Pergamon

0969-6016(94)E00012-H

Int. Trans. Opl Res. Vol. 1, No. 3, pp. 317–336, 1994
Elsevier Science Ltd.
Printed in Great Britain.
0969-6016/94 $7.00 + 0.00

Capacitated Clustering Problems by Hybrid Simulated Annealing and Tabu Search

IBRAHIM H. OSMAN* and NICOS CHRISTOFIDES†

*University of Kent, U.K. and †University of London, U.K.

The capacitated clustering problem (CCP) is the problem in which a given set of weighted objects is to be partitioned into clusters so that the total weight of objects in each cluster is less than a given value (cluster 'capacity'). The objective is to minimize the total scatter of objects from the 'centre' of the cluster to which they have been allocated. A simple constructive heuristic, a λ-interchange generation mechanism, a hybrid simulated annealing (SA) and tabu search (TS) algorithm which has computationally desirable features using a new non-monotonic cooling schedule, are developed. A classification of the existing SA cooling schedules is presented. The effects on the final solution quality of the initial solutions, the cooling schedule parameters and the neighbourhood search strategies are investigated. Computational results on randomly generated problems with size ranging from 50 to 100 customers indicate that the hybrid SA/TS algorithm out-performs previous simulated annealing algorithms, a simple tabu search and local descent algorithms.

Key words: capacitated clustering problems, capacitated p-median, plant location, heuristic, simulated annealing, tabu search, local search

INTRODUCTION

The capacitated clustering problem, CCP, is the problem in which a given set of objects (or customers) is to be partitioned into a set of clusters. Each object has an associated weight (or demand) and must be assigned to exactly one cluster. Each cluster has a given capacity which must not be exceeded by the total weight of objects in the cluster. The dissimilarity measure is a cost (or distance) between any two objects. For a given cluster, a *centre* is that object of the cluster from which the sum of the dissimilarities to all other objects in the cluster is minimized. This sum is called the *scatter* of the cluster. The objective is to find a set of *centres* which minimizes the total scatter of all clusters. A pictorial representation of the CCP is given in Fig. 1.

Clustering of related objects is of practical importance and can be found in diverse fields such as biology, economics, engineering, marketing, operations research, pattern recognition and statistics. Furthermore, the CCP is a very special case of the capacitated plant location problem with single source constraints, and many other combinatorial problems. As a consequence, it can be shown to be NP-complete (Garey and Johnson, 1979). Optimization algorithms by general-purpose branch and bound codes, such as the CPLEX mixed integer optimizer are available though they are ineffective for large-sized instances of the CCP. It seems reasonable to devise approximate algorithms (or heuristics) which are effective in solving large-sized problems. There are two types of heuristics. The first type uses a tailored approach which constructs a solution from the data by looking to the characteristics of the problem to be solved. This type is known as *constructive heuristic* and is generally difficult to generalize to different applications. The second type uses an *iterative improvement* (or local search) approach which starts from an initial solution and iteratively attempts to improve upon it by a series of local changes. The main drawback of local search heuristics is that the search might terminate at a local optima which may be far from a global optima. The quality of the final solution depends on the starting solution and the rules used to generate neighbouring solutions. Recently, a great deal of attention has focussed on two local search heuristics when solving hard combinatorial optimization problems: simulated annealing, SA (Kirkpatrick *et al.*, 1983) and tabu search, TS (Glover, 1986). Both methods help reduce the effects of local optimality and produce near optimal solutions using strategies based on ideas from statistical physics and strategic oscillation ideas from the tenet of intelligent problem solving to explore the search space, respectively.

In this paper, we introduce a λ-interchange mechanism for the CCP which has similar properties to

Correspondence: I. H. Osman, Institute of Mathematics and Statistics, University of Kent, Canterbury, Kent, CT2 7NF, U.K.

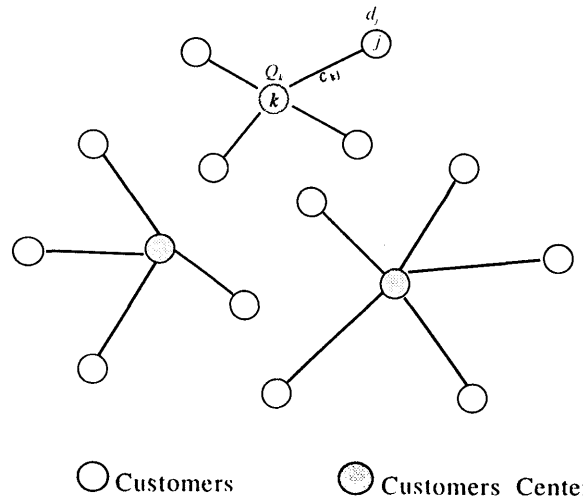


Fig. 1. The capacitated clustering problem.

those introduced by Lin (1965) and Lin and Kerrighan (1973) for the travelling salesman problem. Also, a new non-monotonic cooling schedule based on an oscillating strategy concept is introduced as opposed to classical monotonic cooling schedules in the SA literature. A hybrid SA/TS algorithm is proposed that combines the new cooling schedule with other elements from SA and TS philosophies. We investigate the effect of the initial solution on the final one, the influence of the SA cooling schedule on the quality of the final solution and the effect of the topology of a combinatorial problem on the behaviour of the algorithm.

The organization of this paper is as follows. In the first section, we give a formal problem definition, review some of its available solution methods and summarize its relationships to other combinatorial problems. In the second section, basic local search, simulated annealing and tabu search concepts are reviewed. SA cooling schedules are classified and a new cooling schedule is described. In the third section, a simple constructive heuristic and a λ-interchange mechanism are introduced. Our hybrid simulated annealing and tabu search implementation is also described. The fourth section contains the results of our computational study in which different search and selection strategies are investigated. We compare the hybrid SA/TS algorithm to one TS and three SA algorithms. The final section contains some concluding remarks.

THE CCP PROBLEM

We are given a set of n customers, $O = \{1,\ldots,n\}$, with an $n \times n$ cost matrix, $[c_{kl}]$, indicating the distances between pairs of customers of O. We assume that $c_{kl} \geq 0$, $c_{kk} = 0$ and $c_{kl} = c_{lk}$ for all $k,l \in O$. For each customer k there is a positive demand d_k. Let $S = \{C_1, C_2, \ldots, C_p\}$ denote a partition of O into p clusters for a given integer p, $2 \leq p < n$, and $I = \{1,\ldots,p\}$ be the set of their indices. Let $\Phi = \{\zeta_1,\ldots,\zeta_p\}$ be the set of centres of S, where $\Phi \subseteq O$ and ζ_i is the centre (customer) from which the sum of the distances to all other customers in the cluster C_i is minimized. Let Q_{ζ_i} denote the capacity of cluster C_i and assume that $Q_{\zeta_i} = Q, \forall i \in I$. Note that the equality assumptions are not strictly necessary and they can be different without any loss of generality.

A generic feasible solution to the CCP can be represented as:

$$O = \bigcup_{i=1}^{p} C_i;\ C_i \cap C_j = \varnothing \forall i,j \in I \text{ and } i \neq j;\ \sum_{k \in C_i} d_k \leq Q_{\zeta_i} \forall i \in I;\ Z(S) = \sum_{i=1}^{p} \left(\sum_{k \in C_i} c_{k\zeta_i} \right), \qquad (1)$$

where $Z(S)$ is the objective function value of the solution $S = \{C_1,\ldots,C_p\}$.

An integer programming formulation of the uncapacitated clustered problem (UCP) is very easy to produce and can be found in Vinod (1969); Rao (1971); Mulvey and Crowder (1979). The UCP is a special case of the p-median problem and the uncapacitated facility location problems. Branch and

bound algorithms for it have been suggested by Jarvinen *et al.* (1972); Koontz *et al.* (1975); Christofides and Beasley (1982); Klein and Aronson (1991). There exists a number of heuristics for solving this problem (Teitz and Bart, 1968; Whitaker, 1983; Golden and Skiscim, 1986; Rahman and Smith, 1991).

Mulvey and Beck (1984) were the first to extend the UCP to the CCP by adding capacity restrictions on each cluster. They developed a primal heuristic and a hybrid heuristic-subgradient method to obtain good solutions for the CCP. The primal heuristic starts with a random set of p centres. It then assigns customers to their nearest centres in decreasing order of their regret values. A regret value is defined as the absolute value of the difference in distance between a customer's first and second centres. After all assignments are complete, the centre for each cluster is re-computed. If one or more new centres is found and the objective function decreases by a pre-specified minimum value, the process is repeated beginning with a re-assignment of customers to the newly computed set of centres. The procedure continues until no change in the centre set occurs from one iteration to the next. The heuristic-subgradient procedure is repeated for a predetermined number of iterations. At each iteration, the cluster centres are obtained by solving a relaxed problem that excludes the assignment constraints. A heuristic procedure then attempts to locate a feasible assignment of objects to centres. Improvements to the solution of both heuristics are made through the pairwise interchange of objects between clusters. Computational results on 18 randomly generated problems ranging in size from 10 to 100 customers show that both heuristics produce results with a relative percentage deviation of the solution value from the subgradient lower bound in the range of 0.43–5.4%.

