the loop in the other direction. This can require a manual switchover and some rearrangement (usually at the software level) at the central site' ([21], p.196). Loop structures are usually used in local area network implementations that must be highly connected for the sake of reliability.

Loops may be created in two ways. The first approach consists of running an algorithm that solves the CMST. Furthermore, if the resulting subtrees are considered only as partitions, a *Travelling Salesperson (TSP)* algorithm may be used for each partition to find the least lengthy tour starting and ending at the root of the tree. In such a situation, the nodes in each partition are equivalent to the cities the salesperson must visit exactly once before returning home. The second approach consists of viewing the problem as a *Vehicle Routing Problem (VRP)*. It consists of 'finding a set of routes for a fleet of vehicles which have to service a number of stops from a central depot. It is assumed that every vehicle has the same capacity and the number of vehicles is unlimited. The vehicles depart and arrive at the depot. The demand quantity at each stop is known in advance and is deterministic. No single demand quantity exceeds vehicle

capacity.' ([4], p. 480). It can be added that the total demand quantity assigned to a given vehicle is no more than the capacity of the vehicle; the number of stops of each route is limited and one is seeking to minimize the overall distance that all the vehicles must cover. A heuristic such as the Clarke-Wright algorithm [10] can be used to obtain an initial solution to the VRP. This algorithm starts by considering each terminal on a separate loop. The algorithm then combines loops when the mergers offer a decrease in cost. Similarly to the Esaü-Williams algorithm, a trade-off is associated with link (i, j) given by $t_{ij} = C_{ij} - C_{i0} - C_{j0}$, where C_{j0} and C_{i0} denote the cost of connecting terminals j and i from the central site '0' respectively, while C_{ij} represents the connection cost between terminal i and terminal j. For more information on the VRP, we refer the reader to [24].

4. COMPLEXITY OF CENTRALIZED NETWORK DESIGN PROBLEMS.

In [27], Mirzaian and Steiglitz showed that almost all of the 'star-star' concentrator location problems are strongly NP-complete. However, if the capacity of the concentrators is less than two or all the connection costs are equal ($c_{ij} = c$, for all i and j), then these problems are solvable in polynomial time. They showed that except for these two cases, the concentrator location problems are NP-complete. In general, terminal assignment problems are also considered difficult. However, if all terminals have the same weight ($w_i = w$, for all i), they can be solved in polynomial time; otherwise they are NP-complete [2]. As for the terminal layout problem, Papadimitriou [28] has shown that the CMST is NP-complete when $2 < Q < n/2$, where Q represents the number of terminals a line can carry.

5. SIMULATED ANNEALING.

There are basically two types of solutions for the problems discussed above: (1) *exact solutions*, which are limited to small-size instances because of the inherent difficulties of this class of problems as recognized by the mathematical and computer science community, and (2) *suboptimal solutions* obtained using *heuristic techniques*. Heuristics have the advantage of requiring less computer running time while offering solutions that sometimes are very close to a global optimum. However, solutions obtained by simple local-search (usually of the *Greedy* type) represent only local optima. Well-known heuristics for the Concentrator Location and Terminal Assignment problems are the Greedy Algorithm with tradeoff , the ADD and DROP algorithms, and the COM algorithm, that are all described in sections 3.1.1 and 3.1.2. While for the Terminal Layout Problem, we find such heuristics as the *Esaü-Williams Algorithm* (see 3.2.1) for creating tree and bus structured lines, and the *Clarke-Wright Algorithm* (see 3.2.3) for laying out loop topologies. We refer the reader to [21] and [11], for a detailed explanation of these and other algorithms.

Simulated Annealing was first introduced to model the physical annealing of solids when Metropolis et al. [26] simulated a small displacement in individual atoms for each iteration of the simulation while monitoring the change in system energy the displacement produced. When the change corresponded to a decrease in energy, the resulting change was accepted, while increases in energy were only accepted with a certain probability. At each temperature level, a sufficiently large number of iterations were realized to attain thermal equilibrium, and the acceptance function guaranteed that the system was governed by the Boltzmann distribution.

Kirkpatrick et al. [22], and Cerny [7] independently proposed that the Simulated Annealing process could be applied to optimization problems by comparing the energy

states of the solid to an objective function to be minimized. In the analogy, 'the different states of the substance correspond to different feasible solutions to the combinatorial optimization problem, and the energy of the system corresponds to the function to be minimized.' ([12], p. 273).

In any implementation of the algorithm, an *annealing schedule* must be specified. This schedule stipulates the initial temperature setting (T_0), the reduction rate (α) in temperature (T) (where $0 < \alpha < 1$), the repetition (k) which denotes the number of changes to be attempted at each temperature level, and finally a stopping criterion referred to as 'epsilon' and denoted by 'ϵ' in order to terminate the program. The general Simulated Annealing Algorithm can be described in pseudo-Pascal as follows:

Initialize the parameters (k, T_0, α , and ϵ); Set $T = T_0$.
Repeat
  For $m := 1$ to k **do**
  begin
    Generate a neighbour state (current state $i \rightarrow$ new state j)
    Calculate the change in cost (Δ Cost = Cost i – Cost j)
    If Cost is ≤ 0 **then** make j the current state
    Else
    If $\exp(-\Delta \text{ Cost } /T) >$ random $[0, 1[$ **then** make j the current state
  end;
  $T := T * \alpha$;
Until $T \leq \epsilon$ (until the stopping criterion is reached);

The process involves accepting or rejecting a sequence of k neighbouring states at each temperature level (T), while dropping the temperature gradually at a rate of until the temperature reaches the stopping criterion. For the new state j to be a *neighbour* state of i, it must be reachable in exactly one move form state i, and it must be reversible. Furthermore, if S_i denotes a set of neighbour states reachable in exactly one move from i, then any state of S_i must be capable of being reached from any other in some number of moves. At high temperature levels it is possible for SA to accept new states (i.e., solutions) that produce relatively large increases in cost. As the system cools, the function becomes more likely to accept states generating small increases in cost rather than larger ones. When the temperature approaches zero, the majority of cost increasing moves are rejected. In other words, temperature influences the probability that a transition to a higher cost solution occurs.

