A double annealing algorithm for discrete location/allocation problems

Citation for published version (APA):

Righini, G. (1994). *A double annealing algorithm for discrete location/allocation problems*. (Memorandum COSOR; Vol. 9404). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1994

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

Memorandum COSOR 94-04

**A double annealing algorithm for discrete
location/allocation problems**

G. Righini

Eindhoven, February 1994
The Netherlands

A double annealing algorithm for discrete location/allocation problems*

Giovanni Righini

Dipartimento di Elettronica e Informazione, Politecnico di Milano
piazza Leonardo da Vinci 32, 20133 Milano

Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven
P.O.Box 513, 5600 MB Eindhoven

February 1994

Abstract

Many algorithms inspired by the simulated annealing paradigm for approximately solving combinatorial optimization problems have been presented and tested in the last years. The aim of this paper is to present a double-annealing algorithm, a new idea for the application of annealing-based algorithms to discrete location/allocation problems, with two mutually dependent sets of binary variables. The double annealing algorithm consists of two annealing processes synchronized with each other; each process is executed on a different subset of variables and with a different annealing parameter ("temperature") and the synchronization depends on the saturation of the two variable subsets. In order to improve such synchronization, deannealing steps are also performed. The double annealing algorithm is quite robust and easy to tune (it is rather insensitive to the initial values of the annealing parameters and to the initialization) and it is able to achieve good approximate solutions. The experiments were done on the P -median problem.

Keywords: simulated annealing, location/allocation problems

*Partly supported by research contracts MURST 40% and 60% of the Italian Ministry of University and Scientific Research. This paper was completed while the author was working at Eindhoven University of Technology with the support of the Human Capital and Mobility program of the European Community.

1 Introduction

This paper addresses vertex-constrained multi-facility location problems on graphs, which belong to the class of discrete location/allocation problems. We concentrate on problems with a min-sum objective function, such as the P -median problem. Through a boolean formulation we put in evidence that the binary variables are partitioned into two subsets, that correspond to an allocation subproblem and to (a number of instances of) a location subproblem. Since both subproblems are easily solvable to optimality in polynomial time, a natural way to cope with location/allocation problems is an iterative cluster-and-locate algorithm, that alternates between an allocation step and a location step [14]. Unfortunately, such an algorithm may get trapped in poor local optima. In the attempt to avoid this pitfall, it is possible to take recourse to annealing techniques; in fact, since simulated annealing (SA) was first proposed by Kirkpatrick, Gelatt, Vecchi [11], many applications were developed, pointing out that it can give very good solutions, in comparison with other approximation algorithms, although it is often more time-consuming. A development of SA recently applied to Combinatorial Optimization (CO) is mean-field annealing (MFA) [9] [17], a similar technique inspired by the analogy with the physical annealing process in systems of magnetic spins in interaction; MFA is a deterministic version of SA. Our contribution consists of the application of the MFA technique in a new way, that seems particularly suitable for discrete location/allocation problems. We split the annealing process into two parallel processes, synchronized with each other; we define a value of saturation for each subset of variables and we use it in order to synchronize the two annealing processes. Therefore the annealing schedule is automatically tuned by the algorithm itself. Experimental tests show that the behaviour of such a *double annealing* algorithm is substantially improved when deannealing steps are also performed; in these, the value of the annealing parameters ("temperatures") is increased instead of decreased. The criterion we choose to compare our double annealing algorithm with the cluster-and-locate one is based both on quality and robustness. By *quality* of an algorithm we mean the value of the best solution found by it; by *robustness* of an algorithm we mean the value of the solutions found by it in the worst and in the average case; the average is computed over different instances of the same problem as well as over different runs on the same instance with random initialization and with different choices of parameters values. The double annealing algorithm is quite robust and it often achieves solutions better than those found by the cluster-and-locate algorithm in an equal amount of running time (in particular on large instances).

The paper is organized as follows. Some basic concepts on the MFA technique are outlined at the end of this section. In section 2 we define the problems we consider, we prove their equivalence with the P -median problem, we discuss their computational complexity and we illustrate their decomposition into two polynomially solvable subproblems. In section 3, after a brief review on the state-of-the-art, we present a sequence of algorithms: the former is the classical cluster-and-locate algorithm of

Maranzana [14]; the subsequent ones are based on the MFA technique. In section 4 we report the outcome of our experiments. Finally, in section 5 we outline some conclusions and perspectives for future research.

