Yugoslav Journal of Operations Research
6 (1996), Number 1, 41-54

THE CHAIN-INTERCHANGE HEURISTIC METHOD[1]

Nenad MLADENOVIĆ
GERAD and Ecole des Hautes Etudes Commerciales
University of Montreal, Montreal, Canada, H3T1V6

José A. MORENO and J. Marcos MORENO-VEGA
Departmento de Estadística, I.O. y Computación
Universidad de La Laguna, Spain

Abstract: This paper describes a new Tabu search-like technique for solving combinatorial optimization problems. The *chain-interchange* move is introduced., which is an extension of the well-known 1-interchange move. Instead of interchanging one solution attribute that is in the solution with one that is not, four attributes are interchanged. In that way the Tabu search recency-based memory is easily obtained, i.e., the possibility of getting out of the local optimum trap can be achieved without additional efforts. Some location-allocation problems that could be solved by the same *chain-interchange* algorithm by only changing the objective function are listed. Moreover, most of the problems listed are suggested for the first time to be solved by Tabu search method (TS). Computer results are reported.

Keywords: Heuristics, location, Tabu search, chain interchange.

1. INTRODUCTION

A new local search heuristic **Chain-interchange** method for solving combinatorial optimization problems is proposed. It could be seen as a Tabu search method adapted for a large class of problems, as well as an extended interchange heuristic.

TS was introduced by Glover (1986) and independently by Hansen (1986) (under the name *Steepest Ascent Mildest Descent*) as a meta-heuristic designed to get a global optimum to combinatorial optimization problems. TS belongs to a class of

[1] Work on this paper was initialized during a visit of the first author to La Laguna University.

Local Search Heuristic Methods, or more specifically to a class of *Descent-Ascent* Methods (Hansen and Jaumard (1990)).The major components include a *short-term memory* process which is the core component of the search procedure, an *intermediate memory* process for regionally intensifying the search, and a *long -term memory* process for globally diversifying the search. The short-term memory process is implemented through a set of *tabu conditions* and the associated *aspiration criteria*. The idea is to prevent cycling in the descent-ascent process by avoiding reversal or repetition of previously visited solutions. Intermediate and long-term memory processes are performed by restricting the search in a special subregion and by inducing the search in a new subregion of the solution space. TS has already been applied to an increasingly wide spectrum of problems. For a survey of a variety of successful applications, see Glover (1980, 1990) and Glover and Laguna(1994).

For each particular problem researchers developed neighbourhood structures, short-term, intermediate and long-term memory processes in order to build a few TS rules in existing local search heuristic methods. We have the opposite purpose in this paper: *find a class of optimization problems that could be solved using the same TS memory processes* , i.e., the same data structure. Our work is based on the fact that there are many more optimization problems that could be solved by TS than data structures developed to solve them in an efficient way. Thus our method applies to a large class of combinatorial problems, and for each problem just the subroutine evaluating the objective function value has to be changed.

The method proposed in this paper is an extended Interchange heuristic. Instead of interchanging two variables (attributes) in order to generate the neighbourhood of a current solution, we interchange the positions of four variables. This simple extension of well known descent local search 1-*interchange* methods allows us to obtain the following properties: (i) move at each iteration to the best neighbouring solution , even if it is not better than the current solution (to avoid a local optimum trap); (ii) introduce two waiting (tabu) lists (to prevent cycling). Lists consist of variables waiting to go out of the solution and to go into the solution respectively. Hence, we save time and memory by storing not the tabu solutions themselves, but one of their attributes; (iii) bring into the solution that variable which has been out of the solution for the longest time (to induce the search in a new region of the solution space.). Problem-specific knowledge now can be easily embedded in the algorithm, or, any existing descent heuristic method of the *Interchange* type can be extended, using the same data structure.

We shall illustrate our method on some discrete location-allocation problems. The paper is organized as follows. In the next section we formulate the p-facility location-allocation problem and give a list of some models that could be solved with our method by only changing the objective function. Our list of applications is meant to be illustrative, not exhaustive. In Section 3 we explain the rules of our *Chain-Interchange* (CI) algorithm within the Tabu search (TS) framework. Computer experience is reported in Section 4. Section 5 concludes the paper.