There is a vast literature on problems that are mathematically related or very similar to the CCP including: the capacitated plant location problem with single source constraints (Barcelo and Casanovas, 1984; Klinewicz and Luss, 1986; Darby-Dowman and Lewis, 1988; Sridharan, 1993; Beasley, 1993); the fixed charge assigning users to sources problem (Neebe and Rao, 1983) and the generalized assignment problem (Martello and Toth, 1990). Clustering models are diverse with a wide areas of applications including: depot location in distribution systems (Klincewicz *et al.*, 1988; Bookbinder and Reece, 1988); sale force territorial design (Mulvey and Beck, 1984); location of switching centres in communication networks (Mirzaian, 1985); location of off-shore platforms for oil exploration (Hansen *et al.*, 1992); clustering of customers into different market segments in marketing studies (Chaffray and Lilien, 1973); information systems design (Karimi, 1986; Klein and Aronson, 1991). For a comprehensive review of clustering models and their applications, refer to Brandeau and Chiu (1989); Kaufman and Rousseeuw (1990).

SIMULATED ANNEALING AND TABU SEARCH

Simulated annealing was proposed independently by Kirkpatrick *et al.* (1983) and Cerny (1985). The SA algorithm is based on the analogy between the process of finding an optimal solution of a combinatorial optimization problem and the process of annealing of a solid to its minimum energy state in statistical physics (Metropolis *et al.*, 1956). Tabu search was proposed for solving combinatorial optimization problems by Glover (1986) and it is based on strategic oscillation ideas which are typically supported by memory structures to guide its non-montonic behaviour and to fulfil associated functions of controlling the search. Both SA and TS algorithms are meta-heuristics superimposed on the well-known iterative local search method. They have the ability to guide an iterative local search method to continue its search after a local optimum is detected using either a *deterministic* or *probabilistic* criterion to accept/reject newly generated solutions. Thus, we first briefly review the components of an iterative local search method for a combinatorial optimization problem.

A combinatorial optimization problem COP can be represented by a pair (Ω,Z) where Ω is the finite set of all possible solutions and an objective function Z which is a mapping $Z:\Omega \rightarrow \Re$ to be either minimized or maximized. The objective of a minimization problem is to find an optimal solution $S^* \in \Omega$ such that $Z(S^*) \leq Z(S)$ for all $S \in \Omega$ (Papadimitriou and Steiglitz, 1982). Let a neighbourhood generation mechanism be a mapping function, $N:\Omega \rightarrow 2^{\Omega}$, which defines for each solution $S \in \Omega$ the set of all possible solutions $S' \in \Omega$ that can be generated according to certain

operating rules. A transition from one solution S to another solution S' in $N(S)$ is called a *move* and $N(S)$ is called the neighbourhood of S. A local search algorithm starts with an initial solution and iteratively attempts to find better solutions by searching neighbourhoods. A solution S is called a *local minimum* with respect to the generation mechanism if and only if $Z(S) \leq Z(S') \forall S' \in N(S)$. The simplest type of local search method is a descent algorithm. Given an initial starting solution S and a neighbourhood generation mechanism N, a local search *descent* algorithm starts with a solution S and generates a new solution $S' \in N(S)$. If $\Delta < 0$ where $\Delta = Z(S') - Z(S)$, then S' replaces S in the next iteration; otherwise S is retained. The algorithm is repeated for a number of iterations until a local minimum is found. Unfortunately, the local minimum produced can be of a poor quality and its value may deviate significantly from that of the global minimum.

Simulated annealing and tabu search algorithms attempt to find *near-optimal* minima by occasionally accepting uphill moves, which increase the objective function values, employing probabilistic and deterministic acceptance strategies, respectively. In both SA and TS, downhill moves are generally accepted as in the iterative local search descent algorithm. More precisely, SA algorithm accepts an uphill move ($\Delta > 0$) from S to another $S' \in N(S)$ with probability $e^{-\frac{\Delta}{T}}$, where T is a positive control parameter called *temperature*. Neighbouring solutions are randomly selected as opposed to systematic selection in the descent algorithm. The acceptance probability decreases for increasing values of Δ and for decreasing values of T. The T values are updated according to a defined *cooling schedule*. A SA algorithm terminates when a pre-specified stopping criterion is met. Reviews of the theory and other successful applications of SA can be found in Van Laarhoven and Aarts (1987); Osman and Potts (1989); Connolly (1990; 1992); Eglese (1990); Osman (1991; 1993); Romeo and Sangiovanni-Vincentelli (1991); Van Laarhoven *et al.* (1992); Dowsland (1993); the bibliographies of Collins *et al.* (1988); Osman and Laporte (1994); and the contributions in Laporte and Osman (1994).

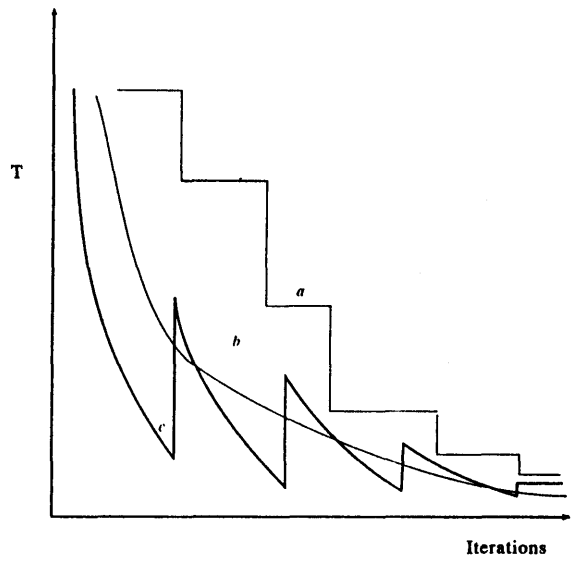
Tabu search was introduced as a memory structure to support and encourage its non-monotonic search. It aggressively explores and guides the search space looking for improvements using a deterministic rather than a probabilistic strategy. A TS algorithm chooses a neighbour $S' \in N(S)$ which produces the most improvement or the least non-improvement in the objective function value at each iteration. This acceptance criterion may lead to cycling, i.e. returning to solutions already visited. To avoid this, TS stores attribute values of the most recently visited solutions in a *tabu list*. Recorded attributes are often used to impose conditions, called *tabu restrictions*, that prevent moves from being chosen that reverse the change represented by these attributes. A move is called *tabu* if it leads to a solution whose attribute values are contained in the tabu list and satisfies tabu restrictions. Tabu moves are not allowed to be selected for a given number of iterations $|Ts|$ which is called a *tabu-list size*. After a number of $|Ts|$ iterations is performed, attributes on the tabu list are removed. *An aspiration criterion* is a test to correct wrongly diagnosed tabu moves. A move is called *admissible* (allowed to be selected) if it is not a tabu move or it passes an aspiration test. A *short term memory* is used to manage what goes in and out of the tabu list, to choose a selection strategy, to decide on admissible solutions, to override tabu restrictions by an aspiration criterion and a stopping criterion. For a general presentation of TS principles and successful applications, the reader may refer to Glover (1989; 1990); Glover and Laguna (1993); Hansen *et al.* (1992); Hasan and Osman (1994); Osman and Salhi (1993); Osman (1993; 1994): and the bibliography in Osman and Laporte (1994); and the contributions in Glover *et al.* (1993); and Laporte and Osman (1994).

Classification of the cooling schedules

The performance of the SA algorithm depends strongly on the chosen cooling schedule. With a good cooling schedule, near optimal solutions can be reached for many combinatorial problems. A great variety of theoretical models using the theory of Markov chains and practical cooling schedules have been suggested in the literature. Convergence results under different assumptions were proven independently by several authors (Gidas, 1983; Hajek, 1988; Lundy and Mees, 1986; Romeo and Sangiovanni-Vincentelli, 1991). Theoretical annealing schedules that have guaranteed convergence properties have been tested, but have been found generally to perform poorly in practical

applications. Thus, practical cooling schedules generally resort to heuristic criteria to obtain a good tradeoff between the running time and the quality of the solution. The interested reader should refer to the comprehensive bibliography by Collins *et al.* (1988).

A SA cooling schedule defines: (i) an initial value T_s of the control parameter; (ii) a decrement function for T; (iii) the number of iterations to be performed at each temperature; and (iv) a termination criterion. In the following, cooling schedules from the literature are classified into three categories. Each category includes both simple cooling schedules based on theoretical derivations and practical schedules which were found successful. We elaborate more on the cooling schedules with which we compare our results. A pictorial representation of the temperature pattern in each category is illustrated in Fig. 2.



a : Stepwise temperature reduction scheme.
b : Continuous temperature reduction scheme.
c : Non-monotonic temperature reduction scheme.

Fig. 2. Classification of cooling schedules according to the temperature changes after each iteration.

(a) *Stepwise temperature reduction schemes.* In this category, a SA cooling schedule specifies a finite sequence of homogeneous Markov chains of finite length, which are generated at monotonic decreasing values of the temperature. The length L_k of the kth Markov chain can be fixed or determined by the number of accepted moves (iterations) at the corresponding T_k value of the temperature. The simple cooling schedules of Kirkpartick *et al.* (1983); Johnson *et al.* (1989) and the theoretical schedules of White (1984); Aarts and Van Laarhoven (1985); and Huang *et al.* (1986) fall into this category. Each scheme has its own updating rule and stopping criterion. For instance, the cooling schedule parameters of Aarts and Van Laarhoven (1985) are determined as follows.

The *initial temperature* T_s is obtained by monitoring the evolution of the system during a number m of moves generated in the neighbourhood of the initial solution S_0 before the actual optimization process starts. Let m^+ be the number of cost increase in the m random moves, Δ^+ be the average cost increase over the moves, and χ is an acceptance ratio which is a real value, $0 < \chi < 1$. Then,

$$T_s = \Delta^+ \times \left(\ln \frac{m^+}{\chi \times m^+ - (1 - \chi) \times (m - m^+)} \right)^{-1} \tag{2}$$

or

$$T_s = \mu \times \Delta_{\max}, \tag{3}$$

where $\mu \gg 1$, e.g. $\mu = 10$ and $\Delta_{\max}$ is the maximum difference in the objective function values between any two solutions S and $S' \in \Omega$.