Simulated annealing and mean-field annealing SA is a probabilistic method: it requires the computation of probabilities. Probabilities can be used to assign binary values to variables of a CO problem as well as to accept or refuse a change in the value of a variable; the former happens in constructive algorithms, the latter in local search based algorithms. The values of the probabilities are computed at each iteration as functions of an annealing parameter T , which plays a role similar to the one of temperature in physical annealing processes: as the value of T decreases, the values of probabilities tend to 0 or 1 and therefore the value assignment to variables tends to become stable. On the contrary, in MFA algorithms real values are assigned directly to the variables. Their calculation is made by the following recurrent formula:

$$u_i(t) = \frac{\partial E(x)}{\partial x_i}(t) \quad x_i(t+1) = s\left(-\frac{u_i(t)}{T(t+1)}\right)$$

where $E(x)$ is the objective function to be minimized, s is a sigmoidal function, usually a hyperbolic tangent $s(z) = \frac{1}{2}(1 + \tanh(z))$, and $T(t)$ is the value of the annealing parameter at iteration t . As T decreases the values of the variables become close to 0 or 1. No probabilities are used; therefore MFA is just a deterministic version of annealing. In general, the constraints of the CO problem may be difficult to handle in MFA algorithms; nevertheless some constraints are easily managed by an iterated normalization: they are the allocation (or set-partitioning) constraints. Imagine that the variables of the problem are organized in a matrix $X_{R \times C}$ and that they are subject to the constraints $\sum_{c=1}^C x_{rc} = 1$, $\forall r = 1, \dots, R$. Set-partitioning constraints belong to the formulation of many CO problems, such as assignment, transportation, travelling salesman, graph partitioning and coloring, plant and facility location. By substitution in the MFA recurrent formula, taking s as the hyperbolic tangent and introducing normalization, one obtains for every variable x_{rc} :

$$x_{rc}(t+1) = \frac{e^{-\frac{1}{T(t+1)} \frac{\partial E(x)}{\partial x_{rc}}(t)}}{\sum_{c=1}^C e^{-\frac{1}{T(t+1)} \frac{\partial E(x)}{\partial x_{rc}}(t)}}$$

which ensures the satisfaction of the allocation constraints at each iteration. Together with a suitable annealing schedule, which defines the initial value of T and the way of decreasing it at each iteration, the formula above is the kernel of any MFA algorithm. Annealing-based algorithms can be applied to a large variety of CO problems: in particular, for applying the MFA technique the only information one needs is the expression of the partial derivative $\frac{\partial E(x)}{\partial x}(t)$. Annealing-based algorithms are therefore general purpose algorithms; the advantage of not needing any special procedure depending on the particular problem at hand has its counterpart in the difficulty of properly tuning the parameters and in the relative slowness, in comparison with specialized algorithms.

2 Min-sum discrete location/allocation problems

Let us introduce a class of discrete location/allocation problems: vertex-constrained multi-facility location problems on graphs. Let us consider an undirected graph $G(U, A)$ with $|U| = V$, with positive integer weights on the arcs. N users and a number P of facilities are given, with $1 < P < N$. Users locations are also given, whereas facilities must be located in an optimal way. Users and facilities are constrained to be on the vertices of G . Without loss of generality, suppose that N vertices of the graph coincide with the users, numbered from 1 to N ; let us indicate with $loc(j)$ the vertex on which facility j is to be located ($j = 1, \dots, P$). The distance between each user $i = 1, \dots, N$ and each facility $j = 1, \dots, P$ may be defined in different ways, as a function of the location of i (vertex i), the location of j (vertex $loc(j)$) and the weights of the arcs. We indicate such a function with $d(i, loc(j))$. A cluster C_j is the set of users that are associated to (served by) the same facility j , according to the minimum distance criterion: $C_j = \{i \mid d(i, loc(j)) = \min_{k=1, \dots, P} \{d(i, loc(k))\}\}$. In the literature on location problems both min-sum and min-max objective functions have been studied. Hereafter we consider only min-sum objective functions since their partial derivative is required. Multi-facility location problems, like all location/allocation problems, consist of the combination of two subproblems: an allocation subproblem, whose output is the set of clusters C_j , and P instances (one for each cluster) of a single-facility location subproblem, whose output consists of the positions $loc(j)$ of facilities. The following boolean formulation puts this structure in evidence:

$$\begin{aligned}
 \min f &= \sum_{j=1}^P \sum_{i=1}^N \sum_{v=1}^V w_{ij} x_{jv} d(i, v) \\
 \sum_{j=1}^P w_{ij} &= 1 \quad \forall i = 1, \dots, N \\
 \sum_{v=1}^V x_{jv} &= 1 \quad \forall j = 1, \dots, P \\
 w_{ij} &\in \{0, 1\} \quad \forall i = 1, \dots, N \quad \forall j = 1, \dots, P \\
 x_{jv} &\in \{0, 1\} \quad \forall j = 1, \dots, P \quad \forall v = 1, \dots, V
 \end{aligned}$$

The problem variables are partitioned into two subsets: allocation variables w and location variables x ; variables w define the allocation of users to facilities ($w_{ij} = 1 \Leftrightarrow i \in C_j$), whereas variables x , define the locations of facilities ($x_{jv} = 1 \Leftrightarrow loc(j) = v$). Both subsets of variables are subject to set-partitioning constraints and all variables are subject to an integrality constraint. Let us define two matrices $W_{N \times P}$ and $X_{P \times V}$, whose entries are w_{ij} and x_{jv} . In the formulation above, the costs $d(i, v)$ depend on input data only and are thus computable a priori in a preprocessing phase, in time polynomial in N and V and independent of P ; in the remainder they are indicated by d_{iv} . Any multi-facility vertex-constrained location problem on a graph with a min-sum objective function is then equivalent to a P -median problem on a graph with weights

$d_{iv} = d(i, v)$ on the arcs. The computational complexity of all multi-facility vertex-constrained location problems on graph is therefore the same as for the P -median problem; because of this equivalence all experiments outlined in section 4 were made on P -median problem instances.