2. THE p-FACILITY LOCATION-ALLOCATION PROBLEM

The Location-Allocation (LA) problem consist of finding the best selection of

points to open facilities at them and the way to serve the users. The objective cost function to be minimized depends on the relative situation of the facility points and the demand points. In a wide class of relevant location-allocation problems, every alternative solution consists of a fixed number, say p, of facility points to be opened and then the optimal way to serve the users. For every set of p facility points, the optimal allocation of users to the facility points gives the optimal cost. Then the problem consists of finding the set of p facility points that minimizes the corresponding cost. These are the p-facility location-allocation problems where the feasible solutions are the sets of p facility points and the objective function ranges from simplest to those computed by solving an allocation problem (see Brandeau and Chiu (1989)).

In order to formalise discrete LA models, we consider set L of m potential location points, set D of n demand points and $m \times n$ matrix with the costs $c(x,u)$ of attending a user at u from a facility at x, $\forall x \in L$, $\forall u \in D$. Every alternative solution is given by the set X of location points chosen for the facilities and the allocation of every demand point $u \in U$ to the facility point $A(u) \in X$ that serves every user at u. The objective function to be minimized depends on the locations X and the allocations $A(.)$.

For directly-allocated models the objective cost function is monotone and every user at u is allocated to the facility point $A(u)$ such that $c(A(u),u) \leq c(x,u)$, $\forall x \in X$. Then the solution space for these problems is $S = \{S = X / X \subseteq L\}$. On the other hand, for any LA problem, given the allocation $A(.)$ of the users, the location of the facilities is given by $X = A(D)$. So the solution space can be considered to be $S = \{S = A / A : D \to L\}$. Another way to give the allocations consists of providing the partition of the demand point set in the points allocated to every facility point, i.e., by giving $U(x) = \{u \in D : A(u) = x\}$, $\forall x \in X$.

However, in order for describe the search procedure for these problems, it is convenient to use all the items related with the solution; i.e., the locations given by X, the allocations given by $A(.)$ and the partition given by $U(.)$. In any implementation we can use any of them to perform any step since $X = A(D)$ and $U(x) = A^{-1}(x)$, $\forall x$. Therefore, we can use the solution space:

$$S = \{S = (X, A, U) / X \subseteq L,\ A : D \to X,\ U : X \to 2^D \text{ with } A(u) = x \text{ iff } u \in U(x)\}$$

The p-facility problem is a directly-allocated LA problem with solution space:

$$S = \{S = X / X \subseteq L : |X| = p\}$$

Then the objective function characterizes the problem. A list of these problems is given in Brandeau and Chiu (1989). We selected 10 of them. This list is meant to be illustrative, not exhaustive. All of them are directly allocated LA problems, i.e., the objective cost function depends on the best allocation for every user. Some of these problems are announced as maximization problems in the literature, but they are all formulated here by objective functions to be minimised.

Let

$$c(X,u) = \min_{x \in X} c(x,u).$$

1. **The *p*-median problem**. The objective function is the average of the allocation costs.

$$f_1(X) = \frac{1}{|U|} \sum_{u \in U} c(x,u).$$

2. **The *p*-center problem**. The objective function is the maximum of the allocation costs.

$$f_2(X) = \max_{u \in U} c(X,u).$$

3. **The balanced *p*-centdian problem**. The objective function is the sum of the objective functions of the *p*-median and the *p*-center problems.

$$f_3(X) = f_1(X) + f_2(X).$$

4. **The median *p*-center problem**. The median *p*-center problem consists of finding the optimal solutions of the *p*-center problem with minimum average allocation costs. The objective function is also a linear combination of the objective functions of the *p*-median and *p*-center problems. It can be defined by

$$f_4(X) = f_1(X) + \alpha f_2(X).$$

where $\alpha \geq \max f_1(X)$. For instance $\alpha = \max_{x \in L, u \in U} c(x,u)$.