The *decrement rule* depends on a small constant decrement value δ and the standard deviation σ_k of the objective function values generated at the kth Markov chain in the following way:

$$T_{k+1} = T_k \times \left\{1 + \frac{T_k \times \ln(1+\delta)}{3 \times \sigma_k}\right\}^{-1}. \tag{4}$$

The length L_k of each Markov chain is set to a constant value which is equal to the size L of the largest neighbourhood, i.e.

$$L = \max_{S \in \Omega} |\mathrm{N}(S)|. \tag{5}$$

The *stopping criterion* is set to depend on the difference between the average of objective values and the optimal solution. The SA algorithm is terminated once $\bar{Z}^k(S) - Z(S^*)$ is small, where S^* is the optimal solution and $\bar{Z}^k(S)$ is the average objective value at the kth Markov chain. Aarts and Van Laarhoven (1985) estimate that if T_k is small, then for a small real number, ε, the algorithm can be terminated if the following test is satisfied

$$\frac{\bar{Z}^k(S) - Z(S^*)}{\bar{Z}^1(S_0)} \leq \varepsilon. \tag{6}$$

Huang *et al.* (1986) use different initial temperatures and decrement rules. For example, for a given integer r, to accept with a probability χ moves whose objective values are up to $r \times \sigma_1$ worse than the initial objective value $Z(S_0)$, T_s can be computed as follows

$$T_s = -\frac{r \times \sigma_1}{\ln(\chi)}. \tag{7}$$

(b) *Continuous temperature reduction schemes.* This category includes the class of cooling schedules which specifies a finite sequence of inhomogeneous Markov chains. All chains have equal length of one iteration. After each iteration k, the temperature T_k is reduced according to some rules. This class includes the cooling schedules of Hajek (1988) and Lundy and Mees (1986). We only elaborate on Lundy and Mees (1986) as it has some common features with the new scheme developed in this paper.

Lundy and Mees set T_s to a value so that $T_s \gg U$ where U is an upper bound on $\Delta_{\max}$. The temperature is reduced at each iteration by a smaller amount using a *constant decrement ratio* $\beta \ll 1/U$, i.e. T is updated according to the following rule:

$$T_{k+1} = \frac{T_k}{(1 + \beta \times T_k)}. \tag{8}$$

It can be shown that if a total number M of iterations is needed to terminate the search, then for given T_s,T_f values, β can be easily evaluated as:

$$\beta = \frac{T_s - T_f}{M \times T_s \times T_f}. \tag{9}$$

(c) *Non-monotonic temperature reduction schemes.* This category includes cooling schedules that reduce the temperature value after each attempted move (iteration) with occasional increases (or resets) in the temperature whenever a special *cycle* is found. A cycle is a completed search of the neighbourhood of the current solution. The philosophy of the temperature increase is based on the idea that there is no point in reducing the temperature any further whenever a cycle takes place without any accepted move. This scheme is based on our new cooling schedule which is introduced in the next section. The cooling schedule of Connolly (1990; 1992) also forms part of this scheme. Connolly's schedule uses the decrement rule in (8) and a constant decrement ratio in (9) until a specified number of consecutive uphill moves have been rejected. At this point, the temperature is reset to T_{found}, the value at which the current best solution was found, β is set to zero and the search completes the remaining iterations at the constant temperature T_{found}. Connolly's SA scheme also

uses a systematic search of the neighbourhood that has been shown to perform better than a SA scheme with random search on the quadratic assignment problem.

The new cooling schedule

This cooling schedule was developed by Osman and Christofides (1989) to handle some of the disadvantages inherent in the previous annealing schemes. Firstly, the neighbourhood is often *randomly* searched to generate neighbours and if there are only a few neighbours which give improved objective values then the previous classes would miss or may take a very long time to find them. Secondly, at high values of T, a large number of bad solutions is unnecessarily accepted. On the contrary, at low values of T, the probability of accepting worse solutions than the current one becomes very small. Moreover, if a good initial solution is available, this would be destroyed using a high value of T. Consequently, a large amount of computation time is wasted. Thus, a better scheme is preferred that minimizes the computation time spent at high and low T values, and spends more computation time in good regions using a systematic search of the neighbourhood. Finally, both cooling schemes of Aarts and Laarhoven, and of Lundy and Mees control the temperature sequences using constant decrement ratios δ and β. The solution quality and computation time of these schemes depend on the value of these ratios. It is possible to reduce these ratios, i.e. increase the total running time, without improving, but also deteriorating the solution quality. The reason is that the temperature trajectories of these cooling schedules change with any change in the constant decrement ratios. It is desirable to design a scheme that can regenerate the same temperature trajectory if an extra computer time is made available without deteriorating the solution quality. If a new temperature trajectory can generate the first portion of the previous search, then we can either find a better solution or retain the best solution obtained from the first part of the search. These features are properties of our new cooling schedule for which the generic parameters are explained next.

(i) *The initial and final temperature values.* The initial value of T_s is not very important. A small value is recommended if we start from a good heuristic solution. The initial T_s and final T_f values could be set arbitrarily to the maximum Δ_{max} and to the minimum Δ_{min} differences in the objective function values, respectively. T_f is only included to show the relationship with the Lundy and Mees scheme.

(ii) *The decrement rule.* After each iteration, k, the temperature is updated according to the sequence:

$$T_{k+1} = \frac{T_k}{(1 + \beta_k \times T_k)}, \tag{10}$$

using a parameter β_k which is updated as

$$\beta_k = \frac{T_s - T_f}{(\alpha + \gamma \times \sqrt{k}) \times T_s \times T_f}, \tag{11}$$

where α and γ are constants and determined experimentally in terms of problem characteristics. Note that if $\gamma = 0$ then β_k is a constant independent of k, and the rules in (10) and (11) become the same as in (8) and (9), respectively. Here, β_k decreases as k increases and the temperature is then decreased more slowly with k.

(iii) *The condition for occasional temperature increase.* If a cycle of search is made without accepting any change, the current temperature T_k needs to be reset to a higher value. The reset value should not be very high since what we want is only to escape from the current local minimum, but not to deviate from it too much as if we are starting from a totally new random solution. The temperature reset T_{reset} could then allow further moves to be accepted, and perhaps better solutions might be generated. Initially, T_{reset} is set to the value of T_s. If the condition for the reset is satisfied, we set $T_{reset} = T_{reset}/2$. If $T_{reset} \geq T_k$, we set $T_{k+1} = T_{reset}$; otherwise $T_{k+1} = T_{found}$. After this reset, the decrement and the occasional reset rules are used until the algorithm stops.

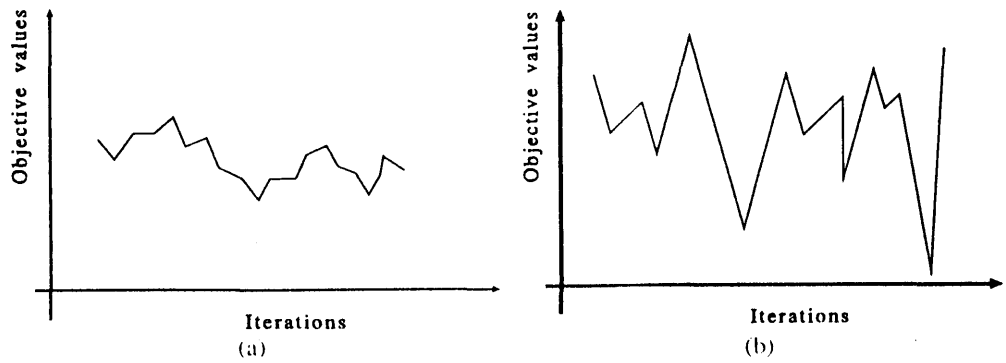


Fig. 3. Topological representation of objective function values.

(iv) *The stopping criterion.* Our SA algorithm is designed so that it gives the user control of the trade-off between the quality of solution and the computation time. Two stopping criteria could be used. Either the algorithm is executed for a pre-specified number of iterations M or terminated after a number of temperature resets R is performed without improving the best solution.

The new hybrid SA/TS procedure.

Step 1. Generate an initial random (or heuristic) solution S. Initialize the generic cooling schedule parameters.

Step 2. Choose *systematically* a solution $S' \in N(S)$ and compute the difference in the objective values, $\Delta = Z(S') - Z(S)$.

Step 3. **If** S' is better than S ($\Delta \leq 0$), or S' is worse than S ($\Delta > 0$), but accepted by the randomization process at the present temperature T, i.e. $\theta \leq e^{-\frac{\Delta}{T}}$ where $\theta \in [0,1]$ is a random number.

Then replace S by S' and if necessary update the best solution S_{best} and the best temperature T_{found}.

Else retain the current solution S, continue the process and go to Step 4.

Step 4. Update the temperature T according to the decrement rule and the reset condition described in (ii)–(iii) of the new cooling schedule.

Step 5. **If** the stopping criterion in (iv) is met, Stop, **Else** go to Step 2.

The above scheme with its non-monotonic schedule induces strategic oscillation in the search pattern. It does not use any random selection of neighbours, nor waste computation effort at high or low temperatures. It rather spreads the time of the search throughout using the decremental and incremental rules. It provides a generalization of Connolly's simulated annealing scheme which shares with us the *reset* criterion and the *systematic* search of the neighbourhood. However, the later resets the temperature only once to T_{found} as opposed to several times in our scheme. Connolly's scheme will work best on problems with an objective function of a *smooth* topology, i.e. the local optima are shallow (Fig. 3a). However, in problems of a *bumpy* topology (Fig. 3b), there are many deep local minima and this scheme may perform poorly. This is because T_{found} will be either a high or low temperature value. In the former case, the scheme would escape from the local minimum, but becomes an ineffective random search because it accepts too many solutions, while in the latter case the escape from the local minimum is more difficult.