Its potential to accept occasional increases in the cost function is what differentiates SA from simple local search algorithms because its trajectory includes migrations through sequences of solutions (including local extrema) in search of the global optimum. Contrarily to simple local search (greedy) algorithms, the SA's sporadic increases in the cost function allow it to 'jump out' of local minima traps.

6. ALGORITHMS AND RESULTS.

The schedules for our programs were arrived at by comparing the results obtained from various combinations of parameters settings on test problems of various sizes. From this process of trial and error, annealing schedules were determined for the concentrator location, terminal assignment, and terminal layout problems. Annealing schedules offering the most savings for each problem size considered were selected and utilized to compute the results given in this section. All programs were coded in Delphi™ and designed to run in the Windows™ environment.

6.1 Concentrator Location.

For both the Concentrator Location and Terminal Assignment Problems, all experiments were carried out on a PC with an Intel Pentium 75 MHz processor. The initial solution used by our algorithm for the concentrator location problem is calculated by the ADD Algorithm described in 3.1.1. Neighbor solutions for the concentrator location problem are generated in three ways. The first consists of considering the effects of 'dropping' a concentrator chosen at random from the current solution. Consequently, all terminals that were connected to that concentrator are reassigned to the remaining concentrators based on their availability and proximity. Note that for 'dropping', if a terminal cannot be connected to any of the other concentrators due to capacity constraints, a line will have to span to the central computer in order to connect the terminal into the network. The second method consists of 'adding' a concentrator at one of the possible but unoccupied randomly chosen concentrator locations. Once again, reassigning the terminals in the network is necessary in order to evaluate whether the added concentrator provides a better solution. Finally, the third neighbor generating method is a combination of the previous two. A concentrator that is part of the current solution is chosen at random and dropped while another concentrator is added into the network at a randomly selected but unoccupied location. Our algorithm selects the best of the possible three neighbor solutions. It should be noted that other neighbour generation methods were tried but that randomly choosing each of the two concentrators to be added and dropped gave us the best results. For example, an alternate procedure considered for dropping concentrators was to select the least significant concentrator (i.e., the concentrator that has the least contribution in lowering the configuration cost) where the contribution of each concentrator was calculated as $Q_j = \sum_{i \in I(j)} (c_i - c_{ij}) - d_j$ (where c_i = cost of connecting terminal i to its best available concentrator other then concentrator j, c_{ij} = cost of connecting terminal i to concentrator j, d_j = cost of locating concentrator j, and $I(j)$ represents all the terminals currently connected to concentrator j).

6.2 Results for the Concentrator Location Problem.

Two experiments were conducted for the Concentrator Location Problem. First, our algorithm was compared to the results of the ADD heuristic which was used to calculate the initial solutions. In addition, we also compared the performance of our program to the results generated by the program called SITATION written by Daskin [11] for solving uncapacitated facility location problems. For the capacitated problem, data sets were generated at random as xy-coordinates indicating the location of the central computer, n terminals, and m potential locations for the concentrators. We assumed that the concentrators to be located were all identical in terms of capacity and installation cost. Therefore, the cost for locating a concentrator at site j was determined only by its distance to the central computer. The capacity of each concentrator was assumed to be 12, and the weight of each terminal was randomly generated between 1 and 2. The cost of a high capacity line was assumed to be twice the price of a low capacity line. Moreover, the first half of the terminal sites were considered as the potential sites of the concentrators to be located.

In terms of (n, m), the test files were of size (100, 50), (200, 100), (300, 150), and (400, 200). For each one, five different data sets were created. The improvements over the ADD algorithm are given in Table 1.

Finally, we compared our program with SITATION. As mentioned above, this program solves a number of uncapacitated facility location problems. One of these is the uncapacitated facility location problem (UFCLP). This problem is very similar to the

Problem Size	Average improvement over the ADD algorithm (%)	Simulated Annealing Average time (in seconds)
(100, 50)	**2.83**	63
(200, 100)	**4.80**	500
(300, 150)	**3.41**	1849
(400, 200)	**5.12**	5116

Table 1: Improvement over the ADD algorithm.

SITATION Exchange (ADD-Based) Average Improvement (%)	SITATION Exchange (DROP-Based) Average Improvement (%)	Simulated Annealing	
		Average Improvement (%)	Average Time (secs)
2.86	2.85	**3.42**	1194

Table 2: SITATION and Simulated Annealing Improvements over the ADD Algorithm.

Concentrator Location Problem, where concentrators act as the facilities and the terminals act as the demand nodes. The only difference, it seems, is that the centre does not exist in the UFCLP. We discovered however, that by setting the connection cost between concentrators and the central computer to '0', our program could mimic this characteristic. In addition, the capacity of each concentrator was set large enough to handle the uncapacitated nature of the UFCLP. Moreover, all demand nodes were considered as the potential location of the facilities.

Ten different data sets of size (150, 150) were randomly generated in order to make the comparison. This size limitation was not intentional, instead, it simply reflected SITATION's limitations in problem size. The distances between the network nodes were randomly generated between 10 and 150, while the demand (i.e., weight) of each terminal was randomly set between 10 and 25. In addition, the fixed cost for installing a facility (i.e., a concentrator) was randomly generated between 500 and 1000.

The SITATION program provides several algorithms for solving a problem. All of them were applied to the test data in order to get the best one. The two exchange-based algorithms provided the best results. These two algorithms differ only in the manner in which they calculate the initial solution. The first is based on the ADD algorithm, whereas the second uses the DROP algorithm to obtain an initial solution. We then compared these results to our SA algorithm. The summary of the comparison is presented in Table 2.