5. **The *p*-capture problem**. This problem consists of maximizing the amount of demand captured by p facility points. We assume that the amount of demand of every user captured by competitors is inversely proportional to the corresponding cost. The objective function of the *p*-capture problem is:

$$f_5(X) = \sum_{u \in U} \frac{c(Y,u)}{c(Y,u) + c(X,u)},$$

where Y represents locations the competitors already estabilished

6. **The p-covering problem.** The optimal solutions of the *p*-covering problem maximize the amount of users within a distance r. The objective function is:

$$f_6(X) = \left| \{ u \in U : c(X,u) > r \} \right|.$$

7. **The p-equity problem.** The optimal solutions are those minimizing the difference between the objective functions of the p-center and p-median problems. The objective functions is:

$$f_7(X) = f_2(X) - f_1(X).$$

8. **The *p*-anticenter problem**. The optimal solutions are those maximizing the minimum of the allocation costs. The objective function is:

$$f_8(X) = \max_{u \in U} -c(X,u)$$

9. **The *p*-mean problem**. The objective function is the average of squared allocation costs.

$$f_9(X) = \frac{1}{|U|} \sum_{u \in U} c^2(X,u).$$

10. **The *p*-log-median problem**. The objective function is the average of logarithm of the allocation costs.

$$f_{10}(X) = \frac{1}{|U|} \sum_{u \in U} \log c(X,u).$$

3. CHAIN-INTERCHANGE HEURISTIC METHOD

Most of the classical heuristic search procedures appearing in the literature to be applied to location-allocation problems can be described using moves (or neighbourhood structures). See Cooper (1964), Handler and Mirchandani (1979), Jacobsen (1983), Kuehn and Hamburger (1963), Maranzana (1964), Moreno et al. (1991), Teitz and Bart (1968) etc. This is formalised below.

3.1. Search moves for LA problems

Let (S,f) be the solution space and the objective function of a combinatorial problem. A heuristic search procedure consists of a strategy to apply a series of search moves. The search moves are operations to obtain a solution $S \in S$ starting from another solution $T \in S$.

For a location-allocation problem, let $S = (X, A, U)$ be the current solution given by the locations X, the allocations A and the partition U. The usual elementary moves for location-allocation problems are the following:

- **Reallocation** of demand point u to facility point x.
  - Take $x \in X$ and $u \in U$ with $U(A(u)) \neq \{u\}$.
  - Do $A(u) \leftarrow x$.

- **Opening** of a facility point at x to serve demand point u.
  - Take $x \notin X$ and $u \in D$ with $U(A(u)) \neq \{u\}$.
  - Do $A(u) \leftarrow x$.

- **Closing** of the facility center attending demand point u.
  – Take $x \in X$ and $u \notin U(x)$ with $U(A(u)) = \{u\}$.
  – Do $A(u) \leftarrow x$.

- **Translation** of the facility point at x to y.
  – Take $x \in X$ and $y \notin X$.
  – Do $A(u) \leftarrow y$ for $\forall u \in U(x)$.

The elementary moves only modify one item of the solution. Reallocation, Opening and Closing moves only modify the allocation of a demand point, so the closed and opened facility center corresponds to singleton demand sets. A translation move only modifies the location of a facility point.

The set of feasible solutions of a directly-allocated location-allocation problem is $S = \{S = X / X \subseteq L\}$. Let X be the current solution, then the usual elementary moves for these problems are:

- **Addition.**
  – Take $x \notin X$ and do $X \leftarrow X \cup \{x\}$.

- **Elimination.**
  – Take $x \in X$ and do $X \leftarrow X \setminus \{x\}$.

Finally, the set of feasible solutions of a p-facility location-allocation problem is $S = \{S = X / X \subseteq L : |X| = p\}$. Then the usual elementary move for these problems is:

- **Subtitution.**
  – Take $x \in X$ and $y \notin X$ and do $X \leftarrow (X \setminus \{x\}) \cup \{y\}$.

Given a set of (elementary) moves, solution T is a neighbour of solution S if there is a move that produces T from S. The neighbourhood $N(S)$ of solution S is the set of neighbours of S. The neighbourhood structure is very important for the success of a search procedure. From elementary moves, one can define multiple or combined moves.

3.2. Chain-interchange moves

A common property of *Reallocation, Opening* and *Closing* on the one hand and *Substitution* moves on the other is that they all interchange positions of some entities (variables) that belong to the solution with some that are not in the solution. If the solution is represented by pairs $(x, u), x \in L, u \in D$ (or by edges of an associated graph), then reallocation, opening and closing moves mean that one edge is moved out of the solution and another one goes in it. On the other hand, if the solution is represented as a subset of $L (X \subseteq L$ then an interchange move is a substitution move: one facility (vertex of associated graph) goes in X, another goes out.