Proposition 1. The hybrid SA/TS algorithm produces at iteration Q, a solution which is always better than or equal to that produced at any iteration $M < Q$, holding constant other cooling schedule parameters.

Holding constant the values of T_s, α and γ, while increasing the number of iterations from M to Q, would generate a search trajectory which consists of the previous trajectory up to M iterations extended with a new trajectory due to extra iterations from M to Q. Consequently, the best solution

would either be improved due to the extended search or at least remain the same. The importance of this property is that it provides a guarantee of no degradation in solutions, if extra computation time is available. However, this obviously most necessary guarantee is not provided by many other cooling schedules in the literature.

Linking the new hybrid algorithm to simulated annealing and tabu search.
The new hybrid SA/TS algorithm combines ideas and concepts from simulated annealing and tabu search. It differs considerably from most classical SA-implementations and contains important elements that go beyond the SA framework and theory. The only common factor with SA-implementations is the probabilistic acceptance criterion. Classical SA-implementations use a monotonic cooling schedule to control the acceptance criterion, a random neighbourhood search and a stopping criterion which depends on the final temperature value. On the contrary, the hybrid SA/TS algorithm uses a *non-monotonic* cooling schedule, a *systematic* neighbourhood search and a stopping criterion which depends on the number of temperature *resets*. These features are typical of tabu search implementations.

The design of our non-monotonic cooling schedule to induce an oscillation behaviour in the temperature is a strategic oscillation concept of tabu search, Glover (1986;1989), which we have specialized to operate specifically as a mechanism for controlling search through *variations in the objective function*. The manipulation of parameter thresholds to produce strategic oscillation in the tabu search framework is typically supported by memory structures to guide its non-monotonic behaviour and to fulfil associated functions of controlling the search. It should be noted that the general form of strategic oscillation is not limited to objective function control, but applies similarly to produce enriched search patterns by varying other instrumental parameters, such as assignment of customers to a centre, quantities of resources used and degrees of infeasibility. Our SA implementation combines the non-monotonic behaviour of tabu search with the simulated annealing acceptance criterion, without much use of memory except for keeping the best known solutions and the temperature reset values. This combination offers a sufficiently rich source of search trajectories to be relied upon as a primary guidance mechanism, with greatly reduced reliance on forms of memory customarily used in tabu search. Hence, our algorithm consists of a hybrid SA/TS algorithm or more specifically a hybrid 'simulated annealing/strategic oscillation' algorithm.

The hybrid SA/TS approach has been shown to yield improved performance over standard SA approaches on a number of hard combinatorial optimization problems: the vehicle routing problem (Osman, 1993; van Breedam, 1994); the generalized assignment problem (Osman, 1991; 1994) and the maximal planar graph (Hasan and Osman, 1994). The success of our approach suggests the possibility of similarly combining the non-monotonic oscillation approach with other forms of probabilistic guidance (as in probabilistic tabu search, for example) as an alternative to the accept/reject choice rules of simulated annealing and the deterministic memory structures of most tabu search implementations. Our approach has opened a new promising line of research in the design of hybrid algorithms which are discussed more in Glover (1993); Glover and Laguna (1993); Rayward-Smith (1994).

IMPLEMENTATION FOR THE CCP

From the previous section, we recall that in order to apply an iterative method to any combinatorial optimization problem, we need a precise definition of the solution representation, an objective function, an initial solution, a neighbourhood generation mechanism and a selection strategy of neighbours. The solution representation and its objective function are represented in (1). Hereinafter, we discuss the remaining other elements in more detail.

An initial solution

A constructive heuristic is proposed to obtain an initial feasible solution for the CCP. This heuristic consists of three iterative stages. The first stage is to choose an initial set of centres. The second stage assigns customers to the centres found in the first stage. The final stage re-computes the new centres according to the final assignment. The heuristic description is as follows.

FIND: Find a set of p centres.

Step 1. Find the largest $c_{k^*l^*}$ then, set $\zeta_1 = k^*$, $\zeta_2 = l^*$ and $\Phi = \{\zeta_1, \zeta_2\}$.
If $p = 2$ go to **ASSIGN**, **Else** set $i = 2$

Step 2. Set $i = i + 1$, find the next centre $\zeta_i \in O - \Phi$ so that the product of the distances from ζ_i to the previous centres is maximized in order to have a good spread of initial centres, i.e. identify $\zeta_i \in O - \Phi$ such that

$$\prod_{j \in \Phi} c_{ij} = \max_{j \in \Phi, l \in O - \Phi} \prod c_{lj}$$

Step 3. Set $\Phi = \Phi \cup \zeta_i$ and if $i = p$ go to **ASSIGN**, else go to Step 2.

ASSIGN: Assign customer points to centres.

Step 4. For each customer $k \notin \{\zeta_1, \ldots, \zeta_p\}$, find the distance to its nearest centre. Arrange the customers in increasing order of these distances. Assign customers in this ordered sequence to their corresponding centres, so long as the resource capacity allows. If not, assign the customer k to its immediately available nearest centre.

RE-COMPUTE: Re-compute cluster centres

Step 5. For each cluster C_i find the new centre ζ_i. In other words,

$$\text{identify } \zeta_i \in C_i \text{ such that, } Z(C_i) = \sum_{k \in C_i} c_{k\zeta_i} = \min_{k \in C_i} \sum_{l \in C_i} c_{kl}. \tag{12}$$

The above heuristic may fail to find a feasible solution when the capacities are very tight. In order to guarantee a feasible solution (if one exists) we would need to solve a multiple-knapsack (or bin-packing) problem exactly. This would be too time consuming to be considered for inclusion in our general heuristic. Alternatively, in stage 2, one could assign customers to their nearest centres in decreasing order of their regret values as in Mulvey and Beck (1984). Moreover, the last two stages can be recursively repeated to improve the solution further until no change in the set of centres is possible. The aim of this simple heuristic is, however, to obtain quickly an initial solution, but not to be used as a stand alone procedure.

A λ-interchange mechanism

The effectiveness of any iterative algorithm is partly determined by the efficiency of the generation mechanism and the way in which a neighbourhood is searched for a better solution. Lin (1965) suggested a generation mechanism called the λ-opt procedure based on *arcs*-exchange for the TSP. We introduce a similar λ-opt procedure based on *customers*-interchange for the CCP.

The neighbourhood generation mechanism for the CCP can be explained as follows. Given a solution $S = \{C_1, \ldots, C_i, \ldots, C_j, \ldots, C_p\}$ with ζ_i as the centre of cluster C_i, a λ-interchange between two given clusters C_i and C_j is a replacement of a subset $\bar{C}_i \subseteq C_i$ of size $|\bar{C}_i| \leq \lambda$ by another subset $\bar{C}_j \subseteq C_j$ of size $|\bar{C}_j| \leq \lambda$ to get two new clusters $C_i' = (C_i - \bar{C}_i) \cup \bar{C}_j$ and $C_j' = (C_j - \bar{C}_j) \cup C_i$ with possibly new different centres ζ_i' and ζ_j', respectively. The new solution becomes $S' = \{C_1, \ldots, C_i', \ldots, C_j', \ldots, C_p\}$. The neighbourhood $N(S)$ of S is the set of all solutions S' generated by the λ-interchange mechanism for a given integer λ and it is denoted by $N_\lambda(S)$.

The size of the neighbourhood is determined by λ and the number of pairs of clusters. There are $p(p-1)/2$ different pairs of clusters (C_i, C_j) to be examined for a given λ. An ordered search selects

systematically all possible pairs (C_i,C_j) of clusters according to a permutation π without repetition as indicated in (13).

$$(C_{\pi(1)},C_{\pi(2)}),\ldots,(C_{\pi(1)},C_{\pi(p)}),(C_{\pi(2)},C_{\pi(3)}),\ldots,(C_{\pi(p-1)},C_{\pi(p)}). \tag{13}$$

The permutation π has a fixed order $(\pi(i) = i,\forall i \in I)$ in the descent algorithm, but it may have different orders in the other proposed algorithms. Furthermore for a given pair (C_i,C_j), we must also define the search order for the customers to be exchanged. We consider the case of $\lambda = 1$. The 1-interchange mechanism uses two processes to generate neighbours which are illustrated in Fig. 4.

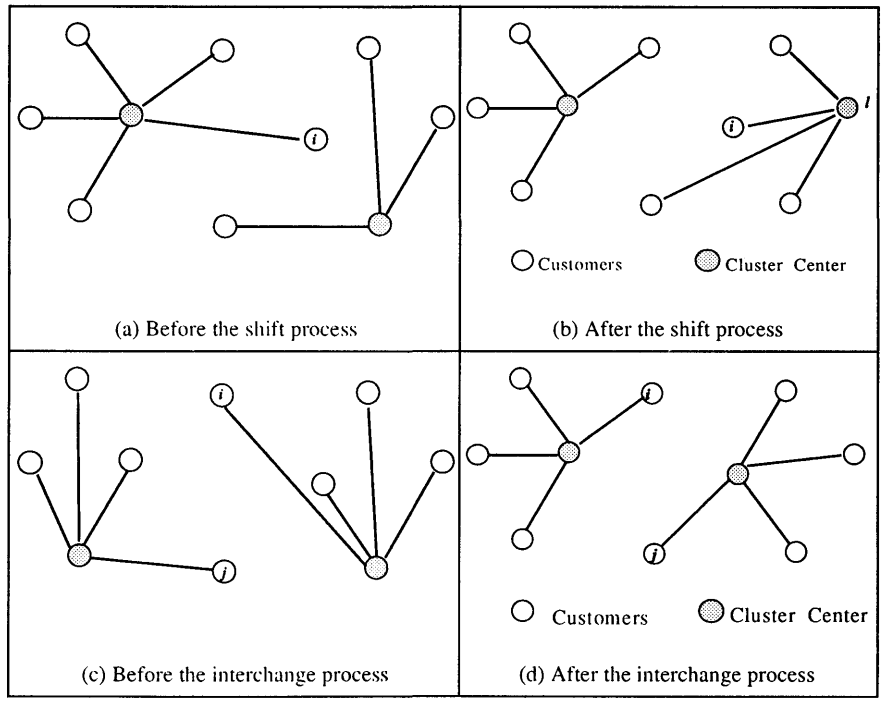


Fig. 4. The λ-interchange mechanism for the case of $(\lambda = 1)$.