Complexity The P -median problem for general P is NP -hard [12], even when the graph is euclidean [16]. However both the allocation and the single-facility location subproblems are solvable to optimality in polynomial time.

Allocation subproblem: procedure P1. Suppose locations of facilities are given; apply the definition of cluster: assign every user to the facility nearest to it. Procedure P1 takes $O(N \times P)$ time and gives an optimal clustering of the users.

Single-facility location subproblem: procedure P2. Suppose now that all clusters are given. For every $j = 1, \dots, P$ do the following enumeration: consider V solutions, obtained locating facility j on each vertex of the graph and choose the best one among them. The enumeration takes $O(V \times N)$ for every facility and therefore $O(P \times V \times N)$ globally.

Both subproblems are still polynomial in their weighted version, when a real weight is associated to every facility or to every user.

3 Algorithms

State of the art There exist both optimization and approximation algorithms for the P -median problem. In [4], [15] and [8] optimization algorithms, which take exponential time, are presented; approximation algorithms are divided into different categories: nodes partitioning [14] [2], nodes insertion [2] [3], nodes substitution [22] [3], greedy [3], truncated branch-and-bound [8]. For a review see, for instance, [6].

For the sake of a clear presentation of our contribution, we sketch the structure of five algorithms: namely, the cluster-and-locate algorithm [14], a partial annealing algorithm [13] [20], in which only allocation variables are subject to annealing, a complete annealing algorithm, in which all variables are subject to annealing, the double annealing algorithm [21], in which variables are subject to different annealing processes and the annealing-deannealing algorithm, in which deannealing steps are also performed. Each algorithm stems from the discussion of the limits put in evidence by the previous one.

3.1 Cluster-and-locate algorithm

The cluster-and-locate algorithm for the P -median problem [14] follows a well known paradigm for all location/allocation problems: the same idea is present, for instance, in the algorithm of Cooper [2] for continuous location problems as well as in some clustering-and-routing algorithms for the Vehicle Routing Problem [1]. The algorithm repeats iteratively two steps in which each subproblem is solved at optimality.

ALGORITHM 1

```
Initialization-of-locations;  
Cycle [P1;  
       $\forall j = 1, \dots, P$  do  
        P2;  
      End-test-of-algorithm];
```

Initialization-of-locations is any procedure that generates initial locations for the facilities (for instance a generator of random numbers in $[1..V]$), P1 and P2 are the polynomial optimization procedures sketched in section 2 and End-test-of-algorithm is a test for the stability of the solution between two consecutive iterations.

Algorithm 1 is very fast. Although there is no bound to the number of iterations required to converge, this number is often small on random generated graphs even of a big size. On the other hand the quality of the solutions is poor. In our experiments the average value of the solutions was far from optimal (between 12% and 15%).

3.2 Partial annealing algorithm

In the partial annealing algorithm [13] [20] the computation of the allocation variables is done by the MFA formula.

ALGORITHM 2

```
Initialization-of-locations;
Cycle  [MFA-Allocation;
        Annealing-of- $T_w$ ;
         $\forall j = 1, \dots, P$  do
            Weighted-P2;
        End-test-of-algorithm];
```

- Initialization-of-locations is as before.
- MFA-allocation is the procedure that computes the values of allocation variables as:

$$w_{ij}(t+1) = \frac{e^{-\frac{1}{T_w(t)} \frac{\partial f}{\partial w_{ij}}(t)}}{\sum_{j=1}^P e^{-\frac{1}{T_w(t)} \frac{\partial f}{\partial w_{ij}}(t)}} \quad \forall i = 1, \dots, N \quad \forall j = 1, \dots, P$$

where $\frac{\partial f}{\partial w_{ij}}(t) = d(i, loc(j)(t))$.

- Annealing-of- T_w is the annealing step. In general, an annealing step decreases the value of the annealing parameter: here the parameter is called T_w because only allocation variables w directly depend on it. Its value is decreased by multiplying it by a scale factor $Sf < 1$ (typical values of Sf range from 0.90 to 0.99). The plot of T_w as a function of the number of annealing steps is therefore a decreasing exponential curve. An annealing step takes place if and only if the values of the variables (allocation variables, in this algorithm) are sufficiently stable: that is, if and only if $\frac{1}{N \times P} \sum_{i,j} |w_{ij}(t+1) - w_{ij}(t)| \leq \delta$, where δ is a small parameter (in our experiments we set $\delta = 0.001$).
- Weighted-P2 is the optimal polynomial procedure that computes the new values of location variables $loc(j)(t)$ at each iteration: when $loc(\vec{j})(t)$ is computed, the weights of the users are given by the vector $w_{i,\vec{j}}(t)$ for all users $i = 1, \dots, N$.
- End-test-of-algorithm is based on the concept of saturation, that is explained hereafter.

