Belgian Journal of Operations Research, Statistics and Computer Science Vol. 33 (1,2)

Solving real–life combinatorial optimization problems using simulated annealing

J. C. Oliveira[1,2] J. S. Ferreira[1,2] R.V.V. Vidal[3]

DEEC–Faculdade de Engenharia[1]
Universidade do Porto
4099 Porto Codex, Portugal

INESC–Instituto de Engenharia de Sistemas e Computadores[2]
Largo de Mompilher, 22, Apt. 4433
4007 Porto Codex, Portugal

IMSOR–Institute of Mathematical Statistics and Operations Research[3]
The Technical University of Denmark
2800 Lyngby, Denmark

Abstract

Two real–life optimization problems, having combinatorial properties and with the same geometric structure, will be presented. These problems are the location of afforestation areas and nesting problems. Heuristic methods based on the principles of simulated annealing have been developed to solve the above mentioned problems. Computational experiences will be reported. Emphasis will be placed on the factors that determine a suitable implementation of simulated annealing based algorithms. A similar approach could be followed to solve other types of geometrical problems such as layout and location problems, clustering problems, and image processing.

Keywords : combinatorial optimization, heuristics, simulated annealing, afforestation, nesting

1. INTRODUCTION

In this paper two real-life combinatorial optimization problems are presented: the afforestation problem and the nesting problem.

The afforestation problem (Section 3) was formulated in connection with recent discussions in Denmark and the EEC related to the optimal utilization of land. The Danish government has decided to increase the amount of forest compartments in the country. Decision making will be carried out at the county level where the planning activities have to include proposals and plans for afforestation, i.e., the location and design of new forests in a given area, taking account of the already existing forests and areas not suitable for forests.

The nesting problem naturaly arises from cutting processes in steel, wood, textile and other industries. It is a special case of the more generic class of Cutting and Packing problems, where two-dimensional irregular shapes must be positioned in a plate so that the non occupied area (the waste) is minimized. The motivation for the work presented in Section 4, comes from an innovative project, in collaboration with ANIM (Associação Nacional das Indústrias das Madeiras), a portuguese association of the wood industries, which aims at the integration of advanced technologies, namely those related with forms recognition (to automatically tackle with the wood defects and shapes), automatic cutting optimization and laser cutting procedures. These technologies will be integrated in a new technological center.

Both problems are large-scale NP-hard combinatorial optimization problems, easy to formulate verbally but very hard to solve because of the tremendous amount of feasible solutions and the existence of many local optima. But they have more in common than just their difficulty of being solved. In fact, both problems can be described as geometrical problems in which certain shapes must be located inside another larger shape: in the afforestation problem we are speaking about locating forests in a given area, while in the nesting problem the goal is, for example, to place two-dimensional pieces of furniture in a wooden plate.

Consequently, it has been decided to develop a heuristc approach, based on the simulated annealing approach, to tackle both problems. The basic structure of these heuristcs is the same. The generic decisions related with the simulated annealing algorithm implementation (the so-called cooling strategy) are the same and even the problem-specific decisions (initial solution generation and neighbourhood strutures) are rather similar. Some computational results are presented for both heuristics. The next section includes a general discussion of simulated annealing techniques.

In this paper we are emphasizing the generality of the heuristic approach, in Vidal [14] a more detailled presentation, emphasizing the particularity of each case, will be presented.

2. SIMULATED ANNEALING

The main principle of the annealing approach was first described already in 1953 by Metropolis et al. [10]. They used the approach in the simulation of energy levels in cooling solids by generating a sequence of states. This approach was applied to optimization problems independently by Kirkpatrick et al. [7] and Cerny [1] in 1983 and 1985 respectively. An extensive bibliography is presented in [2]. These authors attempt to solve difficult combinatorial optimization problems by what amounts to a combination of a Monte Carlo method and a deterministic local search. This may be thought of as an attempt to get some of the speed and reliability of descend algorithms while avoiding their tendency to stick at local optima. The homogeneous simulated annealing algorithm can be described as follows:

1. Fix initial temperature (C_o) and the number of neighbours generated at constant temperature L_o (usually call temperature length).

2. $k = 0$ (counter) and determine or fix the first initial solution ($i = i_{start}$) and its objective function value ($f(i_{start})$).