(i) A *shift process* which is represented by the (0,1) and (1,0) operators. The (0,1) and (1,0) denote the shift of one customer from one cluster (say C_i) to another cluster (say C_j) or vice versa. Figure 4(a) and (b) illustrates a shift process in which a customer i is shifted by the (1,0) operator. As a result, a customer l becomes the new centre.

(ii) An *interchange* process which is represented by the (1,1) operator. This process attempts to exchange every customer from the first cluster with every other customer in the second. Figure 4(c) and (d) shows the change in the clusters after customers i and j are exchanged.

The customers in a given pair of clusters are searched sequentially and systematically for improved feasible solutions by the shift and interchange processes. The order of search, we implement, uses the following order of operators (0,1), (1,0) and (1,1) on any given cluster pairs to generate neighbouring solutions. For the case of $\lambda = 2$, this order could be (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1) and (2,2) of operators.

Definition. A solution S is λ-optimal (λ-opt) if and only if, for any pairs of clusters $C_i,C_j \in S$, there is no improvement that can be made by any λ-interchange move.

Proposition 2. The λ-optimal solutions produced by the descent methods are μ-optimal for any $\mu < \lambda$.

To produce an efficient λ-opt descent algorithm, it is advisable to obtain first a 1-opt solution. The 1-opt solution is then improved to find a 2-opt solution, and so on until the $(\lambda - 1)$-opt solution is finally improved to generate the desired λ-opt solution. The reason for this ordering is that we can

save a lot of computational time. For instance, few iterations are often needed to improve a 1-opt solution in order to find a 2-opt solution. Note, that a λ-opt solution is not unique and depends on the order of the search.

Selection strategies

In this paper, two selection strategies are used for choosing a neighbour $S' \in N_\lambda(S)$ when implementing an iterative improvement method. Firstly, the *best-improve* (BI) strategy examines all solutions $S' \in N_\lambda(S)$ and accepts the one which yields the best solution according to a given acceptance criterion. Secondly, an obvious alternative strategy is to accept the *first-improving* (FI) solution in the neighbourhood which satisfies the acceptance criterion. After such a move is made the search continues from the current position to the end of the cycle and back to the start of a new cycle until the current position is reached again. This is similar to searching a circular list. From our experience we found that the circular search is computationally more efficient than re-starting the search from the start of the cycle each time an improvement has been found.

COMPUTATIONAL EXPERIENCE

Test problems

Two sets of test problems with $n \in \{50,100\}$ and $p \in \{5,10\}$ are generated to compare the performance of the algorithms. One set contains 10 problems of size 50 × 5 and the other set contains 10 problems of size 100 × 10. Customers are located in the plane and their coordinates are randomly generated from a uniform distribution [1,100]. c_{kl} is the Euclidean distance between customers k and l. The demand value, d_k, is generated from the uniform distribution [1,20]. The cluster capacity, Q, for a given problem is chosen such that the tightness $\tau \in [0.82,0.96]$, where τ is a capacity ratio of the total demands over the total cluster capacities, i.e.

$$\tau = \left(\sum_{k \in O} d_k\right) \Big/ \left(\sum_{i \in I} Q_{\zeta_i}\right).$$

It is preferable to evaluate the proposed algorithms by comparison with an optimal solution. Unfortunately, there is no such known exact algorithm for the problem. Hence, optimal solutions are unknown for our test problems. We are aware of a current attempt to develop an exact algorithm by Hansen *et al.* (1993), but it is in an early stage. Consequently, the best known solution Z_{best} obtained by any algorithm is used as an estimate of the optimal value. The relative percentage deviation *RPD* of the heuristic solution Z_H from the best known solution is taken as a performance measure, i.e. $RPD = 100 \times (Z_H - Z_{best})/Z_{best}$. The average relative percentage deviation *ARPD* is taken over the 20 test problems. All algorithms are coded in FORTRAN 77 and run on a VAX 8600 computer. We report the average computation time ACT in CPU sec. of the actual execution time excluding input reading and output reporting time.

λ–Interchange descent algorithm

In this section, we study the quality of solution produced by the constructive heuristic (H.OC) and two local search procedures. The first local search procedure (H1 + FI) starts with the H.OC solution and uses the 1-interchange generation mechanism with the first-improve (FI) selection strategy, whereas the second procedure (H1 + BI) starts with the same H.OC solution and uses the 1-interchange generation mechanism, but with the best-improve (BI) selection strategy. The solution values and their relative percentage deviation RPD from the best known solutions Z_{best} produced by the above three algorithms are reported in Table 1 on the 20 test problems. The problems in each set are ordered in increasing order of their τ tightness values.

Table 1. Computational results for the constructive and 1-interchange descent algorithms

Set	Problem	τ	H.OC	RPD	H1 + FI	RPD	H1 + BI	RPD	Z_{best}
50 × 5	1	0.82	786	10.23	780	9.39	818	14.72	713
	2	0.84	816	10.27	762	2.97	778	5.13	740
	3	0.85	972	29.42	811	7.98	816	8.65	751
	4	0.86	891	36.86	651	0	652	0.15	651
	5	0.90	804	21.08	746	12.34	677	1.95	664
	6	0.92	882	13.36	841	8.09	847	8.86	778
	7	0.92	968	22.99	852	8.25	824	4.70	787
	8	0.92	945	15.24	834	1.70	837	2.07	820
	9	0.93	752	5.17	735	2.79	734	2.65	715
	10	0.96	1017	22.67	844	1.80	891	7.47	829
100 × 10	11	0.85	1761	75.04	1020	1.39	1019	1.29	1006
	12	0.85	1567	62.21	1004	3.93	974	0.82	966
	13	0.86	1847	80.01	1144	11.50	1053	2.63	1026
	14	0.88	1635	66.49	998	1.62	1054	7.33	982
	15	0.88	1517	38.91	1098	0.54	1138	4.21	1091
	16	0.88	1780	86.58	1063	11.42	993	4.08	954
	17	0.89	1665	61.02	1104	6.76	1092	5.60	1034
	18	0.89	1345	28.95	1089	4.41	1136	8.91	1043
	19	0.90	1634	58.48	1105	7.17	1125	9.11	1031
	20	0.94	1872	86.26	1036	3.08	1030	2.48	1005
ARPD				41.56		5.18		5.14	

τ: Tightness or capacity ratio.
H.OC: The constructive heuristic.
H1 + FI: H.OC followed by a local search procedure with 1-interchange mechanism and first-improve selection strategy.
H1 + BI: H.OC followed by a local search procedure with 1-interchange mechanism and best-improve selection strategy.
RPD: Relative percentage deviation from the best known solution Z_{best}.
ARPD: The average of RPDs over all 20 test problems.

In evaluating the results, we observe that the H.OC solutions deteriorate with increase of problem sizes and tightness values. For example, the average relative percentage deviation ARPD is 18.73% for the set of 50 × 5 problems, while it is 64.04% for the set of 100 × 10 problems. On the other hand, it is interesting to see the power of the 1-interchange mechanism and its robustness. The H.OC solutions are significantly improved by the H1 + FI and H1 + BI descent algorithms. The ARPD has dropped from 41.56% to 5.18% and 5.14%, respectively. The robustness is highlighted by the fact that the ARPD for the set of 100 × 10 problems is better than that for the set of 50 × 5 problems. Although we believe that the FI strategy is better than the BI strategy due its variable sampling to perform search intensification, they have shown a similar behaviour. We have investigated the effect of enlarging the neighbourhood size by increasing the value of λ from 1 to 2 on the performance of the H1 + FI. A 2-interchange descent procedure (H2 + FI) is used with a 2-interchange mechanism to improve the H1 + FI solution. As a result, the ARPD of H1 + FI solutions has been reduced from a value of 5.18% to a value of 3.93% for the H2 + FI solutions. This represents an improvement of 24% in the ARPD value with an increase in the ACT by a factor of 7.5 from 5.36 to 45.22 CPU sec.

Another experiment is conducted by replacing the constructive solution with a random initial solution to study the effect on the quality of solution produced by the λ-interchange descent algorithm. The previous experiment is repeated by replacing the initial H.OC solution in the H1 + FI and H2 + FI algorithms with a random initial solution to obtain similarity R1 + FI and R2 + FI algorithms. We report only the findings of this experiment rather than giving tabulated results. For each of the 20 test problems, the R1 + FI descent algorithm is run from five different random solutions. The ARPD of the R1 + FI solutions over all 100 runs is 4.25%, while the ARPD of the R2 + FI solutions is 3.23%. We can see that these ARPD values are comparable and even better than those produced by the H1 + FI and H2 + FI algorithms. Furthermore, if the ARPD is computed over the best 20 solutions which are the best solutions for each of the 20 test problems during the five different runs, we obtain an ARPD of 1.35% for the R1 + FI solutions and 1.16% for the R2 + FI solutions. The above experiments demonstrate that the λ-interchange mechanism is powerful and could improve quickly any random initial solution without the need of a better construction procedure. However, it might be important to use a good construction procedure when employing a weaker neighbourhood mechanism, rather than the one developed here. Despite the

improvement in the solution quality of the R2 + FI algorithm, these solutions are still unsatisfactory and a better method is required.