6.3 Terminal Assignment

For the terminal assignment problem, the initial solution for our Simulated Annealing

Problem Size	Average improvement over greedy algorithm with tradeoff α (%)	Average Time (secs)
(100, 20)	**4.02**	21
(200, 40)	**5.51**	126
(300, 60)	**4.79**	330
(400, 80)	**6.99**	698
(500, 100)	**6.22**	1087

Table 3: Improvement over the modified greedy algorithm.

algorithm was calculated using the modified greedy algorithm [21] also known as the 'greedy algorithm with tradeoff α' [2] discussed in 3.1.2. Our terminal assignment algorithm considers both 'simple moves' (i.e. reassigning a terminal from one concentrator to another) and 'swap moves', consisting of exchanging two terminals between two different concentrators. The algorithm first selects two random concentrators where at least one of the two has a minimum of one terminal connected to it. When both concentrators are servicing at least one terminal respectively, the algorithm selects a move based on the best savings that can be attained. If only one of the two concentrators has terminals assigned to it then the 'swap move' option is ignored. If there are no actions that can save money, the move with the least cost increasing effect is considered for selection by the Simulated Annealing program. For simple moves, a terminal can only be reassigned from one concentrator to another if the latter has sufficient capacity to accept it. Invariably, the possibility of swapping only occurs if both concentrators can accommodate the other's terminal.

6.4 Results for the Terminal Assignment Problem.

For the Terminal Assignment Problem, we compared the results obtained from our algorithm to those given by the modified greedy algorithm. Our test data consisted of 25 randomly generated files of xy-coordinates representing the locations of a central computer, n terminals, and m concentrators. In terms of (n, m), the test data included 5 files of size (100, 20), 5 of size (200, 40), 5 of size (300, 60), 5 of size (400, 80), and 5 of size (500, 100). Capacities for all concentrators were uniform and equal to 12, while terminal weights varied between 1 and 3. Improvements were obtained for all problems and are shown in Table 3.

6.5 Terminal Layout

For the terminal layout, the Simulated Annealing algorithm was run on a portable computer with a Cyrix chip running at 200MHz. All results and execution times are reported herein.

6.5.1 Tree Topology. The initial solution for the CMST is obtained with the Esaü-Williams algorithm described in 3.2.1. The neighbour solutions are obtained by manipulating the order in which the Esaü-Williams algorithm links terminals to one another to create multipoint lines. This technique is related to algorithms of heuristic repetition first developed by Karnaugh [19] and Kershenbaum, Boorstyn, and Oppenheim [20], which are known in the literature as Second Order Greedy Algorithms (SOGA). However, to our knowledge, we are the only ones to have embedded such a technique within a Simulated Annealing Algorithm in order to generate neighbourhoods. As the Esaü-Williams algorithm proceeds, subtrees may fill up by reaching the maximum line

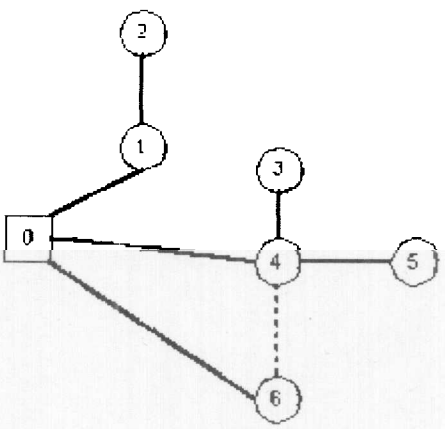


Figure 5: The Esaü-Williams Solution Showing a Capacitated Minimum Spanning Tree.

0	13	26	33	30	48	35
13	0	**10**	13	18	36	34
26	10	0	21	32	45	49
33	13	21	0	**6**	19	26
30	18	32	6	0	**18**	19
48	36	45	19	18	0	21
35	34	49	26	19	21	0

Table 4: A cost matrix used to calculate the Esaü-Williams solution for a CMST problem.

Iteration	Connection		Savings
	From	To	
1	5	4	$30 = 48 - 18$
2	3	4	$24 = 30 - 6$
3	2	1	$16 = 26 - 10$

Table 5: The iterations taken by the Esaü-Williams algorithm to arrive at a solution.

weight constraint (*Wmax*) or a node may not accept any additional child nodes without violating the maximum number of terminals per line constraint. From this point on, the solution will start degenerating away from a minimum spanning tree (MST) because the algorithm is unable to make the best multipoint connections for the remaining terminals. Typically, links created towards the end of the run tend to be non-optimal.

By forcing the Esaü-Williams algorithm to link a node before it would normally be considered for connection, it is possible to generate a neighbour solution with a lower cost. To illustrate the point, consider Table 4 showing a symmetrical distance matrix for a problem with Wmax equal to 3 where all the terminal weights are equal to 1. The Esaü-Williams solution is shown in Figure 5. Initially, the algorithm starts with the star topology where all terminals are connected directly to the central node '0' with a cost

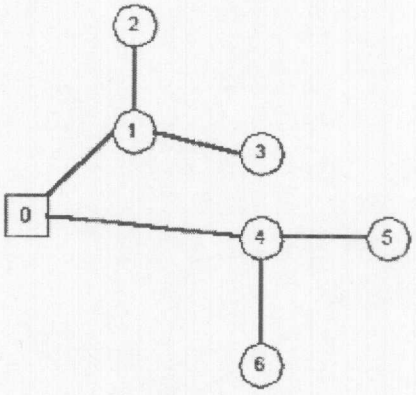


Figure 6: The Modified Esaü-Williams Solution Showing a Capacitated Minimum Spanning Tree.

Iteration	Connection		Savings
	From	To	
1	5	4	$30 = 48 - 18$
2	6 (*modification*)	4	$16 = 35 - 19$
3	3	1	$20 = 33 - 13$
4	2	1	$16 = 26 - 10$

Table 6: The iterations taken by the modified version of the Esaü-Williams to arrive at a solution.

of 185 (13 + 26 + 33 + 30 + 48 + 35). With each iteration of the Esaü-Williams algorithm a new link is created merging terminals into components. By saving the order in which the algorithm links (or merges) one terminal with another, a hierarchy of connections can be recorded. For this example the Esaü-Williams connections are shown in Table 5.

After iteration 3 in Table 5, the algorithm cannot keep merging the terminals without violating the line capacity constraint ($Wmax = 3$). The Esaü-Williams algorithm, therefore, stops executing and returns a solution of 115 (185-30-24-16). Terminals 1, 4, and 6 remain directly connected to the centre (central node 0).