Saturation The saturation of a set of variables is a quantity that gives information regarding the satisfaction of the integrality constraints. We define the saturation $sat(W)$ so that its value is close to 1 if and only if all entries of W have “almost binary” values. It is possible to define saturation in different ways. The def-

Figure 1: Saturation of an entry w_{ij} of matrix W

inition used in our experiments is graphically illustrated in figure 1 for a single entry of W . We indicate the saturation of an entry with $sat(w_{ij})$, the saturation of a row with $sat(w_i) = \frac{1}{P} \sum_{j=1}^P sat(w_{ij})$ and the saturation of the whole matrix with $sat(W) = \frac{1}{N} \sum_{i=1}^N sat(w_i)$. When other definitions of saturation were used (such as the average of the greatest entry values in every row, or the average of the difference between the greatest and the second greatest entries in every row), they did not affect the behaviour of the algorithm. As the annealing process evolves and the value of T_w decreases, $sat(W)$ gradually approaches 1. The stopping criterion End- test-of-algorithm is $sat(W) \geq \sigma$, where σ is a threshold parameter (set to 0.99 in all our experiments).

Algorithm 2 was developed and successfully applied to continuous multi-facility location (that is continuous location/allocation) problems: namely in [13] for locations in the euclidean plane and in [20] for absolute locations on graphs. In spite of its relative slowness in comparison with the cluster-and-locate algorithm, Algorithm 2 is much more robust and often outperformed cluster-and-locate algorithm, finding better solutions in the same amount of time. Though Algorithm 2 can be applied to vertex-constrained location (that is discrete location/allocation) problems as well, the results are unsatisfactory, mainly because of ties; ties arise when two or more facilities coincide in the same vertex, which is likely to occur already in the initial iterations, when T_w is high. When W has some equal columns and X some equal rows, the algorithm is not able to take any decision to resolve the tie and it gets trapped in an infeasible (fractional) local optimum. The addition of a small random noise to allocation variables is useless when T_w is high, whereas it is equivalent to a coin toss between the conflicting alternatives when T_w is low. The coin toss effect on ties at low values of T_w is evident in figure 2. In our experiments such behaviour lead to poor results.

Figure 2: Values of f , T_w and $sat(W)$ during the evolution of the partial annealing algorithm (Algorithm 2) on a $(10, 70, 10)$ P -median problem instance

3.3 Complete annealing algorithm

One idea arising from the observations above is to allow fractional values for the location variables, too. In the following algorithm the MFA technique is applied to all variables.

ALGORITHM 3

Initialization;

Cycle [MFA-Allocation;
MFA-Location;
Annealing-of- T ;
End-test-of-algorithm];

- In Initialization all allocation variables are given a value close to $1/P$ and all allocation variables are given values close to $1/V$. A small random noise is also added to all variables in order to break the initial simmetry.
- Both MFA-Allocation and MFA-Location consist of the application of the MFA formula:

$$w_{ij}(t+1) = \frac{e^{-\frac{1}{T(t+1)} \frac{\partial f}{\partial w_{ij}}(t)}}{\sum_{j=1}^P e^{-\frac{1}{T(t+1)} \frac{\partial f}{\partial w_{ij}}(t)}} \quad \forall i = 1, \dots, N \quad \forall j = 1, \dots, P$$

$$x_{jv}(t+1) = \frac{e^{-\frac{1}{T(t+1)} \frac{\partial f}{\partial x_{jv}}(t)}}{\sum_{v=1}^V e^{-\frac{1}{T(t+1)} \frac{\partial f}{\partial x_{jv}}(t)}} \quad \forall j = 1, \dots, P \quad \forall v = 1, \dots, V$$

where the partial derivatives are $\frac{\partial f}{\partial w_{ij}}(t) = \sum_{v=1}^V x_{jv}(t) d_{iv}$ and $\frac{\partial f}{\partial x_{jv}}(t) = \sum_{i=1}^N w_{ij}(t) d_{iv}$.

- End-test-of-algorithm is $sat \geq \sigma$, where sat is computed disregarding the subdivision of variables into two sets, that is $sat = \frac{1}{(N \times P) + (P \times V)} (\sum_{i,j} sat(w_{ij}) + \sum_{j,v} sat(x_{jv}))$.

The behaviour of Algorithm 3 (see figure 3) is still unsatisfactory. Because of the different length of the rows in W and in X , saturation is not simultaneous in the two matrices. As T is lowered, the entries of X (the largest of the two matrices) are forced to “almost binary” values before the entries of W . In figure 3 a critical value of T is evident, for which the saturation suddenly grows. In correspondence to the critical value $sat(W)$ is still close to 0 while $sat(X)$ is already close to 1. So the almost binary values of location variables influence the values assignment to the allocation variables

but not vice versa: once X is saturated the algorithm is trapped in a (usually bad) local optimum from which it cannot escape. Moreover for low values of T ties occur again for the same reasons as before. The next improvement therefore consists of the use of two different annealing parameters and to split the annealing process into two processes.