3. Repeat L_k times
   j = neighbour (i)
   if $\Delta_{ji} = f(j) - f(i) \leq 0$ then $i = j$
   else if $\exp(-\Delta_{ji}/C_k) >$ random [0,1] then $i = j$

4. $k = k + 1$, calculate L_k and C_k (reduced temperature).

5. Go to 3 until stop criterium.

This algorithm is rather similar to the local search heuristic [5], but to avoid to get trapped in a local optimum, simulated annealing allows an occasional uphill move. This is controlled by a random number generator and the temperature C_k. The core of this procedure is the loop at step 3 in the algorithm, where $\exp(-\Delta_{ji}/C_k)$ is interpreted as a probability when Δ_{ji} is positive and C_k is (always) positive. Thus this probability that an uphill move of size Δ_{ji} will be accepted diminishes as the temperature declines and, for a fixed C_k, small uphill moves have higher probabilities of acceptance than large ones.

In this general presentation of this approach several decision have to be taken before a final algorithm has been developed. Some of them are *problem-specific* : How to generate an initial solution? How to generate the neighbour to a solution? and How to calculate Δ_{ji}? The other

decisions are *generic* to the annealing process: How to fix C_o? How to determine L_k and C_k? and What is the stop criterium? These generic decisions are usually denominated a *cooling strategy* or *annealing scheme*. There are many suggestions in the available literature, see for instance [2] , to design a suitable cooling strategy. We have chosen to use the strategy that is most often utilized in connection with combinatorial optimization problems.

Choice of C_o

Choose an initial value of C_o large enough to be sure that the algorithm might leave any local optimum. Then, C_o must be comparable to the largest Δ_{ji}. If U is an upper bound to f then C_o could be larger than U, as in our cases the lower bound of f is equal to zero. On the other hand, too large a C_o value will obviously increase computation time.

Choice of L_k

We will set $L_o = L_k = L$, that is a constant. This parameter determines the number of iterations to be taken for a given temperature to reach a solution. L will be a % of the total number of feasible neighbour solutions generated for a given temperature, then there is a relation between C_k and L_k.

Choice of C_k

We have utilized the so-called exponential cooling, defined as

$$C_{k+1} = \alpha \times C_k, \qquad k = 0, 1, \ldots ,$$

$$0.8 \leq \alpha \leq 0.99$$

Stop criteria

The algorithm will stop when the change in the object function, between the solutions at the end of two temperature levels, is less than a given small figure. An alternative criterium is to fix the final values of the sequence $\{C_k\}$.

In what concerns the problem specific decisions the same strategy has been followed also for both problems. The initial solution is obtained by randomly locating the different elements (potential afforestation areas or pieces to cut) that together form a solution for the problem (the design of the new forests or the cutting layout). In both problems a raster model for the shapes is used, i.e., either the forests and the pieces are represented by a juxtaposition of squares. The neighbour solutions will be then generated by moving a randomly selected element (a square or a set of squares) in one of the four main directions (up, down, left and right) with equal probability

for all directions and a step length constant and equal to 1. The specific mechanisms will be described in more detail in sections 3 and 4, however, this way to generate a neighbour solution has some advantages:

a) it satisfies the theoretical demands that assure that equilibrium will be reach at the global optimum [7], and

b) from a programming point of view all calculations can be done very effectively because there are few floating point operations,

3. THE AFFORESTATION PROBLEM

3.1. Problem Formulation

The afforestation problem is rather easy to formulate verbally: given an area, see Figure 1, where we have disjointly areas for afforestation, areas of actual existing forests and other areas where afforestation is not recommended, it is desired to locate and design a given number of new forests, each forest having an area bigger or equal to a specific value *a*, so that the addition of the areas of these new forests is bigger or equal to another given value, *A*.