However, the reader should note that replacing the link (6-0) by a link from terminal 6 to terminal 4 would yield an additional decrease in cost but cannot be done without violating the *Wmax* constraint (see Figure 5).

We suggest that by forcing the Esaü-Williams algorithm to consider terminal 6 higher up in the hierarchy of connections, a lower cost solution can be obtained. The steps taken by the modified version of the Esaü-Williams algorithm can be illustrated in Table 6. The modified Esaü-Williams solution obtained is shown in Figure 6.

Table 6 shows that at iteration 3, a connection from terminal 3 to terminal 4 would now violate the constraints, therefore terminal 3 is linked to terminal 1. This modified version of the Esaü-Williams algorithm returns a cost of 103 (185-30-16-20-16) or a saving of 12 over the original Esaü-Williams solution.

To understand how we obtained neighbour solutions, consider Table 7 showing an array representing the connection hierarchy of a solution for a problem consisting of 10

From terminal	7	6	4	9	1	3	**2**	**5**	**8**	**10**
To terminal	4	2	3	10	2	8	**0**	**0**	**0**	**0**

Table 7: The connection hierarchy showing the Esaü-Williams solution for a problem consisting of 10 terminals.

From terminal	7	**6**	**4**	**9**	1	3	2	5	8	10
To terminal	4	**2**	**3**	**10**						

Table 8: Illustration of Step 2 for the modified Esaü-Williams algorithm.

From terminal	7	**9**	**2**	**10**						
To terminal	4	?	?	?						

Table 9: Illustration of Step 3 for the modified Esaü-Williams algorithm.

terminals. The array represents the creation order of the links between the terminals. With the Esaü-Williams solution, the hierarchy of connections in the array represents (from left to right), in terms of savings, the best feasible connections obtained by the algorithm. The bold numbers inside the array represent the links (from the star topology) that could not be replaced due to the constraints imposed on the CMST.

Our neighbour solution generation process for a hierarchy array of size n (where n equals the number of links, including those from terminals to the central computer) was performed as follows:

Step 1. Select a number of modifications (M) that will be made to the Esaü-Williams algorithm (where $M \geq 3$ and $M \leq 0.10 \cdot n$ if $n \geq 30$). In this example, $M = 3$.

Step 2. In the first half of the connection hierarchy of the current solution, select M consecutive connections starting at $m1$, where $m1$ represents the leftmost connection chosen. In our example, $m1 = 2$ (or position 2 in the array). The bold numbers in Table 8 represent the M consecutive connections.

Step 3. Starting at $m1$, randomly consider terminals to the right of $m1$ as replacements for $m1, m1 + 1, \ldots, m1 + M - 1$ (from Table 8, potential terminals include 6, 4, 9, 1, 3, 2, 5, 8, and 10). A possible result of this random process is given in Table 9.

Step 4. Generate a number p of neighbour solutions (i.e., the neighbourhood of the current solution) by:

(i) Keeping the links in the current solution up to $m1 - 1$, then
(ii) Forcing the Esaü-Williams algorithm to consider the three terminals that replace $m1, m1 + 1, \ldots, m1 + M1$ for least cost connection into the network, and finally,
(iii) Allowing Esaü-Williams to resume link selection normally afterwards.

Step 5. Select the least costly solution obtained in step 4.

Problem Instances	Esaü-Williams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
te40_1, $Wmax = 3$	1215	**1192**	102	1196	642
te40_2, $Wmax = 3$	1144	**1117**	44	1122	577
te40_3, $Wmax = 3$	1146	**1115**	34	1119	645
te40_4, $Wmax = 3$	1156	**1144**	73	1146	621
te40_5, $Wmax = 3$	1147	**1115**	55	1123	608
te40_1, $Wmax = 5$	857	875	64	857	274
te40_2, $Wmax = 5$	827	812	44	**809**	235
te40_3, $Wmax = 5$	820	822	39	**807**	270
te40_4, $Wmax = 5$	854	**835**	68	837	277
te40_5, $Wmax = 5$	816	796	46	**794**	263
te40_1, $Wmax = 10$	649	**614**	46	633	113
te40_2, $Wmax = 10$	613	591	64	**573**	182
te40_3, $Wmax = 10$	596	591	45	**583**	104
te40_4, $Wmax = 10$	638	608	69	**600**	105
te40_5, $Wmax = 10$	597	**572**	72	581	93

Table 10: Results for CMST unit demand problems consisting of 40 terminals where the central vertex is located at the end of the vertex scatter.

As we have demonstrated in this section, the modification to the Esaü-Williams algorithm proposed herein may at times return a lower cost solution. The Simulated Annealing program into which this neighbourhood generation process is embedded accepts any lower cost solution identified in step 5 automatically, but also accepts higher cost solutions with a certain probability.

6.5.2 Results for the Tree Topology. The Simulated Annealing programs for the tree topology were tested on two CMST data sets available from the OR-library of Beasley (e-mail: o.rlibrary@ic.ac.uk). The first consists of data with unit demand. The results for problems te40_k, te80_k, tc40_k, and tc80_k ($k = 1, \ldots, 5$) are given in Tables 10, 11, 12, 13. Problems 'te' refer to matrices where the central vertex (i.e , a concentrator or the central computer) is located at the end of the vertex scatter, and problems 'tc' refer to data where the central vertex is at the center of the vertex scatter. The second data set includes non-unit demand problems cm50_rv and cm100_rv (where v = 1, . . , 5), and results are reported in Tables 14 and 15. Included for all these datasets, are the results from the Esaü-Williams Algorithm and those of Sharaiha et al. [29] obtained with a tabu search heuristic. Execution times for the Esaü-Williams algorithm are not reported since solutions typically took less than a second to be calculated. All execution times are in seconds and those reported herein for the Tabu Search algorithm were taken from Sharaiha et al. [29] who ran their algorithm on a Silicon Graphics Indigo workstation (R4000, 100MHz).

