A Study of Heuristic Techniques Inspired in Natural Process for the Solution of the Container Fill Problem

M. Delgado Pineda, J.M. De Pedro Salinas, and J. Aranda Almansa

Universidad de Educación a Distancia,
E.T.S.I. Informática

Abstract. This article shows some techniques based on simulated annealing and genetic alghoritms for the resolution of a filling problem of a container of two dimensions using rectangular pieces of sizes not congruent. This problem is quite related to problems like bin-packing or strip-packing. The comparative was made using several type problems and having into account parameters like run time, number of solutions that converge to the optimum one and quality of the found non-optimum solutions.

1 Introduction

Most problems found in industry and government are either computationally intractable by their nature, or sufficiently large so as to preclude the use of exact algorithms. In such cases, heuristic methods are usually employed to find good, but not necessarily guaranteed optimum solutions.

The effectiveness of these methods depends upon their ability to adapt to a particular realization, avoid to go into local optima, and exploit the basic structure of the problem, such as a network or a natural ordering among its components.

Furthermore, restart procedures, controlled randomization, efficient data structures, and preprocessing are also beneficial. Building on these notions, various heuristic search techniques have been developed that have demonstrably improved our ability to obtain good solutions to difficult combinatorial optimization problems. The most promising of such techniques include simulated annealing [10], tabu search [8], ant colony optimization [11], genetic algorithms [9] and GRASP (Greedy Randomized Adaptive Search Procedures) [7].

In this article is set out the problem of filling a container in its bi-dimensional version, which consists of locating on a rectangular surface of fixed length and width, also rectangular pieces of a certain set, so that it does not find overlapping nor spillage and that the occupied space is the maximum (the totality of the surface, if possible).

There are two potential approaches to the problem depending on the possibility or not to be able to apply turns to the pieces, only the problem without turns was treated. This problem is included within the packing and cutting problems, problems widely studied due to its importance in certain industrial and transport sectors [1] [3]. This kind of problems belongs to the class of non-polynomial problems (NP), therefore an exact solution in a reasonable time cannot be given, in spite of having algorithms like the comprehensive search that could find this solution.

The heuristic techniques acquire a great importance in the solution of these other similar problems since they provide relatively good solutions in an acceptable time. The problems are much more complex in practice, such as the problem of cutting nonrectangular pieces of different forms, but this characteristic increases the complexity, therefore the comparison is restricted to the so-called rectangular problems, that is to say, those in which the objects to pack or to cut have a rectangular form. This paper showns two simulated annealing algorithm and one genetic algorithm for the solution of this kind of problem.

2 Simulated Annealing

Simulated annealing is a generalization of a Monte Carlo method for examining the equations of state and frozen states of n-body systems [6]. The concept is based on the manner in which liquids freeze or metals re-crystallize in the process of annealing. In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state at $T = 0$. Hence the process can be thought of as an adiabatic approach to the lowest energy state. If the initial temperature of the system is too low or cooling is done insufficiently slowly the system may become quenched forming defects or freezing out in meta-stable states (ie. trapped in a local minimum energy state).

The original Metropolis scheme was that an initial state of a thermodynamic system was chosen at energy E and temperature T, holding T constant the initial configuration is perturbed and the change in energy dE is computed.

If the change in energy is negative the new configuration is accepted. If the change in energy is positive it is accepted with a probability given by the Boltzmann factor $e^{-\frac{dE}{T}}$. This processes is then repeated sufficient times to give good sampling statistics for the current temperature, and then the temperature is decremented and the entire process repeated until a frozen state is achieved at $T = 0$.

The connection between this algorithm and mathematical minimization was first noted by Pincus, but it was Kirkpatrick [10] who proposed that it form the basis of an optimization technique for combinatorial (and other) problems. The current state of the thermodynamic system is analogous to the current solution to the combinatorial problem, the energy equation for the thermodynamic system is analogous to at the objective function, and ground state is analogous to the global minimum. The major difficulty (art) in implementation of the algorithm is that there is no obvious analogy for the temperature T with respect to a free parameter in the combinatorial problem. Furthermore, avoidance of entrainment in local minima (quenching) is dependent on the "annealing schedule", the choice of initial temperature, how many iterations are performed at each temperature, and how much the temperature is decremented at each step as cooling proceeds.