Figure 3: Values of f , T and sat during the evolution of the complete annealing algorithm (Algorithm 3) on a $(10, 70, 10)$ P -median problem instance

3.4 Double annealing algorithm

According to the iterative scheme of the cluster-and-locate algorithm, the double annealing algorithm [21] executes alternately the two steps of allocation and location: keeping the values of variables w fixed, new values are computed for variables x and vice versa. Each annealing process is characterized by a different annealing parameter and the task of synchronization is accomplished through $sat(W)$ and $sat(X)$.

ALGORITHM 4

```

Initialization;
Cycle  [Cycle  [MFA-Allocation;
              Annealing-of- $T_w$ ;
              End-test-of-allocation];
      Cycle  [MFA-Location;
              Annealing-of- $T_x$ ;
              End-test-of-location];
      End-test-of-algorithm];

```

- Initialization is as in Algorithm 3.
- MFA-Allocation and MFA-Location consist of the application of the MFA formula as in Algorithm 3, with the difference that $T(t + 1)$ is replaced by $T_w(t + 1)$ in the expression of $w_{ij}(t + 1)$ and by $T_x(t + 1)$ in the expression of $x_{ju}(t + 1)$.
- Annealing-of- T_w and Annealing-of- T_x are the annealing steps: since T_x and T_w are decreased independently, their plots may be quite different even if the same parameters Sf and δ are used in both annealing schedules.
- End-test-of-algorithm is $(sat(W) \geq \sigma) \wedge (sat(X) \geq \sigma)$.
- The synchronization between the two annealing processes is obtained through End-test-of-allocation and End-test-of-location in the following way. Let us call $Z^{(ann)}$ the set of the variables whose values are computed through the MFA formula and $Z^{(fix)}$ the set of the variables that are fixed. The annealing on $Z^{(ann)}$ stops when $sat(Z^{(ann)}) > sat(Z^{(fix)})$. In previous papers (for instance, [17]) the concept of saturation had already been introduced but its use was limited to the definition of a stopping criterion, like End-test-of-algorithm. On the contrary in Algorithm 4 the value of saturation directly influences the annealing schedule.

An example of the behaviour of the algorithm is plotted in figure 4. In some tests with the double annealing algorithm, we observed the following two facts.

Figure 4: Values of f , T_w , T_x , $sat(W)$ and $sat(X)$ during the evolution of the double annealing algorithm without deannealing (Algorithm 4) on a $(10, 70, 10)$ P -median problem instance

- In an early phase, characterized by high values of T_w and T_x , the saturation of each matrix tends to inhibit the saturation of the other (mutual inhibition); in a second phase the saturation of each matrix tends to augment the saturation of the other (mutual excitation); in a final phase the same happens as in the earliest one (mutual inhibition, again). Mutual inhibition and excitation may be explained in the following way. Suppose the allocation step has been performed until $sat(W) > sat(X)$; variables w are then kept fixed and the MFA-location procedure starts; let us call sat_1 the value of $sat(W)$ and T_1 the value of T_w at this point. After the execution of the location step, $sat(X) = sat_2 > sat_1$; the allocation step restarts with $T_w = T_1$. The first time $sat(W)$ is computed again it has value, say, sat_3 : if $sat_3 < sat_1$ (resp. $sat_3 > sat_1$), the MFA-location procedure has inhibited (resp. excited) the saturation of the allocation matrix. It is also possible that more phases characterized by mutual inhibition and mutual excitation take place, as in figure 4: mutual inhibition corresponds to quasi-horizontal parts of the plots of f , $sat(W)$ and $sat(X)$, whereas mutual excitation corresponds to quasi-vertical parts of the plots.
- The saturation of the largest of the two matrices grows faster than the saturation of the other; that is, one of the two MFA-procedures (MFA- allocation) is executed many times every time the other (MFA-location) is executed once. This imbalance occurs both during mutual inhibition and during mutual excitation, although it is more evident during mutual excitation.

3.5 Annealing-deannealing algorithm

A suitable way to improve the performance of the double annealing algorithm is to synchronize the allocation and the location steps, in order to force the two annealing processes to evolve simultaneously: by simultaneously we mean that $sat(W)$ and $sat(X)$ must have similar values all along the execution of the algorithm. To this aim we use deannealing steps, in which the value of annealing parameters is increased instead of decreased (divided by Sf instead of multiplied). A deannealing step is done every time an annealing- based procedure stops and the other begins.