We shall now introduce a new *chain-interchange* move. This move could be obtained as an extension of reallocation, opening, closing and substitution moves: if elements that should be inter-changed from one step to another are edges (x,u) of associated graph, we have *chain-reallocation*, *chain-opening* or *chain-closing* moves; if such elements are facilities $x \in X$, then a *chain-substitution* move is obtained. We shall describe in detail only the chain-substitution move and give a pseudo-code for it in the next subsection. Analogous results hold for chain-reallocation, chain-opening and chain-closing.

Let us divide a set of facility points L into four disjointed subsets X_1, T_1, X_2 and T_2, such that:

$$X = X_1 \cup T_1,\ X_1 \cap T_1 = \varnothing,\ |X| = p$$
$$L \setminus X = X_2 \cup T_2,\ X_2 \cap T_2 = \varnothing,\ |L \setminus X| = m - p,$$

with $|X_1| = p - t_1, |T_1| = t_1, |X_2| = m - p - t_2$ and $|T_2| = t_2$. Sets T_1 and T_2 are two tabu lists with cardinalities t_1 and t_2 respectively. Then the chain-substitution move is as follows:

- **Chain-substitution move.**
  Take $x_{out} \in X_1$, $x_{in} \in X_2$, $x' \in T_1$ and $x'' \notin T_2$. Do the following:
  $X_1 \leftarrow (X_1 \setminus \{ x_{out} \}) \cup \{x'\}$;
  $T_1 \leftarrow (T_1 \setminus \{x'\}) \cup \{ x_{in} \}$;
  $X_2 \leftarrow (X_2 \setminus \{ x_{in} \}) \cup \{x''\}$;
  $T_2 \leftarrow (T_2 \setminus \{x''\}) \cup \{ x_{out} \}$.

As the result of the above four moves we obtain one substitution move, i.e., we take $x_{out} \in X$ and $x_{in} \notin X$ and get $X \leftarrow (X \setminus \{ x_{out} \}) \cup \{ x_{in} \}$. The question is why do we need four instead of only one substitution move ? By taking out facility point x_{out} from $X_1 \subseteq X$, not from X, we force some facility points to be part of the solution for a certain number of steps, even if excluding them from the solution would produce a better solution. By implementing T_1 and T_2 as order lists and if $x' \in T_1$ is chosen in FIFO (First In First Out) manner, then each facility should stay in the solution for $t_1 = | T_1 |$ iterations. In that way at most t_1+1 ascent moves are allowed in the search process (if the current solution is a local minimum). The same holds for facility point x'', i.e., before it goes out from T_2, it is forced to be out of the solution for t_2 iterations. Let us emphasize this fact in the following Property.

Property 1 *Let the lengths of tabu lists T_1 and T_2 be t_1 and t_2 respectively. The sequence of solutions X_i, $i=1,2,...$ obtained by successive implementation of the chain-substitution moves above under the* FIFO *rule in T_1 and T_2, has a period of length $\alpha \geq t_1 + t_2 + 2$.*

P r o o f. The sequence X_i, $i=1,2,...$ could 'get into a loop' (a cycle of numbers that is repeated endlessly) only if the same solution X is obtained after α steps. We shall show that $\alpha \geq t_1 + t_2 + 2$. Indeed, he minimum number of iterations facility $x_{out}^{(i)} \in X_1$ needs in order to belong to X_1 again is $t_1 + t_2 + 2$: one step to go out from X_1,t_2 ite-

rations being in T_2 because of the FIFO rule, at least one step in X_2 and t_1 steps in T_1. It is easy to see that all other facilities that leave the solution after $x_{out}^{(i)}$ could be in T_1, thus, in the solution. ■

To change the positions of two elements with indices *out* and *in* in some ordered set $x(.)$, it is necessary to perform 3 operations: $xx := x(out)$; $x(out) := x(in)$; $x(in) := xx$. Note that for four changes we need only 5 operations (k and l are indices of the first entities introduced in tabu lists T_1 and T_2 respectively):

Chain-substitution-move (out, k, in, l, x)
$xx := x(out)$;
$x(out) := x(k)$;
$x(k) := x(in)$;
$x(in) := x(l)$;
$x(l) := xx$.