Two alternatives for the solution by means of techniques based on simulated annealing appear next: The first is based on the original model of Metropolis, the second supposes that a search for each temperature is done.

2.1 Algorithm of Metropolis

The laws of the thermodynamics say that to a temperature T the probability of a power increase of magnitude dE can be approximated by $P[dE] = e^{-\frac{dE}{kT}}$, where k is a physical constant denominated Boltzmann

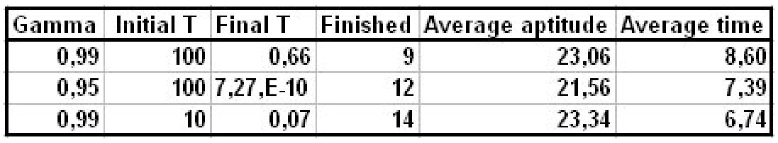
In the model of Metropolis a random disturbance in the system is generated and the resulting changes of energy are calculated: if there is a power fall, the change is accepted automatically; on the contrary, if a power increase takes place, the change was accepted with a probability indicated by the previous expression. The idea consists of beginning with a high temperature and to be decreasing it very slowly until a base state is reached, in our case the temperature is only a parameter of control of the probability of accepting a worsening in the solution, nevertheless we used this nomenclature to state the similarity with the physical process that serves as base to our algorithm. The code of the algorithm of Metropolis would be.

```
$S = Initial\_solution : T = Initialize : Gamma = Initialize$
REPEAT
$S^* = M(S)$
IF $Eval(S^*) < Eval(S)$ THEN $S = S^*$ ELSE
    IF $Random(0,1) < exp((Eval(S) - Eval(S^*))/(kT))$ THEN $S = S^*$
    ELSE $S = S$
    $T = T * Gamma$
UNTIL SHUTDOWN CRITERION.
```

where T is the temperature, k the constant of Boltzman, this constant appears in the natural processes of crystallization but in this case it is only another parameter of control which will take the value $k = 1$, *Gamma* is a constant that takes a value between 0 and 1 and it is used to diminish T, imitating in this form the cooling of the system and $M()$ a function that generates a new solution by means of a small disturbance, for example, it could be the function that was in the previous section.

This type of algorithms, allows with a certain probability, that tends to zero when the number of iterations tends to infinite, the worsening in the evaluation of the solutions, which will allow us to leave possible local minimums.

Three groups of tests was made, changing the different parameters of the algorithm to try to see which ones are better adapted to the treated problems. In this first experiment the algorithm was applied 100 times with 500 iterations each, the initial temperature, 100, and a *Gamma* = 0.99. In the second ex-



Gamma	Initial T	Final T	Finished	Average aptitude	Average time
0,99	100	0,66	9	23,06	8,60
0,95	100	7,27,E-10	12	21,56	7,39
0,99	10	0,07	14	23,34	6,74

Fig. 1. Comparison of different parameters in Algorithm of Metropolis

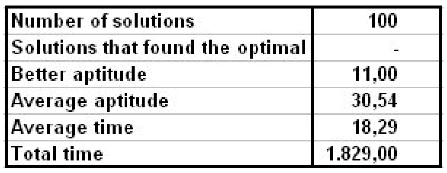
periment the algorithm was applied 100 times with 500 iterations each, thesame initial temperature and a *Gamma* = 0.95. In the third experiment the algorithm was applied 100 times with 500 iterations each, the initial temperature, 10, and a *Gamma* = 0.99. If a comparative of the results of the three experiments is made, several conclusions can be drawn: The best results were obtained in the third experiment, nevertheless, the best average aptitude was obtained in the second, this is due to the formula of acceptance of solutions

```
IF $Eval(S^*) < Eval(S)$ THEN $S = S^*$
ELSE
    IF $Random(0,1) < exp((Eval(S) - Eval(S*))/(kT))$ THEN $S = S^*$
    ELSE $S = S$
```

As the temperature faster decreases and lower is, also the probability of allowing a worsening of the solution is smaller, for this reason, in experiment two the temperature descends quickly then exists a convergence towards local minimums and therefore the average aptitude is smaller, in experiment one the temperature descends very slowly and at any moment there is possibility of accepting a solution of worse aptitude, for that reason, the average aptitude is worse. In case three the initial temperature smaller, the stabilization of the solutions takes place before, but as the temperature descends very slowly it does not lose the possibility of leaving global minimums until the end. With the second problem several groups of tests was made.