ALGORITHM 5

Initialization;

```

Cycle  [Cycle  [MFA-Allocation;
              Deannealing-of- $T_w$ ;
              End-test-of-deannealing;]
      Cycle  [MFA-Allocation;
              Annealing-of- $T_w$ ;
              End-test-of-annealing;]
      Cycle  [MFA-Location;
              Deannealing-of- $T_x$ ;
              End-test-of-deannealing;]
      Cycle  [MFA-Location;
              Annealing-of- $T_x$ ;
              End-test-of-annealing;]
      End-test-of-algorithm];

```

The End-test-of-annealing is $sat(Z^{(ann)}) > sat(Z^{(fix)})$ as before, whereas the End-test-of-deannealing is its logical negation: $sat(Z^{(ann)}) \leq sat(Z^{(fix)})$. Mutual excitation is thus contrasted by deannealing at each iteration. This phenomenon is clear in the left part of the plot of $sat(W)$ and $sat(X)$ in figure 5. As a consequence of deannealing steps, T_w and T_x are not monotonically decreasing any longer. The evolution of the double annealing algorithm is still driven by $sat(W)$ and $sat(X)$ more than by T_w and T_x and therefore the algorithm is insensitive to the choice of the initial values of T_w and T_x . The algorithm does not even require any special initialization of the variables and it is easy to tune, since Sf is the only parameter that needs to be tuned. Moreover, the quality of the solutions obtained is quite better: although it is much slower than Algorithm 1, the annealing-deannealing algorithm is so robust that it is quite competitive with it, as shown in the following section.

Figure 5: Values of f , T_w , T_x , $sat(W)$ and $sat(X)$ during the evolution of the double annealing algorithm with deannealing (Algorithm 5) on a $(10, 70, 10)$ P -median problem instance

4 Experimental results

We choose the P -median problem as a benchmark for the test of the algorithms, since it represents the whole class of vertex-constrained multi-facility location problems on graph with min-sum objective function. The size of the problem instances is indicated by the triple (V, N, P) . Every problem instance was randomly generated: arc weights were drawn from a uniform distribution in $[10, 50]$. Running time does not include the preprocessing needed to compute the all-pairs shortest paths matrix, from which the costs d_{iv} were derived; such preprocessing time was however negligible in comparison with the amount of time required by the annealing algorithms. Time is indicated by *minutes : seconds.hundredths*. All experiments were performed on a 66MHz PC. All algorithms were coded in Borland Turbo-Pascal.

Parameter tuning of double annealing algorithm The following table shows how the running time and the quality of the solutions depend on Sf in Algorithm 5. Data were obtained with 5 runs on a $(50, 30, 5)$ problem instance.

$Sf = 0.99$		$Sf = 0.95$		$Sf = 0.90$		$Sf = 0.85$		$Sf = 0.80$	
f	t	f	t	f	t	f	t	f	t
344	10:33.23	328	2:04.90	328	1:02.39	353	29.33	343	19.44
340	11:03.72	328	2:02.70	328	1:07.61	383	30.43	341	19.88
340	11:03.56	328	2:08.85	328	1:06.51	328	29.55	383	28.29
340	11:22.29	328	2:01.60	328	1:07.88	364	26.42	346	28.06
340	11:18.54	328	2:03.52	328	1:02.12	332	27.96	332	24.05

When Sf is properly tuned, the algorithm is rather insensitive to the random noise used in the initialization. The value of σ was set the first time to 0.99, since it seemed to be a reasonable choice, and it was never necessary to change it. The initial values of annealing parameters were $T_w = 10$ and $T_x = 500$.

Comparison with other algorithms The set of experimental results shown below concerns the comparison between the five algorithms outlined in section 3. The comparison was made on P -median problem instances of different sizes: $(50, 30, 5)$ and $(100, 70, 10)$. Annealing-based algorithms were run with different parameters values. The same amount of running time was used for each of the algorithms and for every set of parameters values; the number of runs of each algorithm, allowed in such a fixed time limit, is indicated in the rightmost column of the tables below. The number of times the best solution was obtained is reported between parentheses.

Size $V = 50$, $N = 30$, $P = 5$. Time=2 min.					
Algorithm	Sf	best solution	worst solution	average value of solutions	number of runs
Alg.1		330 (4)	487	371.704	5000
Alg.2	0.99	349 (1)	374	361.500	4
Alg.2	0.95	342 (1)	395	360.700	20
Alg.2	0.90	350 (2)	396	369.275	40
Alg.3	0.99	338 (1)	353	345.500	2
Alg.3	0.95	355 (1)	371	361.667	5
Alg.3	0.90	349 (1)	395	369.875	8
Alg.4	0.99	352 (1)	352	352.000	1
Alg.4	0.95	349 (2)	377	356.167	6
Alg.5	0.95	328 (1)	328	328.000	1
Alg.5	0.90	328 (2)	328	328.000	2

Size $V = 100$, $N = 70$, $P = 10$. Time=40 min.					
Algorithm	Sf	best solution	worst solution	average value of solutions	number of runs
Alg.1		709 (1)	948	820.485	12000
Alg.2	0.99	742 (1)	802	773.182	11
Alg.2	0.95	738 (1)	839	779.100	60
Alg.2	0.90	727 (1)	865	787.842	120
Alg.3	0.99	746 (1)	794	770.000	2
Alg.3	0.95	727 (1)	827	761.222	9
Alg.3	0.90	720 (1)	827	777.471	17
Alg.4	0.99	741 (1)	748	744.500	2
Alg.4	0.95	710 (1)	807	750.000	10
Alg.5	0.95	741 (1)	742	741.500	2
Alg.5	0.90	703 (1)	724	714.250	4