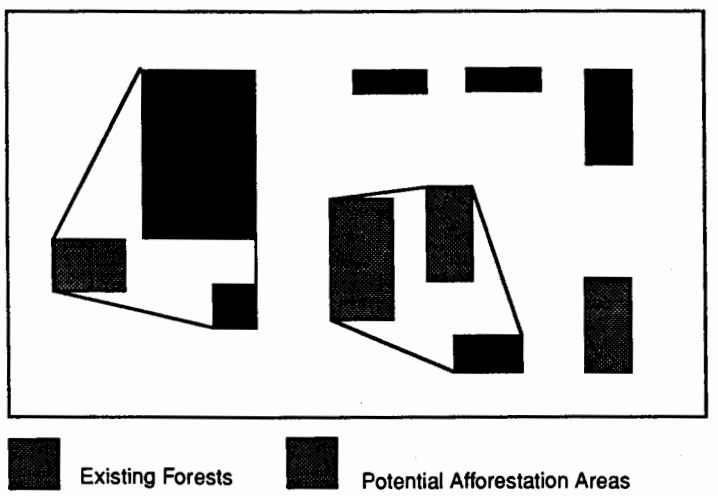

Figure 1 - Two new forests are located and designed

The area of a new forest is determined by finding the minimum convex polygon containing its respective potential afforestation and existing forest areas. The objective in the afforestation problem is to minimize the total area where afforestation is not recommended and that are included in the new forests to be located and designed, see [6] for more details about problem formulation.

We have designed a Decision Support System (DSS) to solve the above mentioned problem, see [13] for more details. The user is an experienced planner (having an education in natural sciences) that will develop afforestation plans for the decision-makers (politicians and public officials).

Using this system the planner can, with the help of a mouse, find a feasible solution to the afforestation problem and evaluate the consequences of such a solution. Using the interfaces module different types of solutions can be generated. The DSS also has a mathematical model and an algorithm that generate solutions. The interplay between the user and the model will hopefully converge to a satisfying solution, since some qualitative aspects of the problem are not included in the model and should be subjectively taken account by the experienced planner.

We had access to a large data base having a huge amount of geographical information. From this we selected relevant information, to be stored in raster form, to our own data base. This information will be used to identify areas for potential afforestation, areas of already existing forests and other areas where afforestation is not recommended (towns, lakes, highly productive agricultural land, etc.)

3.2. The Heuristic Method

A mathematical model for the afforestation problem has been constructed (see [13]). It is a combinatorial model of large scale, the number of its feasible solutions growing exponentially. An one-step approach using directly simulated annealing, was developed but the expected results were quite discouraging. Thus our first estimations of the needed computer time was of the order of 5 to 6 hours in a mainframe computer. Therefore we opted to develop a heuristic method. This method should have the following two characteristics:

a) It should produce a good solution in a realistic computational effort. Here a good solution is not just the one close to the global optimum of the mathematical model, but also one that satisfies some unquantified requirements of the user. In this sense the user should be able to modify a solution and evaluate the consequences of such modification. In our case a realistic computational effort to find a solution is of around 10 minutes. Moreover, the heuristic approach should have a good performance on the average and the chance of a very poor solution should be low.

b) Since the heuristic method will be a module in a DSS, it should somehow resemble the way how the user will solve this problem without using a mathematical model merely based in his experience. Moreover, any procedure in the approach should be easier to understand by the user, who has a background in natural sciences.

Based on the heuristic principles presented in [12] and discussions with the user, the afforestation problem, was decomposed in two sub-problems, then a two-step procedure was developed. The heuristic method has the following two main procedures:

1. Location of the "best" afforestation areas disregarding the shape and size of them. An algorithm based on simulated annealing has been developed, see further Section 3.2.1, and

2. Forests will be designed at the locations identified at step 1. An algorithm based on a construction heuristics has been developed, see further Section 3.2.2.

3.2.1. Locating Afforestation Areas

Here a number of squares of a given area will be optimally located in the total area in question. This number, for instance 20, is usually larger than the desired numbers of forests, in our case 10. The approach is based on simulated annealing as described in Section 2. As already mentioned in Section2, the values of some of the parameters used in the problem-specific decisions have to be specified.

Initial Solution

The number of squares that represent the new forests, and will be randomly located in the total area in question to form an initial solution, was 20, each one with an area approx. equal to 400 ha, the average size of a new forest.

Generation of a Neighbour Solution

As mentioned before, we have used the fact that our data is stored in a raster form, then each square will be identified by a corner and the length of a side. The neighbours generation (a random walk in the four main directions, as described in Section2) will be done parallel for all squares. In the process of the iterations two squares might converge to the same point then one of them will be randomly reallocated. It should be pointed out that:

a) Step 3 in the algorithm presented in Section 2 can be run in parallel for all squares since in principle the location of the squares are independent of each other,

b) The changes in the objective function Δ_{ji} are easily evaluated.