Note also that two important TS steps are performed with only these five statements: (i) move to a new current solution; (ii) update tabu list.

3.3. Algorithm

Let the number of demand (fixed) points n, the number of facility points m and the number of new facilities p be given. Let us assume that we are able to calculate the objective function value $f(X)$ if solution $X \subseteq L$ is known. These are all the input data we need. Output values of the *chain-substitution* procedure are: best found solution $X_{opt} = \{ x_{opt}(j), j = 1, \ldots, p \}$ and $f_{opt} = f(X_{opt})$.

Chain-substitution $(m, n, p, f_{opt}, x_{opt}(.))$.

Step 1. Parameters of the method
- t_1, t_2: lengths of the tabu lists T_1 and T_2;
- *nbmax*: the maximum number of iterations between 2 improvements;
- *nlong*: number of times a long term memory process will be used (i.e., how many times the procedure will be restarted in an unexplored area of solution space).

Step 2. Initialization
Let $x(j), j = 1, \ldots, m$ be the set of facility location points L in some order,
The first p of elements of $x(j), j = 1, \ldots, m$ represents the solution.
- Find an initial ordering of array $x(j), j = 1, \ldots, m$.
  { Obtained at random , (i.e., a random permutation of $1 \ldots m$)
  or by any heuristic method (for example by a greedy heuristic).}
- Put $x_{opt}(j) := x(j), j = 1, \ldots, m$; $f_{opt} := f(x_{opt})$;
  { $x_{opt}(j) := x(j), j = 1, \ldots, p$ represents the incumbent solution
  (best solution found so far)}

– $nbiter := 0$; { current iteration }

– $bestiter := 0$; { iteration when the best solution has been found }

– $cnt1 := p; cnt2 := m$; { initial values for counters in two tabu lists }

– $d(j) := 0; j = 1, \ldots, m$; { Long - term memory array $d(j), j = 1, \ldots, m$, represent the iteration number when variable j left the solution the last time. }

Step 3. Diversification loop.
for $k = 1$ to *nlong* do

Step 4. Descent-ascent search.
while $(nbiter - bestiter < k \cdot nbmax)$ do

$nbiter := nbiter + 1$;

Step4.1. Find the best solution in the neighbourhood
– Generate a set of neighbour solutions $N(x)$;
– choose non tabu solution x^* optimizing f over $N(x)$;
($x^*(j)$ differs from $x(j)$ in only two indices *out* and *in*)
– check if there is a tabu solution better than f_{opt} (aspiration level);
in that case, $x_{opt} := x$.

Around $(m, n, p, t1, t2, x, f^*, x_{opt}, f_{opt}, out, in)$.

Step 4.2. Keep the best incumbent solution
if $(f^* < f_{opt})$ **then**

$f_{opt} := f^*; x_{opt}(out) := x(in); bestiter := nbiter;$

endif;

Step 4.3. Update long - term memory
$d(x(out)) := nbiter$;

Step 4.4. Move to the new solution and update tabu lists
Chain - substitution - move $(out, cnt1, in, cnt2, x)$;
$cnt1 := cnt1 - 1$; **if** $(cnt1 = p - t1)$ **then** $cnt1 := p$; **endif**;
$cnt2 := cnt2 - 1$; **if** $(cnt2 = m - t2)$ **then** $cnt2 := m$; **endif**;

endwhile

Step 5. Diversification
{ Bring into the solution variable j^* that has been out for the longest time,
i.e., – find the minimum of $d(j), j = p+1, \ldots, m$, and
– exchange its position with one from T_1. }

$j^* := \text{argmin}\{ d(j), j = p+1, \ldots, m \}$;

$xx := x(p); x(p) := x(j^*); x(j^*) := xx$;

endfor ***k***.

Obviously, the complexity of one iteration of the descent-ascent search (*Step 4*) depends on the complexity of the procedure **Around** (*Step 4.1.*) since **Chain-substitution-move** is $O(1)$. We shall now give a pseudo-code for the *greedy or stee-*

pest descent mildest ascent (Hansen (1986)) version of the algorithm, i.e., a version that explores the complete neighbourhood consisting of $p(m-p)$ solutions. Then modifications in order to get defferent versions are obvious: (i) *partial greedy* - take one facility out of the solution at random and exchange it with all $m-p$ facilities that do not belong to the current solution; (ii) *anxious search* - enumerate all neighbourhood solutions until the first improvement; (iii) *random search* - take v (a parameter) neighbourhood solutions at random; (iv) *random walk* - take one neighbourhood solution at random, etc. Note that for the random walk algorithm we need no tabu lists ($t_1 = t_2 = 0$), since the probability of cycling is almost 0. A similar conclusion holds for random search. Note also that 1-*substitution* move (with all versions mentioned above) can in fact be viewed as a special case of our *chain-interchange* move, where the parameters $t_1 = |T_1|$ and $t_2 = |T_2|$ are set at zero.