Comparisons of SA cooling schedules

We now compare three different SA algorithms which use different cooling schedules under the same conditions of the initial temperatures, starting solutions, neighbourhood search and 1-interchange generation mechanism. We can then fairly assess the effectiveness and merit of the corresponding cooling schedules. Initially, a cycle of search on the initial starting solution, S, is performed without accepting any neighbours to initialize the parameters of the cooling schedules: the largest Δ_{max} and the smallest Δ_{min} change in the objective function, the standard deviation σ and the total number of feasible solutions *Nfeas* in this cycle. The compared SA schemes are abbreviated and explained as follows.

HSS.AL starts from a constructive *heuristic* solution, selects *systematically* pairs of clusters in a fixed order of $\pi(\pi(i) = i \ \forall i \in I)$ in Eqn (14), searches *systematically* customers between clusters using the 1 − interchange mechanism, uses the Aarts and van Laarhoven (1985) cooling schedule with $\delta = 0.5, \varepsilon = 0.001$ and $L = |N_1(H)|$ following their suggestions and our initial experimentations.

HSS.OC uses the above HSS definition with the Osman and Christofides non-monotonic cooling schedule. The generic parameters are set as follows, $\alpha = p \times Nfeas, \gamma = n$ and $T_f = 1$. The total number of iterations M are set to $\alpha = n \times Nfeas$, so that the HSS.OC and HSS.AL perform approximately the same number of iterations and hence have a similar ACT time.

RSS.LM starts with a *random* solution, performs a *systematic* search of clusters and a *systematic* search of customers. The Lundy and Mees cooling schedule is used with M and $T_f = 1$ are set as in HSS.OC.

RSS.OC uses the same definition of RSS and the values of the generic parameters in HSS.OC.

RS̄S.OC uses the same settings in RSS.OC, but a new permutation π is generated at the end of each cycle and cluster selections are continued as in (13).

RRR.OC starts with a *Random* solution, selects clusters *randomly*, and generates a *random* 1-interchange move and the values of the generic parameters in HSS.OC.

RS̄S.$\overline{OC}$ uses the same definition of RS̄S, but the non-monotonic cooling schedule with $\gamma = 0$, i.e. it employs the same temperature reset and update rules but with a constant β rather than a variable β_k.

The above SA schemes are compared using different initial temperature values of T_s. In Table 2, T_s was set to take the values of: Δ_{max}; 2std and 3std which are computed using (7) with $\chi = 0.8$ and $r = 2$ and 3, respectively; Avg is obtained using (2) with $\chi = 0.8$; and $10 \times \Delta_{max}$.

For each test problem, the SA schemes with 'R' at the beginning are started from five different random solutions. However, for the SA schemes with 'H' at the beginning, the random number generator seed for the acceptance probabilities is changed so that a total of five different runs can be obtained. Each scheme performs 100 runs in total on the 20 test problems and the ARPD of all solutions is listed in Table 2. It can be seen from the results in Table 2, that our schemes *.OC perform better than that of Aarts and van Laarhoven, and Lundy and Mees under all the different settings. For example, the worst ARPD of HSS.OC is 0.448% which is twice as good as the best ARPD value of 0.866% produced by the HSS.AL. Similarly, the worst average of RSS.OC is better than the best average of RSS.LM. The superiority of our cooling schedule over its counterparts is due mainly to the non-monotonic oscillation control of the temperature values. Unlike the other SA schemes, the strategic oscillation makes our SA schemes less sensitive to the choice of the initial temperature.

We shall now discuss issues concerning the neighbourhood search. It can also be seen in Table 2, that the RS̄S.OC yields better results on average than that of RRR.OC. This is, perhaps because a random search may miss a neighbour having a better solution, whereas the ordered search gives a guarantee to find it within a complete cycle of search. The results of the RS̄S.$\overline{OC}$ algorithm are comparable to the RS̄S.OC results. Both of these SA algorithms gave the best relative percentage

Table 2. The average relative percentage deviations, ARPDs, of the simulated annealing schemes under different initial temperature values for all test problems

	Initial temperature values of T_s					All T_s	
SA Schemes	Δ_{max}	2std	3std	Avg	$10 \times \Delta_{max}$	AARPD	STD
HSS.AL	1.099	0.915	1.028	0.866	1.066	0.994	0.089
HSS.OC	0.448	0.278	0.361	0.436	0.363	0.377	0.061
RSS.LM	0.744	1.134	1.077	0.848	1.115	0.983	0.157
RSS.OC	0.405	0.571	0.405	0.494	0.294	0.435	0.092
R$\bar{S}$.SOC	0.391	0.495	0.330	0.394	0.271	0.376	0.074
RRR.OC	0.503	0.367	0.423	0.432	0.619	0.460	0.086
R$\bar{S}$S.$\overline{OC}$	0.256	0.221	0.356	0.310	0.372	0.303	0.057

HSS: Heuristic starts with five different seeds for each problem, systematic search of pairs of clusters, systematic search between clusters.
R$\bar{S}$S: Five random starts for each problem, systematic search with random permutations in (13), systematic search between clusters.
RRR: Five random starts for each problem, random selection of clusters, random search between clusters.
AL: Aarts and Van Laarhoven (1985) cooling schedule.
LM: Lundy and Mees (1986) cooling schedule.
OC: Our non-monotonic cooling schedule.
$\overline{OC}$: Our non-monotonic cooling schedule with $\gamma = 0$.
AARPD: Average of the ARPD over all T_s.
STD: Standard deviation of the ARPD over all T_s.

deviations of less than 0.40% for all the given initial temperatures. The R$\bar{S}$S.$\overline{OC}$ scheme demonstrates the importance of the temperature reset scheme without the need of dynamically varying the decrement ratio.

Finally, experimental results show the HSS.OC scheme which starts from a heuristic initial solution produces better solution quality than that of RSS.OC which uses a random initial solution. The average of the ARPDs over all initial temperatures and the corresponding standard deviation are 0.377 and 0.061 for HSS.OC compared to 0.435 and 0.092 for RSS.OC, yet the former scheme uses less computation time. Hence, starting SA from a good constructive solution is preferred as it reduces the computation time without deteriorating the solution quality. However, the R$\bar{S}$S.OC scheme which uses a random initial solution and a random permutation for selecting clusters in (13) produces better results than the RSS.OC scheme which uses a fixed permutation. This improvement is demonstrated by the reduction in the average of ARPD and its standard deviation of the R$\bar{S}$S.OC scheme to 0.376 and 0.074, respectively. These values are almost the same as those for the HSS.OC scheme. Yet, the HSS.OC scheme uses slightly less computation time as seen in Table 3, due to the absence of random generation of permutations. Since the schemes with random permutations produce clearly better results than those without at a slightly extra computation time, we would expect the H$\bar{S}$S.OC scheme to do better than both of the HSS.OC and R$\bar{S}$S.OC schemes. Hence, it will be chosen in our final comparison.

We now comment on the computation requirements of each scheme. From the different runs on each algorithm, we obtained the empirical probability, ρ, of finding the best known solution in one single run. We shall establish based on the empirical probability, ρ, the expected number of different runs, NR (and hence the total CPU) required to achieve the best known solution with a given certainty. The probability that an algorithm fails to produce the best known solution in NR runs is $(1 - \rho)^{NR}$. By setting this probability to a small value of $\psi\%$, then the best known solution can be found in NR independent runs with a certainty of $(100 - \psi)\%$ when $NR = \left[\frac{\ln(\psi\%)}{\ln(1-\rho)}\right]$, where $[\ \times\ ]$ denotes the smallest integer greater than $\times$. The total number of estimated runs with their corresponding total time T_{NR} for different algorithms with $\psi = 1$ are given in Table 3. This form of comparison was also used in Connolly (1989).

From Table 3, we observe that all our SA schemes have produced the best known solutions with the highest probabilities $\rho \in [0.482,0.504]$ in one run, compared to $\rho \in [0.278,0.365]$ for the other SA schemes. It is observed that in most cases the total CPU time required for HSS.OC, and all *.OC, is less than half that of HSS.AL, RSS.LM and R2 + FI. Our SA algorithms found the best known results in 20 out of the 20 test problems. The comparable figures for HSS.AL, HSS.LM and R2 + FI

Table 3. The expected number of runs, NR, and the total CPU time required to find the best known solution with 99% certainty ($\psi = 1$) for all test problems

Schemes	ρ	NR	ACT	Total CPU
HSS.AL	0.278	15	91	1365
HSS.OC	0.482	7	95	665
RSS.LM	0.365	11	101	1111
RSS.OC	0.494	7	101	707
RSS.OC	0.496	7	117	819
RSS.$\overline{OC}$	0.504	7	116	812
R1 + FI	0.100	∞	5	∞
R2 + FI	0.130	33	46	1518

ρ: Probability of finding the best known solution in a single run.
ACT: The average CPU time in sec. of a single run.
Others: As defined in Table 2.

were 15 out of 20, 16 out of 20 and seven out of 20, respectively. Although the R1 + FI algorithm takes the least ACT per problem amongst all algorithms, it may never find the best solutions since it failed to find the best known solution for any problems of size 100 × 10. It only found the best solutions for only seven out of 20 problems of small sizes 50 × 5. Consequently, its expected number of runs is set to infinity in Table 3.