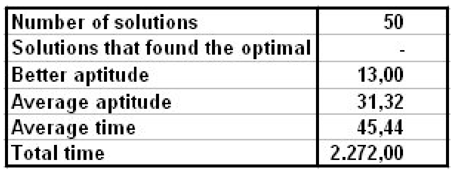
In the first experiment the algorithm was applied 100 times with 500 iterations each, the initial temperature, 10 and a *Gamma* = 0.99. In the second experiment the algorithm was applied 50 times with 1000 iterations each, the initial temperature, 10000, and a *Gamma* = 0.99.As we can observe the results in the first experiment have been better than in the second, as much in time as in average aptitude, for that reason we make a third experiment with 1000 iterations and the parameters of the first experiment adapted to the number of iterations, that is to say, the initial temperature must be of 1515 so that the final one agrees with the one of the first experiment. In the third experiment the algorithm was applied 100 times with 1000 iterations each, then initial temperature, 1515, and *Gamma* = 0.99.

As it is observed there exists a light improvement of the average aptitude as well as of the best found aptitude. Taking as a base the algorithms of metropolis, in the early 80s Kirkpatrick and Cerny developed algorithms for the design of circuits VLSI and the resolution of TSP (Traveling Salesman Problem) and showed how they could be applied to combinatorial optimization problems, like



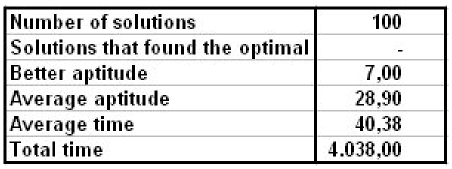
Number of solutions	100
Solutions that found the optimal	-
Better aptitude	11,00
Average aptitude	30,54
Average time	18,29
Total time	1.829,00

Fig. 2. Experiment 1: Results for case 2



Number of solutions	50
Solutions that found the optimal	-
Better aptitude	13,00
Average aptitude	31,32
Average time	45,44
Total time	2.272,00

Fig. 3. Experiment 2: Results for case 2



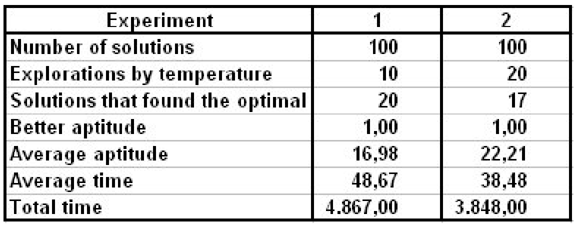
Number of solutions	100
Solutions that found the optimal	-
Better aptitude	7,00
Average aptitude	28,90
Average time	40,38
Total time	4.038,00

Fig. 4. Experiment 3: Results for case 2

the one which occupies us, the algorithm that they developed was termed Simulated Annealing [2], which is the new algorithm that was treated.

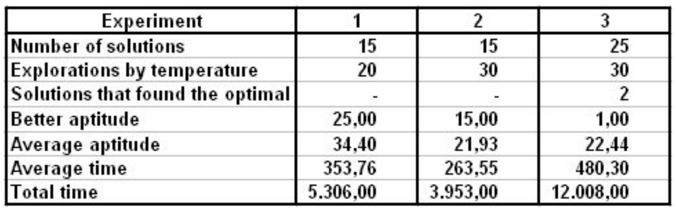
A Simulated Annealing Algorithm Alternative. The basic difference between the former algorithm and the following one is that it search for each temperature [1] [2], which is translated in the following code.

```
$S = Initial\_solution : T = Initialize : Gamma = Initialize$
$Niter = Number$ of iterations that we wished to be produced
REPEAT
        REPEAT
        $S^* = M(S)$
        IF $Eval(S^*) < Eval(S)$ THEN $S = S^*$
        ELSE
                IF $Random(0,1) < exp(Eval(S) - Eval(S^*))/T)$ THEN $S = S^*$
                ELSE $S = S$
        UNTIL NUMBER OF ITERATIONS = $Niter$
        T= T * Gamma
UNTIL SHUTDOWN CRITERION.
```



Experiment	1	2
Number of solutions	100	100
Explorations by temperature	10	20
Solutions that found the optimal	20	17
Better aptitude	1,00	1,00
Average aptitude	16,98	22,21
Average time	48,67	38,48
Total time	4.867,00	3.848,00