The detailed pseudo-code of **Around** follows.

Around $(m, n, p, t_1, t_2, x, f^*, f_{opt}, x_{opt}, out, in)$

```
f* := big;
for i = 1 to p { Enumerate candidates that go out of the solution }
    jj := x(i);
    for k = p+1 to m { Enumerate candidates that go into the solution }
        x(i) := x(k); { Interchange facilities }
        f := f(x); { Find objective function in x }
        if (i ≤ p − t1 and k ≤ m − t2) then { Keep the best non tabu solution }
            if (f < f*) then
                f* := f; out := i; in := k; endif
            else { Aspiration criterion }
                if (f < f_opt) then
                    f_opt := f; out := i; in := k; x_opt(out) := x(in);
                endif;
        endif;
    endfor k;
    x(i) := jj;
endfor i;
```

The *Around* procedure employs a simple type of aspiration criterion, consisting of removing a tabu status from a trial move when it yields a solution better than the best obtained so far (f_{opt}). Then it becomes both, the new current and new incumbent solution.

3.4. Extensions

In the framework of TS both intensification and diversification of the search are performed using so - called long - term memory function. It has been not-

ed that search with memory has many more chances to visit unexplored regions of the solution space than random multistart search. Moreover, multistart implementation suffers from 'central-limit catastrophe' (see Boese, Kahng and Muddu (1994)) when the problem size grows large. In Boese, Kahng and Muddu (1994) it is shown that usually in combinatorial problems, very good solutions are located near other good solutions. That result may explain why simulated annealing, tabu search and other descent-ascent local search heuristics have been so successful in practice. But these also explain why TS has been successful even without using the long-term memory function (see Glover and Laguna 1994 & 3.4. for such applications).

However, there are easy ways to implement TS long -term memories in our algorithm. One possibility has already been explained in steps 4.3. and 5 of the algorithm: d_j represents the last iteration number when facility point j (attribute of the solution) has been in the solution (Sum and McKeown (1993)). Note that only one operation in each iteration is needed to update sequence d_j (Step 4.3).

Another possibility is to introduce *frequency-based-memory* (Glover and Laguna (1994)) in the chain-interchange method. Let r_j represent counts of the number of occurrences of a facility point j in the solution in previous iterations. When improvement with local search is no longer possible, intensification and diversification could be performed using array r_j (in Step 5 of the algorithm) in some of the following ways:

Table 1: Results for 47 European Cities: $maxit$=10000, $nlong$=1, t_1=1, t_2=3

		Objective function value			% deviation			CPU Time (sec)		
p	z_{opt}	RW	1-int	CS	RW	1-int	CS	RW	1-int	CS
5	28711	28711	29938	28711	0.00	4.27	0.00	9.00	0.36	2.47
10	17677	17845	18474	17677	0.95	4.51	0.00	5.54	0.42	1.55
15	12749	12918	12749	12749	1.33	0.00	0.00	4.47	0.73	1.29
20	9328	9762	9657	9328	4.65	3.53	0.00	3.96	0.75	1.15
25	6561	6908	6920	6621	5.29	5.47	0.91	3.68	0.53	1.06
30	4455	4705	4530	4455	5.61	1.68	0.00	3.50	0.59	0.98

- **Intensification by Selection.**
  Take p largest elements from r_j and put corresponding facilities into the new solution. Among them, put corresponding facilities into the new that correspond to t_1 largest members of r_j in T_1 ;

- **Diversification by Negative Selection.**
  Bring into the solution p facilities with corresponding p smallest values of array r_j (or d_j), etc.