Comparisons of the non-monotonic SA schemes and the TS algorithm

This section investigates the performance of two SA algorithms which use the non-monotonic cooling schedules with a TS implementation. Specifically, we compare our HSS.OC scheme, Connolly's SA algorithm which is denoted by HSS.C, and a TS implementation which is denoted by TS1 + FBI. The HSS.C scheme uses $T_s = \Delta_{max} + 0.1 \times (\Delta_{max} - \Delta_{min})$, and $T_f = \Delta_{min}$, while the initial starting solution, 1-interchange mechanism and the stopping criterion are the same as our HSS.OC scheme. The HSS.OC scheme uses $T_s = \Delta_{max}$ and the settings described earlier. However, another stopping criterion is used, which is based on the number of temperature resets R with $R = 3$. The reason for the use of this stopping criterion is to demonstrate that our scheme would guarantee improvement in the solution quality and gives the users the tradeoffs between the quality of solution and the running time. Both of the compared SA schemes use the same stopping criterion.

Let us now describe the implementation of TS1 + FBA algorithm. The TS1 + FBA algorithm starts from the heuristic H.OC solution and uses the 1-interchange mechanism with an FBA selection strategy. The FBA strategy selects the first admissible move that improves the current objective value or the least non-improving move if no such improving move exists. If a 1-interchange move exchanges $k \in C_i$ with $l \in C_j$, then these attributes are stored in a tabu list. Tabu restrictions then prevent the return of k to C_i and l to C_j, simultaneously. An aspiration criterion is used to allow a tabu move to be accepted if it improves the best solution. A variable tabu list size is used. Three values $\left[\frac{n}{4}\right]$ and $\pm$ 10% of $\left[\frac{n}{4}\right]$ are computed and a random permutation of them is generated. $|Ts|$ is assigned to one of the three values according to the order in the permutation and $|Ts|$ is changed every $2 \times |Ts|$ iterations. The TS1 + FBA algorithm is terminated after performing $5 \times n$ iterations without improving the best solution. Computational results from the above three schemes are reported in Table 4. More detail on the above TS1 + FBA implementation and other TS schemes can be found in Osman (1991).

We observe that the solution quality of the HSS.OC scheme is better than that of the HSS.C and TS1 + FBA algorithms. This is expected as HSS.OC extends HSS.C and employs some of the TS1 + FBA features. It must be noted that the poor performance of HSS.C can be explained by the fact that the scheme falls in a deep local minimum where T_{found} is too small in problems 1, 7 and 20. In this case, it is difficult to escape from the local optimality. It is interesting to note that if T_{found} is too high as in the case of problem 12, the HSS.C becomes a random search procedure. The best known

Table 4. Computational results of the HS̄S.OC, HS̄S.C and TS1 + FBA algorithms

Sets	No.	HS̄S.OC	RPD	CPU	HS̄S.C	RPD	CPU	TS1 + FBA	RPD	CPU	Z_{best}
50 × 5	1	713	0	12.74	734	2.94	12.77	734	2.94	8.00	713
	2	740	0	10.50	740	0	13.62	740	0	4.34	740
	3	751	0	13.49	751	0	22.89	751	0	11.65	751
	4	651	0	7.93	651	0	18.95	651	0	6.21	651
	5	664	0	9.13	664	0	16.07	664	0	26.86	664
	6	778	0	34.03	778	0	14.24	778	0	10.96	778
	7	787	0	11.33	805	2.28	26.76	787	0	10.72	787
	8	820	0	47.49	820	0	39.18	821	0.12	20.78	820
	9	715	0	52.09	715	0	25.21	715	0	21.25	715
	10	829	0	33.59	829	0	13.69	829	0	45.16	829
100 × 10	11	1006	0	650.85	1006	0	190.94	1009	0.29	313.24	1006
	12	966	0	631.11	966	0	1536.54	968	0.20	79.05	966
	13	1026	0	191.42	1026	0	223.51	1026	0	253.77	1026
	14	985	0.30	131.98	982	0	251.32	985	0.30	671.96	982
	15	1091	0	175.51	1091	0	426.06	1096	0.36	50.55	1091
	16	954	0	198.79	954	0	401.13	957	0.31	347.93	954
	17	1039	0.48	144.14	1037	0.29	269.48	1040	0.58	1022.0	1034
	18	1045	0.19	299.07	1045	0.19	156.52	1045	0.19	142.19	1043
	19	1031	0	474.51	1032	0.09	143.08	1034	0.29	246.56	1031
	20	1005	0	484.52	1019	1.39	134.39	1005	0	326.93	1005
Averages		—	0.04	180.24	—	0.36	196.82	—	0.28	181.11	—

HS̄S: Heuristic starts, systematic search with random permutations in (13), systematic search between clusters.
OC: Osman and Christofides' non-monotonic cooling schedule.
C: Connolly's non-monotonic cooling schedule.
TS1 + FBA: Tabu search with the first-best-admissible strategy.

solution for problem 12 was found, but it took too much time to terminate. Finally, the TS1 + FBA algorithm has failed to produce the best solution for problem 1. Since this problem has the smallest ratio τ of 0.82, it affords more feasible solutions. The failure of both approaches to this problem can be attributed to its topological structure and its solution distribution which are handled in our hybrid SA/TS approach.

CONCLUDING REMARKS

This paper introduces a simple constructive heuristic to generate quickly an initial solution for the capacitated clustering problem. A λ-interchange mechanism is also developed to generate neighbouring solutions and embedded in the local search methods. Simulated annealing cooling schedules are classified into three different categories: stepwise temperature reduction, continuous temperature reduction; and non-monotonic temperature reduction. Moreover, a new non-monotonic cooling schedule is developed that controls the temperature values using decremental and occasionally incremental rules in order to have a strategic oscillation behaviour in the objective function. The non-monotonic schedule has variable (rather than a static) reduction factors to provide smoother temperatures as the number of iterations increases. A hybrid simulated annealing and tabu search algorithm (SA/TS) is developed that uses the acceptance criterion of SA annealing which is controlled by the non-monotonic cooling schedule. The new cooling schedule, by design, is based on the oscillating behaviour of tabu search philosophy in the objective function. The linkage between the hybrid approach and both the simulated annealing and tabu search is discussed. Finally, a simple tabu search algorithm (TS1 + FBA) which uses a first-best-admissible selection criterion is implemented.

Extensive computational results on randomly generated test problems of sizes varying from 50 to 100 customers have demonstrated the following findings. The 1-interchange local search descent algorithm has improved significantly the solution quality of the constructive heuristic highlighting the power of the 1-interchange mechanism. The effect on solution quality of initial temperatures, different search strategies, for all the existing SA schemes are investigated and the superiority of the hybrid SA/TS algorithm over them has been shown. It was found that if there was a good initial solution and a systematic search of a good neighbourhood, there was only a slight reduction in the

computation time. The hybrid SA/TS algorithm is further compared with both Connolly's SA algorithm and the simple tabu search algorithm. It was found that the hybrid SA/TS algorithm performs better than both of them. Although Connolly's SA algorithm produces good results for problems of smooth topological structures, it has been out-performed on average by the simple tabu search algorithm.

Finally, we conclude that the hybrid SA/TS algorithm is flexible and problem independent, in that, no *a priori* knowledge is required. It has been used to solve successfully without changing its default settings other hard combinatorial optimization problems such as the vehicle routing, maximal planar graph and generalized assignment problems. It is an important property in that it gives the user control in balancing the trade-off between the computation time and the desired quality of solution. It is always guaranteed that there is no degradation in solutions for a bigger number of iterations. The success of our hybrid approach suggests the possibility of similarly combining the non-monotonic oscillation approach with other forms of probabilistic or thresholding guidance, as an alternative to the accept/reject choice rules of simulated annealing and the deterministic memory structure of most tabu search implementations. The topological structure of combinatorial optimization problems and its impact on the behaviour of both simulated annealing and tabu search algorithms deserve further consideration. Other strategies for the TS algorithm, such as a long-term memory function, a data structure to cut the running time, must be investigated to see whether the performance of the simple tabu search algorithm can be improved. The marriage between simulated annealing and tabu search has been found successful and there is considerable scope for much more hybridization involving other artificial intelligence and optimization techniques. The development, analysis, and parallel implementations of such hybrids will be needed and deserve more research.

Acknowledgements – This work was supported by the Hariri Foundation, Lebanon. Thanks are due to Professor Fred Glover for helpful discussions, Professor Rolfe Tomlinson, Dr Chris Potts and Dr David Connolly for their useful comments.

REFERENCES

Aarts, E. H. L. & Van Laarhoven, P. J. M. (1985). Statistical cooling: a general approach to combinatorial optimization problems. *Philips Journal of Research*, Vol. 40, pp. 193–226.

Barcelo, J. & Casanovas, J. (1984). A heuristic Lagrangean algorithm for the capacitated location problem. *European Journal of Operational Research*, Vol. 15, pp. 212–226.

Beasley, J. E. (1993). Lagrangean heuristics for location problems. *European Journal of Operational Research*, Vol. 65, pp. 383–399.

Bookbinder, J. H. & Reece, K. (1988). Vehicle routing considerations in distribution system design. *European Journal of Operational Research*, Vol. 37, pp. 204–213.

Brandeau, M. L. & Chiu, S. S. (1989). An overview of representative problem in location research. *Management Science*, Vol. 35, pp. 645–674.

Cerny, V. (1985). A thermodynamical approach to the travelling salesman problem: an efficient simulated annealing algorithm. *Journal of Optimization Theory and Applications*, Vol. 45, pp. 41–51.

Chaffray, J. & Lilien, G. (1973). A new approach to industrial market segmentation. *Sloan Management Review*, Vol. 18, pp. 41–54.