3.2.2. Design of the New Forests

Having "optimally" located a number of squares, say 20, then we select the "best" n squares, say n = 12, as input for this heuristic method. The best squares are those with the smallest value of f(j). Note that we select more squares than the required number of new forests, 10, this is because it might happen that during the iterations two forests could overlap each other, in such situation a single forest will be designed by joining the two overlaping forests.

The developed procedure follows the principles of a construction heuristics, as described in [12], and it is depicted in Figure 2.

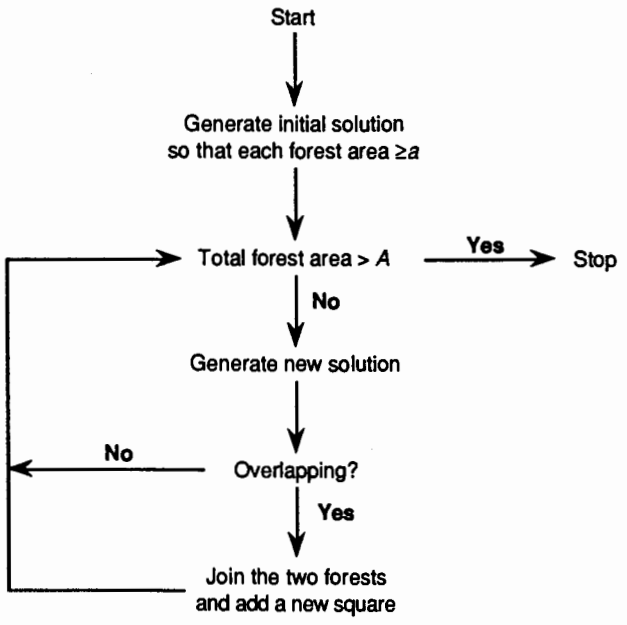

Figure 2 - The construction heuristics for the design of the new forests.

The initial and new solutions can be generated by the user or by a computer program, further details can be found in [14] where a DSS for a mainframe and a PC is presented. An element of this procedure is a program that determines the border of a new designed forest, this is done using a modification of the method presented in [8].

3.3. Experiences

The two procedures of the heuristic method were validated independently of each other.

The first part, the location heuristics, was validated by testing to what extent it was possible to generate the same results with different initial locations keeping the other parameters (cooling strategy) constant. Since this initial location was generated randomly and every random number generator needs a start number, this number, called a 'seed', was used as a varying parameter.

This approach was possible, because our extensive experimentation to find a suitable cooling strategy proved that the behaviour of the annealing algorithm was rather unsensitive to the choosen cooling strategy for different types of input data. Other researchers have published similar results [2]. Tests with one of our real cases, in which four seeds were selected, have shown that the 'best' nine squares were always located at the same place. The simulated annealing algorithm is not by all means infallible. In our study we found nonconvergence in exceptional runs. Fortunately, it was easy to see when the runs went wrong and it happened only 3 times out of 500 test runs. The replacement of seed number usually helps.

The second part, the design heuristics, was validated by the user. The parameters of the algorithm were fixed to those values acceptable for the user.

The implementation of our heuristic method demands a very strong platform, both concerning software and hardware. The method was implemented in the C-language, running under UNIX on a HP-300 workstation. As user interface was utilized X-WINDOW's environment.

Good solutions could be obtained in relatively short time (10 minutes) but if more time was avaiable, better solutions could be generated on the average.

4. THE NESTING PROBLEM

4.1. Problem Description

The nesting problem is a special case of the more generic class of Cutting and Packing problems (see [3] or [4]). In a Cutting or Packing problem big objects are to be divided in smaller ones of fixed dimensions, according with some goals. When only two dimensions of the objects are relevant for the problem, the big object is rectangular (a plate) and the smaller ones have irregular (non-rectangular shapes), then we have a nesting problem. This problem appears in the context of various production processes, such as those in the wood, furniture or textile industries. It is a rather difficult problem to formulate and there are no known approaches that

directly use any mathematical model in its resolution, instead interactive computer graphics approaches are usualy used. In this section we propose a simulated annealing based algorithm that automatically generates a layout of the plate. Excepting previous unpublished work of the authors [11] we are not aware of any publication dealing with this type of approach to the nesting problem.