When compared to the Tabu Search algorithm, our Simulated Annealing algorithm did better in terms of cost on smaller sized non-unit demand problems (i.e., 'cm_50' problems) and on unit demand problems where the central vertex was located at the end of the vertex scatter (i.e., 'te' problems). Execution times for both the Tabu Search and the Simulated Annealing algorithms are somewhat comparable. Our Simulated Annealing algorithm typically took longer to execute for problem instances were maximum

Problem Instances	Esaü-Williams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
te80_1, $Wmax = 5$	2592	2570	419	**2559**	2151
te80_2, $Wmax = 5$	2631	**2574**	410	2602	2655
te80_3, $Wmax = 5$	2723	2741	254	**2665**	2430
te80_4, $Wmax = 5$	2630	2672	227	**2604**	2377
te80_5, $Wmax = 5$	2595	2557	205	**2514**	2305
te80_1, $Wmax = 10$	1735	**1688**	205	1709	813
te80_2, $Wmax = 10$	1787	**1678**	229	1697	903
te80_3, $Wmax = 10$	1828	1775	245	**1736**	902
te80_4, $Wmax = 10$	1691	1906	224	**1656**	813
te80_5, $Wmax = 10$	1731	1685	258	**1633**	725
te80_1, $Wmax = 20$	1336	1311	378	**1285**	351
te80_2, $Wmax = 20$	1295	1266	254	**1248**	329
te80_3, $Wmax = 20$	1340	1329	276	**1308**	353
te80_4, $Wmax = 20$	1349	1337	258	**1315**	385
te80_5, $Wmax = 20$	1271	1259	259	1259	306

Table 11: Results for CMST unit demand problems consisting of 80 terminals where the central vertex is located at the end of the vertex scatter.

line capacity was set to a relatively small number (i.e., $Wmax = 3$ for 'te' and 'tc' problems and $Wmax = 200$ for 'cm' problems). However, when the value of Wmax increased, our Simulated Annealing algorithm had comparable or shorter execution times than those reported for the Tabu Search Algorithm.

We note that Amberg et al. [1] also have reported results using Simulated Annealing for this problem. However, the manner in which the results have been reported (average relative improvements over Esaü-Williams and best results obtained from several runs with different parameters) makes it difficult to perform a meaningful comparison.

6.5.3 Bus and Loop Topologies. Similarly to the tree topology, the initial solution for the bus topology was generated by the Esaü-Williams algorithm. However, terminal nodes in the solutions were restricted to having only one child node. For the loop topology the Clarke-Wright algorithm [10] was used to obtain initial solutions. The process of generating a neighbour solution for these topologies begins by selecting 2 terminals at random denoted Ti and Tj,. By changing the link(s) that connect each of these two terminals to their respective lines we can consider two simple *'moves'* and one pairwise *'interchange'*. Of course, the terms 'moving' and 'interchanging' is figurative since the terminal locations are geographically fixed. The process is actually one of including or excluding terminals from lines by adding and dropping the links that connect them to their respective lines. Once a pair of terminals is selected, up to three potential neighbours can be generated by evaluating the change in total distance (cost) of:

1. moving terminal Ti to another line,
2. moving terminal Tj to another line, and
3. swapping both terminals.

Each new configuration produced in one of the ways described above that does not violate the topology or the maximum line weight (Wmax) and maximum number of

Problem Instances	Esaü-Williams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
tc40_1, $Wmax = 3$	777	**744**	39	761	253
tc40_2, $Wmax = 3$	749	**728**	42	735	244
tc40_3, $Wmax = 3$	728	**722**	43	725	229
tc40_4, $Wmax = 3$	804	793	39	**782**	246
tc40_5, $Wmax = 3$	760	**741**	31	745	224
tc40_1, $Wmax = 5$	595	**590**	37	595	121
tc40_2, $Wmax = 5$	588	585	40	**583**	116
tc40_3, $Wmax = 5$	602	**577**	98	585	104
tc40_4, $Wmax = 5$	645	**618**	41	623	113
tc40_5, $Wmax = 5$	615	**602**	57	611	117
tc40_1, $Wmax = 10$	516	**500**	36	506	63
tc40_2, $Wmax = 10$	505	490	33	490	65
tc40_3, $Wmax = 10$	517	**500**	36	508	66
tc40_4, $Wmax = 10$	524	513	36	**512**	66
tc40_5, $Wmax = 10$	540	504	38	504	70

Table 12: Results for CMST unit demand problems consisting of 40 terminals where the central vertex is located at the center of the vertex scatter.

Problem Instances	Esaü-Williams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
tc80_1, $Wmax = 5$	1182	**1133**	417	1163	852
tc80_2, $Wmax = 5$	1170	**1124**	301	1146	763
tc80_3, $Wmax = 5$	1146	**1095**	395	1118	680
tc80_4, $Wmax = 5$	1160	**1108**	214	1132	761
tc80_5, $Wmax = 5$	1344	**1324**	205	1332	750
tc80_1, $Wmax = 10$	931	**901**	244	915	329
tc80_2, $Wmax = 10$	917	**886**	201	897	321
tc80_3, $Wmax = 10$	912	**880**	273	899	369
tc80_4, $Wmax = 10$	924	**874**	1124	903	327
tc80_5, $Wmax = 10$	1092	**1005**	280	1038	365
tc80_1, $Wmax = 20$	856	**834**	246	842	265
tc80_2, $Wmax = 20$	856	**820**	220	832	265
tc80_3, $Wmax = 20$	852	**828**	199	836	269
tc80_4, $Wmax = 20$	860	**820**	209	830	289
tc80_5, $Wmax = 20$	969	**916**	232	941	276

Table 13: Results for CMST unit demand problems consisting of 80 terminals where the central vertex is located at the center of the vertex scatter.