4. COMPUTATIONAL EXPERIENCE

Two sets of experiments were conducted for the *p-median* problem only. The

first one examined road distances between 47 important European cities (Moreno, Garcia and Moreno-Vega (1994)); the second one used the Ruspini (1970) data for 75 fixed points, which is widely referenced in classification or clustering theory. Without loss of generality, in all testing it is assumed that a set of potential facilities is equal to a set of customers ($L=D$, $m=n$). We reported in Moreno, Moreno and Mladenović (1994) a comparison of Tabu search, Simulating Annealing and Genetic Algorithms in solving the p-median problem. It was shown that the Tabu search method gives better results than both the Simulating Annealing and Genetic Algorithm. Here we compare the results of our Chain-Substitution method (CS) with another two well-known methods: the Random Walk (RW) algorithm (with local improvement of the solution) and descent 1-Interchange (1-Int). In the Random Walk algorithm both, the facility that goes out and that goes in are chosen at random. The new solution determines the partition of the set of customers into p groups. Local improvement consists of solving separately p 1-facility problems, i.e., the best center for each given set of customers is found. We obtain the 1-Interchange method as a special case of our Chain-Interchange method by setting parameters t_1 and t_2 to zero (tabu lists are empty). RW and CS terminate when a given maximum number of iterations *maxit* is reached, while 1-Int stops naturally (when there is no better solution in the neighbourhood of a current solution). The results for the 47 European cities problem are given in Table 1. p represents the number of new facilities and z_{opt} the exact minimum value (Klose (1994)). The next columns contain the objective function values, % deviation calculated relative to z_{opt} and CPU times for all methods compared.

Table 2: Results for Ruspini data: *maxit*=25000, *nlong*=1, t_1=1, t_2=3

		Objective function value			% deviation			CPU Time (sec)		
p	z_{opt}	RW	1-int	CS	RW	1-int	CS	RW	1-int	CS
5	779.68	779.68	796.44	779.68	0.00	2.15	0.00	49.57	1.68	14.09
10	512.81	515.14	512.81	512.81	0.45	0.00	0.00	29.51	3.52	8.67
15	389.98	390.28	393.83	393.83	0.08	0.99	0.99	22.75	5.45	7.12
20	314.10	318.66	315.05	315.05	1.45	0.30	0.30	19.20	6.90	6.83
25	252.43	263.50	257.45	255.80	4.39	1.99	1.33	17.12	6.11	5.97
30	199.41	214.28	208.56	207.37	7.45	4.59	3.99	15.79	5.68	5.40
35	159.90	169.88	166.48	163.10	6.24	4.11	2.00	14.93	5.07	4.98
40	127.63	139.56	132.46	128.62	9.35	3.78	0.77	14.30	4.74	4.62

The results for Ruspini data are summarized in Table 2. using the same format as Table 1. The Chain-Substitution (CS) algorithm out-performs both 1-Int and RW, while using smaller CPU times.

5. CONCLUSIONS

The 1-Interchange (swap) move was successfully applied to a large class of combinatorial problems, but it did not allow escaping from local optima traps. We suggest a simple extension of this move, *chain-interchange* move, that allows us to obtain an efficient descent-ascent local search heuristic of the Tabu search type.

The list of the implementations of 1-interchange, and thus chain-interchange moves is very large. Even the simplex method for linear programming and some con-

cave minimization methods are of that type: one variable goes out of the basis, another goes in. On the other hand, the same problem could have different interchange strategies, i.e., different data structures could be developed to solve the same problem by the using interchange method. For example, in directly-allocated LA problems the same solution can be represented as a subset of vertices (locations) as well as a subset of edges (allocations) of some graph. Obviously, the set of neighbourhood solutions obtained by all possible interchanges in both cases is not the same. This fact could limit a general approach in which problem-specific knowledge is ignored. In this paper we suggested some "middle" approach: recognize a class of problems with similar properties , such that the same data structure (the same algorithm) may be applied. We addressed our attention to location-allocation problems which have similar problem-specific knowledge. To solve them we developed a version of the chain-interchange method, the *chain-substitution* algorithm.