Fig. 5. Results SA for case 1



Experiment	1	2	3
Number of solutions	15	15	25
Explorations by temperature	20	30	30
Solutions that found the optimal	-	-	2
Better aptitude	25,00	15,00	1,00
Average aptitude	34,40	21,93	22,44
Average time	353,76	263,55	480,30
Total time	5.306,00	3.953,00	12.008,00

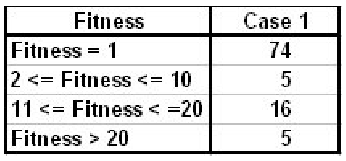
Fig. 6. Results SA for case 2

This type of algorithms, allows with a certain probability, that tends to zero when the number of iterations tends to infinite, the worsening in the evaluation of the solutions, which will allow us to leave possible local minimums. Unlike the previous algorithm, this algorithm makes an exploration for each temperature, which increases the time necessary for its execution but also the probability of finding the global optimum. Two groups of tests was made, changing the different parameters of the algorithm to try to see which ones are better adapted to our problems: In the first experiment the algorithm was applied with (Times, Iterations, Initial Temperature, *Gamma*)=(100, 500 , 100, $0,99$). These parameters were the best ones in the previous section. $Niter$ = 10, that is that ten iterations was madee for each power state (for each temperature).

In the second experiment the algorithm was applied with (100, 200, 100, $0,99$). These parameters were the best ones in the Algorithm of Metropolis, with. $Niter = 20$.

Three groups of tests was made with the second problem changing the different parameters of the algorithm to try to see which ones are better adapted to this problem.

- First experiment. The algorithm was applied with (15, 500, 1515, 0.99). $Niter = 20$.
- Second experiment. The algorithm was applied with (15, 200 ,, 10000, 0.95). $Niter = 30$.
- Third experiment. The algorithm was applied with (25, 500 , 10,0.99). $Niter$ $= 30$.



Fitness	Case 1
Fitness = 1	74
2 <= Fitness <= 10	5
11 <= Fitness < =20	16
Fitness > 20	5

Fig. 7. Results GA for case 1

It is observed that the results of the third experiment are substantially better than those of first and second experiment, nevertheless, the necessary time was greater also. It seems logical to think that the parameters used in the third case are more appropriate that the two firsts, in the first case the final temperature was of 9.95, a too high temperature that allows worsening with too much probability and therefore exists a excessively random behavior, in the second experiment this final temperature was of 0.35, therefore, the algorithm tends to allow worsening less and less and therefore the search can be considered less random. An improvement in the average aptitude exists and the time was even smaller, nevertheless, the global optimum was not found. The results of third experiment were quite good, the global optimum was found, the 13,33% of the solutions had a satisfactory convergence, however, the necessary time was greater than in the other experiments. Another interesting point is the increase in the number of explorations for each power

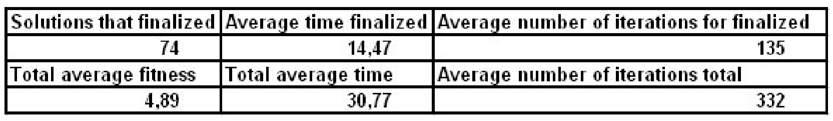
3 Genetic Algorithms

The Genetic Algorithms (AGs) introduced by John Holland in 1970 [4] are techniques of optimization that are based on concepts like the natural and genetic selection. In this method the variables are represented like genes in a chromosome [4].

In this case the representation of the solutions is the already presented, the initial possible solutions population was generated of randomly, the evaluation function was the free space, the crossing operator was PMX that along with BLF is the one that presents better results [5], the selection was made by means of the roulette method and a mutation was a transposition of two elements of the solution.. A series of tests of 100 executions was made with the first problem and the following parameters:

(Recombination, Mutation, Initial solution, Number of solution, [Number of iterations with solution, Number of iterations without solution])

– Recombination: The recombination was by pairs using operator PMX and passing to the following generation the children.
– Mutations: The mutations was by change, and its probability was of 3%.
– Initial solutions: Random.



Solutions that finalized	Average time finalized	Average number of iterations for finalized
74	14,47	135
Total average fitness	Total average time	Average number of iterations total
4,89	30,77	332

Fig. 8. Results GA for case 1

– Number of solutions: 20.
– Number of iterations: 1000 or to find the optimum one.