Problem Instances	Esaü-Will-iams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
cm50_r1, $Wmax = 200$	1135	1180	61	**1118**	475
cm50_r2, $Wmax = 200$	1023	1061	76	**1001**	334
cm50_r3, $Wmax = 200$	1249	1229	74	**1205**	429
cm50_r4, $Wmax = 200$	834	811	119	**801**	215
cm50_r5, $Wmax = 200$	970	993	66	**949**	288
cm50_r1, $Wmax = 400$	731	726	57	**707**	121
cm50_r2, $Wmax = 400$	642	680	59	**639**	102
cm50_r3, $Wmax = 400$	741	780	61	**739**	118
cm50_r4, $Wmax = 400$	583	617	59	**572**	70
cm50_r5, $Wmax = 400$	643	628	67	**615**	93
cm50_r1, $Wmax = 800$	550	544	58	**516**	53
cm50_r2, $Wmax = 800$	531	542	66	**522**	35
cm50_r3, $Wmax = 800$	565	554	58	**541**	64
cm50_r4, $Wmax = 800$	514	472	52	**480**	45
cm50_r5, $Wmax = 800$	515	501	56	501	51

Table 14: Results for CMST non-unit demand problems consisting of 50 terminals where the central vertex is located at the end of the vertex scatter.

Problem Instances	Esaü-Will-iams	Tabu Search	Execution time for TS secs	Simulated Annealing	Execution Time for SA secs
cm100_r1, $Wmax = 200$	728	**551**	1154	682	2149
cm100_r2, $Wmax = 200$	800	**616**	747	695	2529
cm100_r3, $Wmax = 200$	750	**608**	747	697	2274
cm100_r4, $Wmax = 200$	625	**445**	482	544	1732
cm100_r5, $Wmax = 200$	637	**442**	821	571	1918
cm100_r1, $Wmax = 400$	375	**259**	431	293	775
cm100_r2, $Wmax = 400$	376	**278**	399	337	919
cm100_r3, $Wmax = 400$	363	**238**	793	312	799
cm100_r4, $Wmax = 400$	322	**223**	501	274	654
cm100_r5, $Wmax = 400$	323	**227**	436	270	733
cm100_r1, $Wmax = 800$	255	**182**	608	228	523
cm100_r2, $Wmax = 800$	235	**179**	396	198	510
cm100_r3, $Wmax = 800$	238	**175**	351	204	535
cm100_r4, $Wmax = 800$	248	**183**	356	220	521
cm100_r5, $Wmax = 800$	232	**187**	350	210	565

Table 15: Results for CMST non-unit demand problems consisting of 100 terminals where the central vertex is located at the end of the vertex scatter.

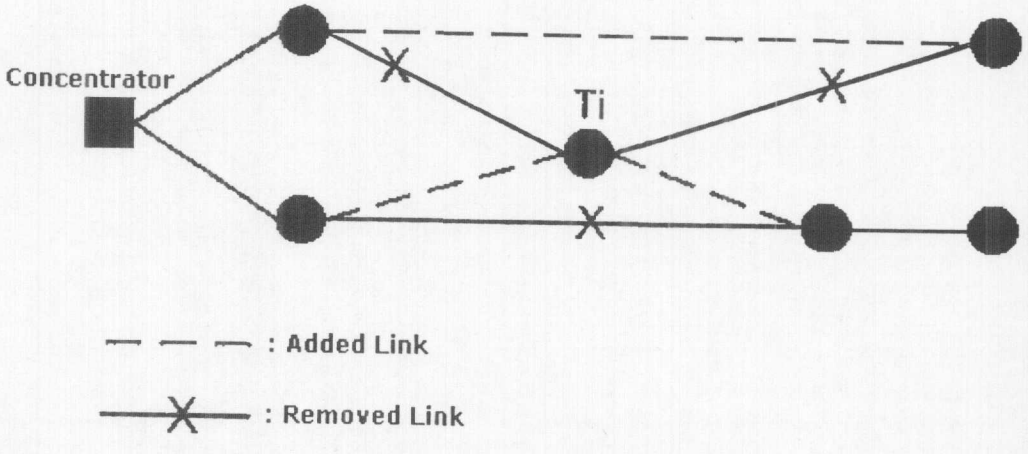


Figure 7: Moving terminal Ti to line J.

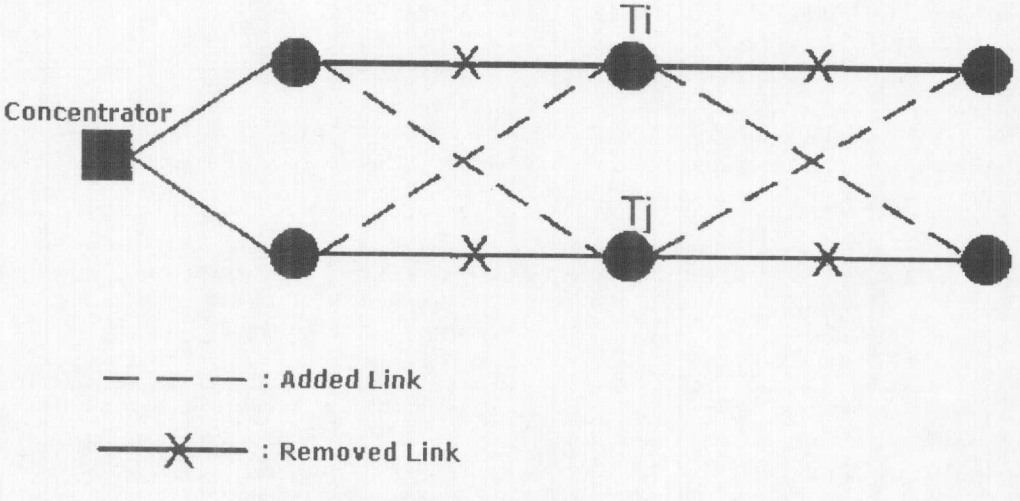


Figure 8: Swapping terminals Ti and Tj.

terminals per line constraints, is considered to be a valid neighbour solution. Figures 7 and 8 illustrate how neighbour solutions can be created by manipulating the links connecting the terminals.