As the size of the problem increases, the average quality of the local minima in which Algorithm 1 gets trapped becomes worse and worse (see column “average value of solutions” in the tables above). Algorithm 5 is much more robust, since the average and the worst quality of the solutions found by it are quite better than those of Algorithm 1. Furthermore, when Algorithm 5 is properly tuned ($Sf = 0.90$ in this example), its quality is also competitive.

Comparison between speed and robustness The usual way to compare speed to robustness is the multistart approach: a fast algorithm is run several times, while a slower and more robust algorithm is run once or a small number of times, such that the

total time used by the two algorithms is the same; then the solutions found by both are compared. We followed this approach to compare Algorithm 1 (fast) with Algorithm 5 (robust). Algorithm 1 was run on a (50, 30, 5) problem, until it was able to find the same best solution (whose value is 328) found by Algorithm 5. It took 10 minutes and 25000 trials to obtain it once. Algorithm 5 was run for 5 times, it took 10 minutes globally, and it always achieved that best solution. Further experiments were done on problem instances of different size; the outcome is summarized in the following tables.

Size (50, 30, 5) Time=4 min.									
	Algorithm 1 (10000 runs)			Algorithm 5 (4 runs)			Algorithm 5 (2 runs)		
				$Sf = 0.90$			$Sf = 0.95$		
Instance	best	worst	average	best	worst	average	best	worst	average
1	328 (6)	478	373.1	331 (2)	389	356.8	338 (1)	338	338.0
2	326 (29)	486	381.8	331 (4)	331	331.0	331 (2)	331	331.0
3	327 (8)	470	373.8	327 (2)	377	350.5	331 (2)	331	331.0
4	297 (10)	454	344.0	303 (4)	303	303.0	303 (2)	303	303.0
5	319 (24)	476	374.9	324 (4)	324	324.0	324 (2)	324	324.0
6	331 (15)	486	380.1	335 (2)	339	337.0	335 (1)	339	337.0
7	312 (4)	463	357.4	340 (1)	360	351.5	342 (1)	367	354.5
8	312 (4)	463	357.0	316 (1)	363	333.8	316 (1)	318	317.0
9	310 (24)	456	366.1	314 (1)	363	331.3	314 (2)	314	314.0
10	348 (9)	488	398.0	362 (2)	378	366.5	353 (2)	353	353.0

Size (100, 70, 10) Time=40 min.									
	Algorithm 1 (12000 runs)			Algorithm 5 (4 runs)			Algorithm 5 (2 runs)		
				$Sf = 0.90$			$Sf = 0.95$		
Instance	best	worst	average	best	worst	average	best	worst	average
1	709 (1)	948	820.5	703 (1)	724	714.3	741 (1)	742	741.5
2	707 (1)	950	801.9	732 (1)	767	749.8	708 (1)	742	725.0
3	723 (1)	944	817.1	733 (1)	788	759.5	741 (1)	791	766.0
4	700 (1)	941	798.3	699 (1)	766	729.8	721 (1)	769	745.0
5	710 (1)	970	818.6	722 (1)	764	744.3	708 (1)	718	713.0
6	718 (1)	973	817.9	727 (1)	829	772.8	706 (1)	787	746.5
7	691 (1)	976	791.3	681 (1)	757	723.8	673 (1)	739	706.0
8	695 (1)	947	802.9	696 (1)	779	743.3	693 (1)	723	708.0
9	712 (1)	997	806.9	705 (1)	744	727.5	741 (1)	762	751.5
10	704 (1)	945	800.2	702 (1)	787	738.3	708 (1)	773	740.5

From the viewpoint of robustness algorithm 5 always dominates algorithm 1; from the viewpoint of quality the superiority of algorithm 5 is clear on big size instances, whereas

algorithm 1 multistarted is preferable on small instances.

One drawback of Algorithm 5 is that it requires more computing time than all the others do. Nevertheless, if a good feasible solution must be obtained quickly, the running time of Algorithm 5 (as well as for any annealing-based algorithm) can be reduced by tuning Sf to a lower value, even if this implies a degradation in the quality of the solutions: the characteristics of annealing algorithms are better exploited when a large amount of running time is available, but it is not impossible to run them as fast heuristics.