Future work on this "middle" approach with the chain-interchange methods could be developed in four directions: (i) *enlarge* the class of problems that could be solved efficiently with the *chain-substitution* algorithm proposed in & 3.3 of this paper (by only changing the subroutine that evaluates objective function), and compare results with other good heuristics (for each particular problem). These could be for example *p*-cluster problems, *p*-dispersion problems, simple plant location problems, quadratic knapsack problems, concave minimization problems etc.; (ii) to solve large instances of LA problems *extend* the basic version of the *chain-substitution* algorithm using some hybrids or other known ideas from TS; (iii) develope data structures that allow implementation of other version of the chain-interchange method such as *chain-reallocation*, *chain-opening*, *chain-closing* etc. and compare results with the *chain-substitution* algorithm; (iv) in this paper we propose a 1-chain-interchange method. Obviously it could be generalized to a *k-chain-interchange* algorithm. Then it becomes a hybrid of the Variable Neighbourhood Algorithm (Mladenović (1995)) and TS.

Acknowledgment. This work was partly supported by project number 91/237 of the Gobierno Autónomo de Canarias, Spain. The first author was also supported by the Office of Naval Research grant N00014-92-J-1194, Canada.

REFERENCES

[1] Brandeau, M.L., and Chiu, S.S., "An overview of representative problems in location research", *Management Science* 35 (1989) 645-674.

[2] Boese, D.K., Kahng, B.A., and Muddu, S., "A new adaptive multi-start technique for combinatorial global optimizations", *Operations Research Letters* 16 (1994) 101-113.

[3] Cooper, L., "Heuristic methods for location-allocation problems", *SIAM Review* 6 (1964) 37-53.

[4] Glover, F., "Future paths for integer programming and links to artifical intelligence", *Comput. & Oper. Res.* 5 (1986) 533-549.

[5] Glover, F., "Tabu search Part. I", *ORSA J. Computing* 1 (1989) 190-206.

[6] Glover, F., "Tabu search Part II", *ORSA J. Computing* 2 (1990) 4-32.

[7] Glover, F., and Laguna, M., "Tabu search", in C. Reeves(ed.), *Modern Heuristic Techniques for Combinatorial Problems* (Chapter 3), Oxford, Blackwell, 1994, *Annals of Operations Research* 41 (1993) 3-28.

[8] Handler, G.Y., and Mirchandani, P.B., *Location on Networks*, MIT Press. 1979.

[9] Hansen, P., "The steepest ascent mildest descent heuristic for combinatorial programming", Presented at the *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[10] Hansen, P., and Jaumard, B., "Algorithms for the maximum satisfiability problem", *Computing* 44 (1990) 279-303.

[11] Jacobsen, S.K., "Heuristic for the capacitated plant location model", *E.J.O.R.* 12 (1983) 253-261.

[12] Klose, A., "A comparison between the Erlenkotter algorithm and a branch and bound algorithm based on subgradient optimization to solve the uncapacitated facility location problem", *OR Proceeding*, Springer, 1994, 335-339.

[13] Kuehn, A.A., and Hamburger M.J., "A heuristic program for locating warehouses", *Management Science* 9 (1963) 643-666.

[14] Maranzana, F.E., "On the location of supply points to minimize transportation costs", *Operations Research Quarterly* 12 (1964) 138-139.

[15] Mladenović, N., "Variable neighbourhood algorithm - a new meta-heuristic for combinatorial optimization", *Optimization days*, Montreal, May 10-12, 1995, pp-22.

[16] Moreno, J.A., Moreno-Vega, J., and Mladenović, N., "Tabu search and simulated annealing in p-median problem", *CORS'94 - Optimization days*, May 29 - June 1, 1994, pp-125.

[17] Moreno, J.A., Garcia, R.L., and Moreno-Vega, J.A., "A parallel genetic algorithm for the discrete p-median problem", *Studies in Locational Analysis*, 7 (1994) 131-141.

[18] Moreno, J.A., Rodriguez, C., and Jimenez, N., "Heuristic cluster algorithm for the multiple facility location-allocation problem", *Revue de Automatique, Informatique et de Recherche Operationelle* 25 (1991) 97-107.

[19] Ruspini, E.H., "Numerical methods for fuzzy clustering", *Information Sciences* 2 (1970) 319-350.

[20] Sum, M., and McKeown, "Tabu search applied to the general fixed charge problem", *Annals of Operations Research* 41 (1993) 405-420.

[21] Teitz, M.B., and Bart, P., "Heuristic methods for estimating the generalized vertex median of a weighted graph", *Operations Research* 16 (1968) 955-961.