6.5.4 Results for the Bus Topology. Data sets representing 50, 100, 150, and 200 terminals were created. In all, 20 sets were produced, 5 in each size category. All data sets were generated at random to represent the xy-coordinates of the terminals relative to a concentrator positioned at $(0, 0)$. In addition, each terminal (Ti) was randomly assigned a weight ranging between 1 and 10. The 'maximum line weight' constraint (Wmax) was set to 50 for problems consisting of 50 and 100 terminals, while for problems of 150 and 200 terminals, Wmax was set to 100. Thus, on average we would expect the resulting bus topologies to range between 10 and 20 terminals per line ($Wmax/averagew_i$) for the problems consisting of 50 and 100 terminals, and to range between 30 and 40 terminals per line for problems consisting of either 150 or 200 terminals. The bus topology problems were further constrained by imposing a limit on the maximum number of ter-

Problem Size	SA Average improvement over the Esaü-Williams (%)	Average time (secs)
(50)	**3.89**	10
(100)	**5.68**	15
(150)	**5.68**	70
(200)	**5.13**	81

Table 16: Improvement over the Esaü-Williams algorithm.

Problem Instances	Best Reported Result	Standard TABU-ROUTE Results	TABUROUTE Execution Time (mins)	Simulated Annealing	Simulated Annealing Execution Time (min:sec)
c50.dat	524.61*	524.61	6.0	524.61	2:13
c75.dat	835.32*	**835.77**	53.8	844.68	3:26
c100a.dat	826.14**	**829.45**	18.4	830.78	6:15
c100b.dat	819.56*	819.56	16.0	819.56	6:25
c120.dat	1042.11**	1073.47	22.2	**1046.72**	22:50
c150.dat	1029.64***	**1036.16**	58.8	1046.81	12:53
c199.dat	1300.89**	**1322.65**	90.9	1351.39	16:16

* Result obtained by both [16] and [30] using multiple runs.
** Result obtained by [30] using multiple runs.
*** Result obtained by [16] using multiple runs.

Table 17: Simulated Annealing results as compared to Capacitated Vehicle Routing problem benchmarks described in Christofides et al. [9].

minals any line could carry, these were a maximum of 10 terminals per line for problems of size 50 and 100, and a maximum of 15 terminals for problems of size 150 and 200. Table 16 includes the percentage of improvement for each problem size as compared to the Esaü-Williams algorithm, and the average SA running time.

On average, for all problem sizes considered, the SA algorithm outperforms the Esaü-Williams algorithm by 5.10 %.

6.5.5 Results for the Loop Topology. The data sets used for the loop topology were obtained at: http://www.idsia.ch/~eric/problemes.dir/vrp.dir/taillard.dir/.
These problems were originally described in [9]. The problems range in size from 50 to 199 terminals (cities) in addition to the central site (depot). The results from our Simulated Annealing algorithm are compared to the 'best results' found thus far for the same problems. Moreover, we compare our results to those obtained using TABUROUTE reported in [16]. The comparisons are presented in Table 17. The results for the Simulated Annealing were obtained on a portable computer with a Cyrix chip running at 200MHz and those from TABUROUTE on a Silicon Graphics workstation running at 36 MHz (5.7 Mflops).

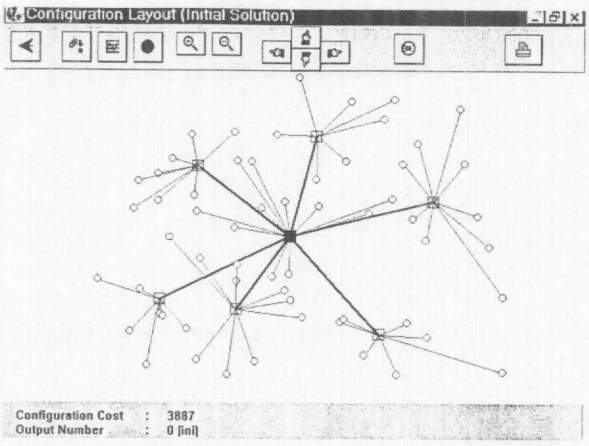


Figure 9: An Example of the Graphical Output Capabilities of our Software Package.

When compared to the Standard TABUROUTE results, our algorithm outperformed it only once (for the c120.dat problem). We believe that this may be due to the small number of test problems that we used to establish our annealing schedules. For each problem size, we randomly generated only 5 test problems. Furthermore, for each set of test problems, we manipulated the Simulated Annealing parameters no more than seven times and chose the parameter combination that offered the greatest average savings for all 5 test problems. We believe that with additional test problems and parameter manipulation, our program may provide better results that match or even surpass those reported by [16]. Nevertheless, our Simulated Annealing algorithm seems promising, the results indicate that some of our solutions (i.e., c50.dat and c75.dat) match the cost of the 'best reported results' obtained, as reported by [30] and [16].

7. CONCLUSION.

We developed the prototype of a Decision Support System that allows the user to simulate various scenarios in a user-friendly environment and to obtain quick solutions with a graphical display, while improving on the quality of the solutions that are provided by well-known heuristics commonly used in the Telecommunications industry.

Our results are competitive with some of the best currently available solution methods. Moreover, we are certain that additional computational experiments using other annealing schedules than the one we have tried could increase the quality of our SA algorithms.

Our programs were designed to run in the Windows environment. Special attention was given to make them intuitively easy to use and capable of loading data in either xy-coordinate or cost matrix formats. Furthermore, the graphical capabilities of our programs offer centralized network designers a means of visually assessing the solutions obtained. An example is given in Figure 9. The large node found at the centre of the display represents the central computer.

ACKNOWLEDGEMENTS

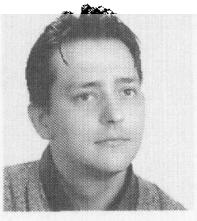
The authors wish to thank Professor Samuel Pierre for suggesting this research topic and four anonymous referees for their useful comments. They also gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada under grant number OGP0009359.