4.2. The Nesting Algorithm

The algorithm to be presented assumes a raster representation of the pieces (Figure 3).

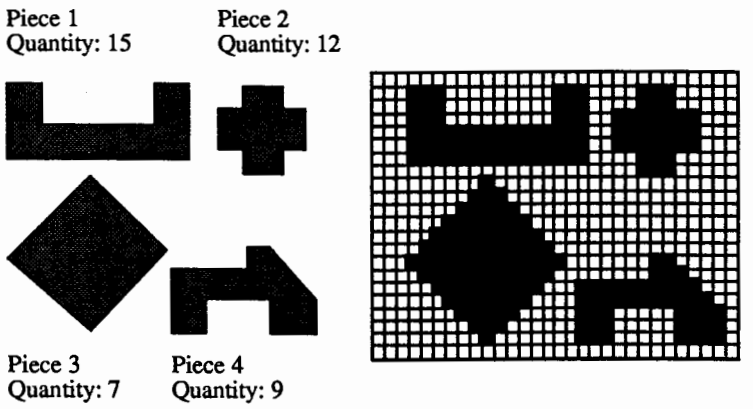

Figure 3 - A discrete model of the pieces.

The pieces in discrete form will be randomly placed over the plate and the simulated annealing based algorithm will try to minimize the pieces overlap by moving them over the plate, following the rules of neighbour generation presented in Section 2. If a movement decreases the overlap then the layout generated by that movement is accepted as the current solution, otherwise the layout may be accepted or not, with a certain probability. Although in this paper only problems in which the pieces have fixed orientations are considered, the algorithm can be easly extended to tackle problems where rotation is acepted.

An initial solution, obtained by the pieces random location mechanism, is presented in Figure 4.

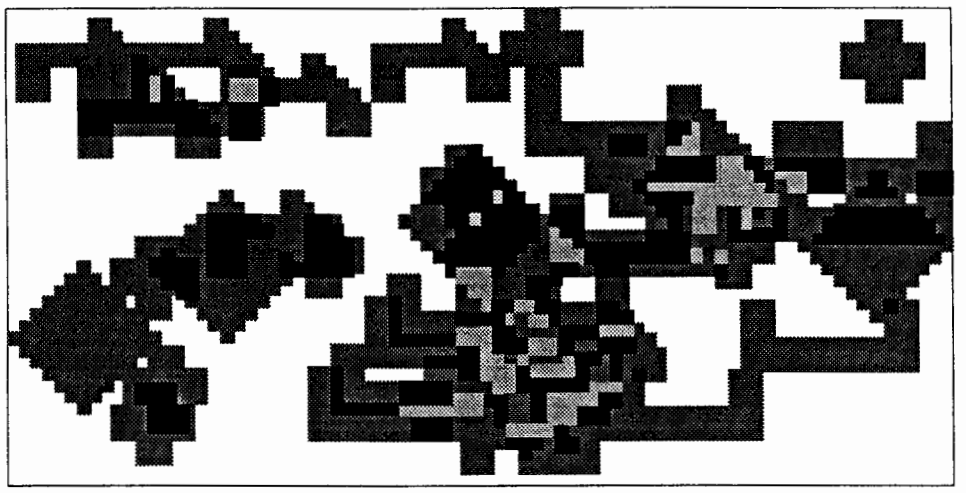

Figure 4 - Initial solution

Calculation of Δ_{ji}

For each solution *j*, the objective function *f(j)* is calculated by computing the overlap variation between solution *i* and solution *j*. This can be easly done by subtracting the overlap originated by the piece in its old position and adding it again, but now computed for its new location.

4.3. Implementation and Computational Results

The discrete models of the plate and pieces have been implemented by matrices, in which each matrix element corresponds to a grid unit. The pieces matrices are filled with 0s and 1s: 1 if the piece exists in the correspondent grid unit and 0 if not. The plate matrix is initialized with zeros and the pieces matrices are summed to it in the proper positions. If a matrix element is equal to *m* that means that *m* pieces are overlaping in the correspondent grid position.