(PMX & passing the childen, 3%, Random, 20, [1000, _])

The total time of execution was of 3, 077 seconds, that is , 51 minutes, to speed up the tests we will include a new criterion of shutdown, if in 100 iterations the best fitness found does not experience improvement we finalized the experiment.

Five experiments with 100 executions was made with the following parameters:

1. (PMX & passing the childen, 3%, Random, 20, [1000, 1000]).
2. (PMX & passing the childen, 5%, Random, 20, [1000, 100]).
3. (PMX & passing the childen, 3%, Random, 20, [1000, 100]).
   We calculate the recombination probability by adding 20 to each one of the fitness of the solutions, to equal the probabilities and avoid a premature convergence to local minimums.

$$\begin{aligned} SF &= \sum_i (Fit(S_i) + 20); \; S_iA = \tfrac{SF}{Fit(S_i)}; \\ SFI &= \sum_i S_iA; \qquad P(S_i) = \tfrac{Fit(S_i)+20}{SFI}. \end{aligned}$$

4. (PMX & passing the childen, 3%, Random, 20, [1000, 100]).
   We calculate the recombination probability by elevating to the fourth power the original probability, this favors the propagation of the solutions with better fitness, but also the danger to fall in local minimums.

$$\begin{aligned} SF &= \sum_i Fit(S_i); \; S_iA = \left(\tfrac{SF}{Fit(S_i)}\right)^4; \\ SFI &= \sum_i S_iA; \quad P(S_i) = \tfrac{Fit(S_i)+20}{SFI}. \end{aligned}$$

5. (PMX & passing the best one, 3%, Random, 20, [1000, 100]).

The best experiment was the first one for the time used and for the results obtained. We did not put the criterion of shutdown of 100 iterations without improvement, obtained many more convergences, 74% as opposed to 49%, which represents a 51% of convergences, nevertheless, the used time went as 3077 seconds opposed to 636, which supposes a 384% but time.

4 Conclusions

This kind of problems is very treatable with techniques based on simulated annealing. The genetic algorithms present initially the problem of the suitable election of the representation of the solutions.

The genetic algorithms present good results because a certain intermediate solution contains a sub-chain that comprises the optimal solution, and when crossing several of these solutions those sub-chains are prospering and joining themselves until obtaining the optimum one, nevertheless, in this problem this is not fulfilled since what is evaluated is the space occupied within the container, however, in the crossing of the genetic algorithm the arrangements of pieces are crossed.

Remark: All experiments were made in a computer with Pentium IV, CPU 2,4 GHz, 522,228 KB of RAM, and. the code of programs were Visual BASIC code.

References

1. E. Hopper and B. C. H. Turton, An empirical investigation of Mete-heuristic and Heuristic algorithms for a 2D packing problem, E.J.O.R 128/1,pp 34-57, (2000)..
2. K. A. Dowsland, Heuristic desirn and fundamentals of the Simulated Annealing, Revista Iberoamericana de Inteligencia Artificial. N 19 (2003), pp 93-102..
3. A. Gómez, Resolución de problemas de packing en una empresa de calcamonias mediante algoritmos geneticos, XII congreso nacional y VII congreso Hispano-Francs de AEDEM, (1998), pp 999-1008.
4. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Velag, (1994).
5. A. Gómez, Resolución del problema de Strip-Packing mediante la metaheurstica algoritmos genticos, Boletn de la SEIO (2000), pp. 12-16.
6. N. Metrópolis A.W. Rosenbluth, M.N. Rosenbluth, and A.H. Teller. Equation of state calculations by fast computing machines. (1953). The Journal of Chemical Physics, vol. 21, no. 6, pp. 1087-1091
7. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109-133. 1995.
8. F. Glover and M. Laguna. Taby search. Kluwer Academic Publisher, 1997.
9. D.E. Goldber. Genetic algorithm in search, optimization and machine learning. Addison-Wesley, 1989.
10. S. Kirkpatrick. Optimization by simulate annealing: quantitative studies. Journal of Statistical Physics, 34:975-986. 1984.
11. M. Dorigo and T. Sttzle. The ant colony optimization metaheuristic: Algorithms, applications, and advances, 2001. in Handbook of Metaheuristics, F. Glover and G. Kochenberger.