REFERENCES

1. Amberg, A., Domschke, W. and Darmstadt, S.V. (1996): 'Capacitated Minimum Spanning Trees: Algorithms Using Intelligent Search', *Combinatorial Optimization: Theory and Practice*, 1, pp. 9-40.
2. Abuali, F. N., Schoenefeld, D. A. and Wainwright, R. L. (1994): 'Terminal Assignment in a Communications Network Using Genetic Algorithms', Proceedings of 22nd ACM Computer Science Conference, Phoenix, Arizona, pp. 74-81
3. Balakrishnan, A., Magnanti, T. L., Shulman, A., and Wong, R. T (1991): "Models for Capacity Expansion in Local Access Telecommmnication Networks", *Annals of Operations Research*, 33, pp. 239 — 284.
4. Breedam, A V. (1995): 'Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing', *European Journal of Operational Research*. 86, pp. 480- 490.
5. Bohachevsky, I.O., Johnson, M.E. and Stein, M.L: 'Generalized Simulated Annealing for Function Optimization', *Technometrics*, 28, 1986, pp. 209-217.
6. Boorstyn, R.R., and Frank, H.(1977): 'Large Scale Network Topological Optimization', *IEEE Transactions on Communications*, Com-25, pp. 29- 47.
7. Cerny, V (1985): 'Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm', *Journal of Optimization Theory and Applications*, 45, pp. 41-51.
8. Chardaire, P., and Lutton, J. L. (1993): "Using Simulated Annealing to Solve Concentrator Location Problems in Telecommunication Networks". In R.V.V. Vidal (Ed.), Applied Simulated Annealing, Springer-Verlag, Berlin, pp. 175-199.
9. Christofides, N., Mingozzi, A., and Toth P. (1979): 'The Vehicle Routing Problem'. In N. Christofides, A. Magozzi, P. Toth, and C. Sandi (eds.), Combinatorial Optimization, Wiley, pp. 315-338.
10. Clarke, G. and Wright, J. (1964): 'Scheduling for Vehicles from a Central Depot to a Number of Delivery Points', *Operations Research*, 12, pp. 568- 581.
11. Daskin, M. S. (1995): *Network and Discrete Location: Models, Algorithms and Applications*, Wiley.
12. Eglese, R. W (1990): 'Simulated Annealing: A Tool for Operational Research', *European Journal of Operational Research*, 46, pp. 271-281.
13. Esaü, L. R., and Williams, K. C. (1966): 'On Teleprocessing System Design', Part II, *IBM System Journal*, 5, no. 3, pp. 142-147.
14. Garey, M. R., and Johnson, D. S. (1979): Computers and Intractability — A Guide to the Theory of NP-Completeness, Freeman, San Fransisco.
15. Gavish, B. (1991): 'Topological Design of Telecommunication Networks — Local Access Design Methods', *Annals of Operations Research*, 33, pp. 17 — 71.
16. Gendreau, M., Hertz, A., and Laporte, G. (1994): "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science*, 40, pp. 1276 — 1290.
17. Gouveia, L., and Lopes, M. J. (1997): 'Using Generalized Capacitated Trees for Designing the Topology of Local Access Networks', *Telecommunication Systems*, 7, pp. 315-337.
18. Kapantow, Gene H.M. (1996): "Solving Concentrator and Terminal Assignment Problems Using Simulated Annealing", M.Sc.A. Thesis, Concordia University, Montreal, Canada.
19. Karnaugh, M. (1976): 'A new Class of Multipoint Network Optimization.', *IEEE Transactions on Communications*, 24. pp. 500-505.
20. Kershenbaum, A., Boorstyn, R. and Oppenheim, R. (1980): 'Second-Order Greedy Algorithms for Centralized Teleprocessing Network Design', *IEEE Transactions on Communications*, 28, pp. 1835-1838..
21. Kershenbaum, A. (1993): *Telecommunications Network Design Algorithms*, McGraw-Hill, NY.
22. Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P. (1983): 'Optimization by Simulated Annealing', *Science*, 220, pp. 671-680.

23. Krarup, J., and Pruzan, M. (1990): 'Ingredients of Location Analysis'. In P.B. Mirchandani and R. L. Francis (eds.), Discrete Location Theory, Wiley, pp. 1-54.
24. Laporte, G., and Nobert, Y. (1987): 'Exact Algorithms for the Vehicle Routing Problem'. In Annals of Discrete Mathematics, 31, 'Surveys in Combinatorial Optimization', edited by S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, pp.147-184.
25. Mirchandani, P., and Francis, R.L., eds. (1990): Discrete Location Theory, Wiley.
26. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., and Teller, A. H. (1953): 'Equation of State Calculations by Fast Computing Machines', *Journal of Chemical Physics*, 21, no. 6, pp.1087- 1092.
27. Mirzaian, A., and Steiglitz, K. (1981): 'A Note on the Complexity of the Star-Star Concentrator Problem', *IEEE Transactions on Communications*, COM-29, pp. 1549-1552.
28. Papadimitriou, C. H. (1978): 'The Complexity of the Capacitated Tree Problem', *Networks*, 8, pp. 217- 230.
29. Sharaiha, Y. M., Gendreau, M., Laporte, G., and Osman, I. H. (1997): 'A Tabu Search Algorithm for the Capacitated Minimum Spanning Tree Problem', *Networks*, 29, pp. 161-171
30. Taillard, E. (1992): "Parallel Iterative Search Methods for Vehicle Routing Problems', Working Paper ORWP 92/03, Département de Mathématiques. Ecole Polytechnique Fédérale de Lausanne, Switzerland.
31. Tomiuk, Daniel B. (1996): "Using Simulated Annealing to Minimize the Cost of Multipoint Lines in Centralized Computer Networks. Implementation in WINDOWS3.1™", MScA. Thesis, Concordia University, Montreal, Canada.

Jean-Marie Bourjolly is Associate Professor at Concordia University (Faculty of Commerce and Administration, Department of Decision Sciences and MIS). His research interests include the applications of Operations Research to problems arising in the field of Telecommunications.



Daniel Tomiuk is a Ph.D. candidate (Management Information Systems) at Concordia University, Montreal, Canada. He received his MScA. degree (Decision Sciences and MIS) in Business Administration from Concordia University in 1997. His research interests include telecommunications and electronic commerce.

Gene Kapantow is a Ph.D. candidate at the University of Queensland, Australia, in the department of Geographical Sciences and Planning. He received the MScA. degree (Decision Sciences and MIS) in Business Administration from Concordia University in 1997 and the MS in Computer Science from the University of Indonesia, Jakarta in 1990. email: s801001@student.uq.edu.au