Following the guidelines presented in Section 2, for the problem shown in Figure 3, the following values for the simulated annealing parameters have been choosen:

- $C_0 = 20$ (initial temperature)
- $L = 500$ (temperature length)
- $\alpha = 0.95$ (cooling rate)

The initial temperature was choosen so that the probability of a worst solution being accepted was, at least, of 50%. As mentioned before, that was achieved by considering an upper bound for the variation of the objective function. Considering the neighbourhood structure used (pieces moving one grid unit in orthogonal directions) an upper bound for the overlap variation that any movement can produce is the larger dimension of the biggest piece. For the problem used in the computational tests this upper bound was 15 what gives the following expression for the initial temperature calculation:

$$0.5 \approx \exp -(15 / C_0)$$

Tunning the cooling rate was only possible by experimentation. Cooling slowly assures the algorithm convergence but takes a lot of time. Cooling too fast speeds up the algorithm but may conduce to a poor solution. After some experiments $\alpha = 0.95$ showed up as a good compromise between time consumption and quality of results (Figure 5).

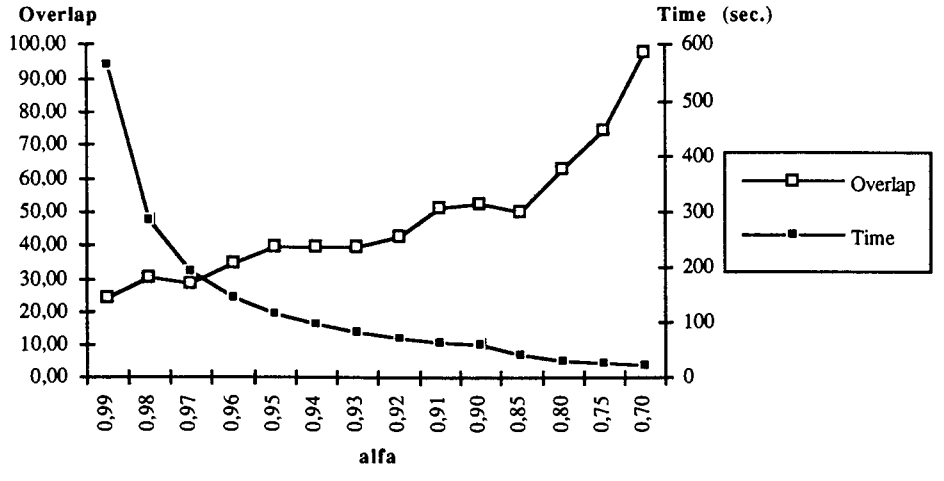

Figure 5 - Final overlap and execution time versus cooling rate (average values for 10 runs)

In what concerns the temperature lengths, L = 500 was used. This means relatively short temperature lengths, but this value corresponds (for the problems solved) to three times the size of the neighbourhood of each solution (4 possible directions for each movement x 43 pieces to move). The opposite approach, fast cooling with long temperature chains, was tried but the results were clearly worse.

The computational results presented in Table 1 concern the problem of Figure 3 (with a 80x40 plate). They were obtained after 10 runs of the algorithm, using a personal computer with the 80486 processor running at 33 MHz.

Table 1 - Computational results

	Overlap	Time	Number of iterations
Average	40.0	1' 57.7"	45 255.1
Standard-deviation	11.4	2.3"	969.9

It is clear that the algorithm converges to good solutions (with a low overlap) in a reasonable amount of time. In the 250 test runs executed the algorithm has always converged. One of these final solutions is presented in Figure 6. However, to obtain a solution for the nesting problem, a layout without overlap must be generated, what may be achieved by iteratively applying this algorithm with successively increased plate lengths, until the final layout overlap is zero.