5 Conclusions

The aim of this paper is to describe a new annealing-based approach to discrete location/allocation problems, such as vertex-constrained multi-facility location problems on graphs; these problems are characterized by two sets W and X of mutually dependent variables. Only location problems that involve non-interacting facilities have been considered here; nevertheless, the method is applicable to interacting facilities as well; the only modification consists of the expression of the partial derivative $\frac{\partial f}{\partial x_j}$. The relaxation of integrality constraints on both subsets of variables and the application of the MFA technique to compute the real values of the variables at each iteration, gives origin to a double annealing algorithm, in which the usual annealing parameter T (“temperature”) is replaced by two independent parameters T_w and T_x . The behaviour of the algorithm strongly depends on the synchronization of two resulting annealing processes; for such purpose the saturation of the matrices of the allocation and location variables is used. Suitable deannealing steps are done, for forcing the two sets of variables to anneal simultaneously.

The originality of our contributions consists of the following ideas: (1) the annealing process is performed twice in parallel on two different subsets of variables, (2) the values of the annealing parameters are adapted by the algorithm itself and (3) they are not necessarily monotonically decreasing. The algorithm is also easy to program and to tune.

Future research Even if Algorithm 5 is a definite improvement over a previous more rudimentary version of the double annealing algorithm [21], further improvements are possible. Probably the most promising lines of research concern the tuning of Sf : it may be done a priori, according to the size of the variable matrices as well as a posteriori in an adaptive way, according to the observed rate of growth of saturation in each matrix. Also the use of two distinct scale factors Sf_w and Sf_x seems to be of interest and deserves investigation. The ideas of saturation-driven annealing schedule and deannealing steps may be also extended to traditional annealing algorithms.

Acknowledgements

The comments of all the referees have brought substantial improvements to the paper. I am grateful to Jan Karel Lenstra for reading a draft of this paper and giving very useful suggestions.

References

- [1] N.Christofides, A.Mingozzi, P.Toth
The Vehicle Routing Problem
in *Combinatorial Optimization*, Christofides, Mingozzi, Toth, Sandi eds.,
J.Wiley, 1979 [315-338]
- [2] L.Cooper
Heuristic methods for location-allocation problems
SIAM Review 6, 1, 1964 [37-53]
- [3] G.Cornuejols, M.L.Fisher, G.L.Nemhauser
Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms
Management Science 23, 8, 1977 [789-810]
- [4] R.D.Galvao
A dual-bounded algorithm for the P-median problem
Oper. Res. 28, 5, 1980 [1112-1121]
- [5] M.R.Garey, D.S.Johnson
Computers and Intractability: a Guide to the Theory of NP-completeness
W.H.Freeman, San Francisco, 1979
- [6] G.Y.Handler, P.B.Mirchandani
Location on networks - Theory and algorithms
MIT Press, 1979
- [7] R.Holzman
An axiomatic approach to location on networks
Mathematics of Operations Research 15,3, 1990 [553-563]
- [8] P.Jaervinen, J.Rajala, H.Sinervo
A Branch-and-Bound algorithm for seeking the p-median
Oper. Res. 20, 1972 [173-178]
- [9] J.J.Hopfield, D.W.Tank
Neural computation of decisions in optimization problems
Biological Cybernetics 52, 1985 [141-152]
- [10] O.Kariv, S.L.Hakimi
An algorithmic approach to network location problems. Part I-II
SIAM J. on Appl. Math., 37, 1979 [513-538]
- [11] S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi
Optimization by simulated annealing
Science 220, 1983 [671]

- [12] A.Kolen
Complexity of location problems on networks
BW 107/79 - Stichting Mathematisch Centrum, Amsterdam, 1979
- [13] F.Maffioli, G.Righini
Heuristics for static and dynamic multi-resource allocation problems
Workshop on "Models and Algorithms for Planning and Scheduling Problems",
Menaggio (Como), 1993
- [14] F.E.Maranzana
On the location of supply points to minimize transport costs
O.R.Quarterly 15, 1964 [261-270]
- [15] S.C.Narula, U.I.Ogbu, H.M.Samuelsson
An Algorithm for the p-median problem
Oper. Res. 25, 1977 [709-712]
- [16] C.H.Papadimitriou
Worst-case and probabilistic analysis of a geometric location problem
SIAM J. Comput. 10, 3, 1981 [542-557]
- [17] C.Peterson, B.Soederberg
A new method for mapping optimization problems on neural networks
International Journal of Neural Systems 1, 1989 [3-22]
- [18] G.Righini
Modelli di reti neurali per ottimizzazione combinatoria [Neural networks models for combinatorial optimization]
Ricerca Operativa 62, 1992 [29-67]
- [19] G.Righini
Algoritmi distribuiti per problemi di localizzazione [Distributed algorithms for location problems]
Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, 1993
- [20] G.Righini
Annealing algorithms for multi-resource absolute location problems on graph
Internal Report 93.028, Dipartimento di Elettronica, Politecnico di Milano, 1993
- [21] G.Righini
Doppio annealing per problemi di localizzazione multi-risorse su grafo [A double annealing algorithm for multi-facility location problems on graphs]
Internal Report 93.033, Dipartimento di Elettronica, Politecnico di Milano, 1993

- [22] M.B.Teitz, P.Bart
Heuristic methods for estimating the generalized vertex median of a weighted graph
Oper. Res. 16, 1968 [955-961]