Christofides, N. & Beasley, J. E. (1982). A tree search for the p-median problem. *European Journal of Operational Research*, Vol. 10, pp. 196–204.

Collins, N. E., Eglese, R. W. & Golden, B. L. (1988). Simulated annealing – an annotated bibliography. *American Journal of Mathematical and Management Sciences*, Vol. 9, pp. 209–307.

Connolly, D. (1989). A comparison of solution techniques for the quadratic assignment problem and the development of general purpose simulated annealing algorithm. Ph.D. dissertation. U.K.: London School of Economics.

Connolly, D. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research*, Vol. 46, pp. 93–100.

Connolly, D. (1992). General purpose simulated annealing. *Journal of Operational Research Society*, Vol. 43, pp. 494–505.

CPLEX optimization, Inc., Suite 279, 930 Tahoe Blvd. Bldg. 802, Incline Village, NV 89451-9436, e-mail: info@cplex.com.

Darby-Dowman, K. & Lewis, H. S. (1988). Lagrangean relaxation and the single-source capacitated facility-location problem. *Journal of Operational Research Society*, Vol. 39, pp. 1035–1040.

Dowsland, K. A. (1993). Simulated annealing. In C. R. Reeves (Ed.) *Modern Heuristic Techniques for Combinatorial Problems*, pp. 20–69. Oxford: Blackwell Scientific.

Eglese, R. W. (1990). Simulated annealing: a tool for operational research. *European Journal of Operational Research*, Vol. 46, pp. 271–281.

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: a Guide to the Theory of NP – Completeness*. San Francisco: Freeman.

Gidas, B. (1983). Nonstationary Markov chains and convergence of the annealing algorithm. *Journals of Statistical Physics*, Vol. 39, pp. 73–131.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, Vol. 1, pp. 533–549.
Glover, F. (1989). Tabu search Part I. *ORSA Journal on Computing*, Vol. 1, pp. 190–206.
Glover, F. (1990). Tabu search Part II. *ORSA Journal on Computing*, Vol. 2, pp. 4–32.
Glover, F. (1993). *Tabu Thresholding: Improved Search by Non-monotonic Trajectories*. Graduate School of Business and Administration, University of Colorado at Boulder, U.S.A.
Glover, F. & Laguna, M. (1993). Tabu search. In C. R. Reeves (Ed.) *Modern Heuristic Techniques for Combinatorial Problems*, pp. 70–150. Oxford: Blackwell Scientific.
Glover, F., Taillard, E. & de Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research*, Vol. 41, pp. 3–28.
Golden, B. & Skiscim, C. (1986). Using simulated annealing to solve routing and location problems. *Naval Research Logistics Quarterly*, Vol. 33, pp. 261–279.
Hajek, B. (1988). Cooling schedule for optimal annealing. *Mathematics of Operations Research*, Vol. 13, pp. 311–329.
Hansen, P., Filho, E. L. P. & Riberio, C. C. (1992). Location and sizing of off-shore platforms for oil exploration. *European Journal of Operational Research*, Vol. 58, pp. 202–214.
Hansan, P., Jaumard, B. & Sanlaville, E. (1993). Private communication.
Hasan, M. & Osman, I. H. (1994). Local search strategies for the Maximal Planar Layout problem. *International Transactions in Operational Research*, Vol. 1.
Huang, M., Remeo, F. & Sangiovani-Vincentelli, A. (1986). An efficient general cooling schedule for simulated annealing. IEEE, *Proceedings of the International Conference on Computer Aided Design*, Santa Clara, pp. 381–384.
Jarvinen, P., Rajala, J. & Sinervo, H. (1972). A branch and bound algorithm for seeking the p-median. *Operations Research*, Vol. 20, pp. 173–178.
Johnson, S., Aragon, C., Mccgeoch, L. & Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation, Part I, graph partitioning. *Operations Research*, Vol. 37, pp. 865–892.
Karimi, J. (1986). An automated software design methodology using CAPO. *Journal of Management Information Systems*, Vol. 3, pp. 71–100.
Kaufman, L. & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley.
Kirkpatrick, S., Gelatt, C. D. & Vecchi, P. M. (1983). Optimization by simulated annealing. *Science*, Vol. 220, pp. 671–680.
Klein, K. & Aronson, J. E. (1991). Optimal clustering: a model and method. *Naval Research Logistics*, Vol. 38, pp. 447–461.
Klincewicz, J. G. & Luss, H. (1986). A Lagrangean relaxation heuristic for the capacitated facility location with single constraints. *Journal of Operational Research Society*, Vol. 37, pp. 495–500.
Klincewicz, J. G., Luss, H. & Yu, C. S. (1988). A large multilocation capacity planning model. *European Journal of Operational Research*, Vol. 34, pp. 178–190.
Koontz, W. L. G., Patrenahall, M. N. & Fukunaga, K. (1975). A branch and bound clustering algorithm. *IIE Transactions on Computers*, Vol. 24, pp. 908–915.
Laporte, G. & Osman, I. H. (1994) (Eds). *Metaheuristics In Combinatorial Optimization*. Annals of Operations Research. Switzerland: Baltzer AG, in press.
Lin, S. (1965). Computer solutions of the travelling salesman problem. *Bell System Technical Journal*, Vol. 44, pp. 2245–2269.
Lin, S. & Kernighan, B. (1973). An efficient heuristic for the travelling salesman problem. *Operations Research*, Vol. 21, pp. 498–516.
Lundy, S. & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical programming*, Vol. 34, pp. 111–124.
Martello, S. & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. New York: Wiley.
Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, M. & Teller, E. (1956). Equation of steady state calculation by fast computing machines. *Journal of Chemical Physics*, Vol. 21, pp. 1087–1092.
Mirzaian, A. (1985). Lagrangean relaxation for the start-star concentrator location problem: approximation algorithm and bounds. *Networks*, Vol. 15, pp. 1–20.
Mulvey, J. M. & Crowder, H. P. (1979). Cluster analysis: an application of Lagrangean relaxation. *Management Science*, Vol. 24, pp. 329–340.
Mulvey, J. M. & Beck, M. P. (1984). Solving capacitated clustering problems. *European Journal of Operational Research*, Vol. 18, pp. 339–348.
Neebe, A. W. & Rao, M. R. (1983). An algorithm for the fixed charge assigning users to sources problem. *Journal of Operational Research Society*, Vol. 34, pp. 1107–1113.
Osman, I. H. & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, Vol. 17, pp. 551–557.
Osman, I. H. & Christofides, N. (1989). Simulated annealing and descent algorithms for capacitated clustering problem. (Research Report, Imperial College, University of London. Presented at EURO-X conference Beograd, Yugoslavia, 1989.)
Osman, I. H. (1991). Metastrategies, Simulated annealing and Tabu Search for combinatorial optimization problems. Ph.D. dissertation, The Management School, Imperial College, University of London, U.K.
Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithm for the vehicle routing problem. *Annals of Operations Research*, Vol. 41, pp. 421–451.
Osman, I. H. & Salhi, S. (1993). Local search strategies for the Mix Fleet Vehicle Routing Problem. Report no. UKC/IMS/OR94/2b, University of Kent, U.K. Presented at the ORS35 conference, York, 1993.
Osman, I. H. (1994). Heuristics for the generalised assignment problem: simulated annealing and tabu search. (Report no. UKC/IMS/OR93/10b, University of Kent, Canterbury, Forthcoming in OR Spketrum.)
Osman, I. H. & Laporte, G. (1994). Modern heuristics for combinatorial optimization problems: an annotated bibliography. In I. H. Osman & G. Laporte (Eds) *Metaheuristic in Combinatorial Optimization. Annals of Operations Research* (in press).
Papadimitriou, C. H. & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs: Prentice-Hall.
Rahman, S. & Smith, D. K. (1991). A comparison of two methods for the p-median problem with and without maximum distance constraints. *International Journal of Operations and Production Management*, Vol. 11, pp. 76–84.
Rao, M. R. (1971). Clustering analysis and mathematical programming. *Journal of the American Statistical Association*, Vol. 66, pp. 622–626.
Rayward-Smith, V. J. (1994). A unified approach to tabu search, simulated annealing and genetic algorithms. In J. Taylor, A.

Gammerman & V. Rayward-Smith (Eds) *Adaptive Computing and Information Processing*. Brunel: UNICOM.
Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell Scientific.
Romeo, F. & Sangiovanni-Vincentelli, A. (1991). A theoretical framework for simulated annealing. *Algorithmica*, Vol. 6, pp. 302–345.
Sridharan, R. (1993). A Lagrangean heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research*, Vol. 66, pp. 305–312.
Teitz, M. & Bart, P. (1968). Heuristic methods for estimating the generalised vertex median of a weighted graph. *Operations Research*, Vol. 16, pp. 955–961.
Van Breedam, A. (1994). Improvement heuristics for the vehicle routing problem based on simulated annealing. Working paper 93/13, University of Antwerp, RUCA, Belgium. *European Journal of Operational Research* (in press).
Van Laarhoven, P. J. M. & Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications*. Dordrecht: D. Reidel.
Van Laarhoven, P. J. M., Aarts, E. H. L. & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, Vol. 40, pp. 113–125.
Vinod, H. D. (1969). Integer programming and the theory of groups. *Journal of the American Statistical Association*, Vol. 64, pp. 506–519.
Whitaker, R. A. (1983). A fast algorithm for the greedy interchange for large scale clustering and median location problems. *Canadian Journal of Operational Research and Information Processing*, Vol. 21, pp. 95–108.
White, S. R. (1984). Concept of scale in simulated annealing. *IEEE Proceedings of the International Conference on Computer Design*, pp. 646–651. Portchester.