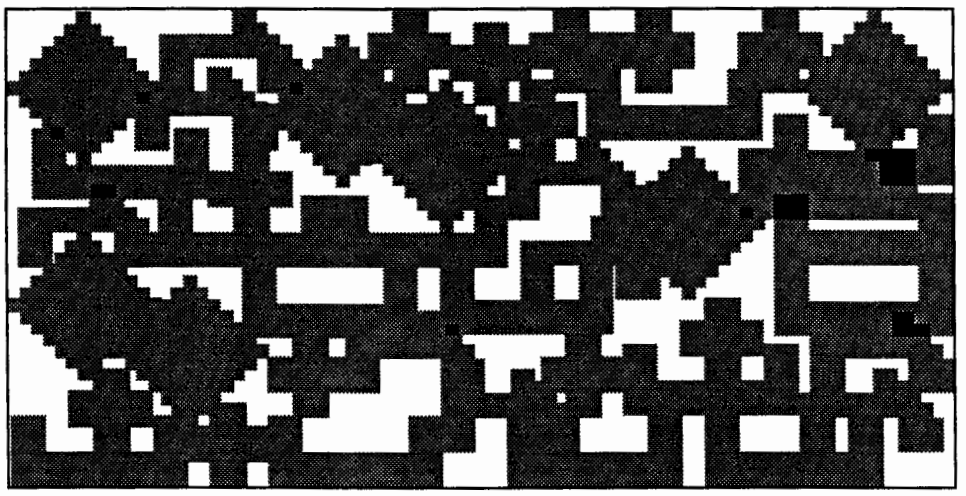

Figure 6 - Final layout

The discrete model approach is not an exact approach in the sense that it cannot represent without error an arbitrary irregular shape. This is clearly shown in Figure 3, namely in the 45 degrees lines. If the precision of the discrete model is increased (by decreasing the size of the grid unit) the execution times become larger. However for the cases in which the shapes are rectangles or a juxtaposition of rectangles, this becomes an exact approach. Another version of our approach where the pieces are represented as exact models (polygonial shapes) will be presented in [14].

5. CONCLUSIONS

Two real-life optimization problems, the afforestation problem and the nesting problem, with common characteristics have been presented. These problems have the same structure and heuristics based on a similar simulated annealing implementation (generic and problem-specific decisions) have been developed allowing for the generation of solutions for these problems. These are not necessarily optimal, but quite satisfactory, specially if we consider that they are large-scale problems where classic combinatorial optimization techniques may not be able to give any solution in an acceptable period of time.

The developed method has several attractive features. Firstly, it is very easy to implement, although it may demand some effort to tune up the simulated annealing parameters. Secondly, it is flexible and therefore applicable to different types of problems. In fact, other geometric problems, as the layout, location, clustering problems and imaging processing, could be handled in a similar way. Thirdly, it is very suitable as a tool to be included in a DSS for interactively problem solving.

REFERENCES

[1] Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem: an efficient simulation algorithm", Journal of Optimization Theory and Applications 45, 1985, pp 44-51.

[2] Collins, N.E., Eglese, R.W., and Golden, B.L., "Simulated Annealing . an annotated bibliography", American Journal of Mathematical and Management Sciences 8, 1988, pp 209-307.

[3] Dagli, C. and Tatoglu, M., "An Approach to Two-Dimensional Cutting and Packing Problems", International Journal on Production Research 25, 1987, pp 175-190.

[4] Dyckhoff, H., "A Typology of Cutting and Packing Problems", European Journal of Operational Research 44, 1990, pp 145-159.

[5] Evans, J.R., "Structural Analysis of Local Search Heuristics in Combinatorial Optimization", Computers and Operations Research 14, 1987, pp 465-477.

[6] Jorgensen, R.M. and Thomsen, H., "Areal Allokering: Konstruktion af et Beslutningsstottesystem", MSc Thesis nr. 1/90, IMSOR, Technical University of Denmark, 1990 (in Danish).

[7] Kirkpatrick, S., et al, "Optimization by Simulated Annealing", Science 220, 1983, pp 1087-1092.

[8] Kurozumi, Y. and Davies W.A., "Polygonial Approximations by the Minimax Method", Computer Graphics and Image Processing 19, 1982, pp 248-264.

[9] Lundy, M. and Mees, A., "Convergence of an Annealing Algorithm", Mathematical Programming 34, 1986, pp 11- 124.

[10] Metropolis, N., et al., "Equation of State Calculation by Fast Computing Machines", Journal of Chemical Physics 21, 1953, pp 1087-1092.

[11] Oliveira, J. and Ferreira J., "A Simulated Annealing Algorithm for Nesting Problems", INESC - Technical Report, 1992.

[12] Silver, E.A., et al, "A Tutorial on Heuristic Methods", European Journal of Operational Research 5, 1980, pp 153-162.

[13] Vidal, R.V.V., " The Afforestation Problem: A Heuristic Method Based on Simulated Annealing", EJOR 56, 1992, pp 184-191.

[14] Vidal, R.V.V. (Ed.), " Applied Simulated Annealing", 1993 , Lecture Notes in Economics and Mathematical Systems, Springer-Verlag.