Journal of Computer Science 5 (12): 974-979, 2009
ISSN 1549-3636


# Simulated Annealing with Deterministic Decisions

Taisir Eldos
Department of Computer Engineering,
Jordan University of Science and Technology, Al Ramtha, Irbid Jordan

**Abstract: Problem statement:** Simulated Annealing (SA) algorithms have been used in solving a wide range of discrete optimization problems for many years, with well know drawbacks like the computational time and difficulties related to the parameters settings. One of the other issues that open the door for research is the acceptance decision that provides for hill climbing; the standard SA algorithms use a stochastic method which fails to justify the acceptance of a cost increasing solutions while rejecting mildly cost increasing ones. **Approach:** To resolve this dilemma, the reversible deformation mechanism we developed earlier replaced the stochastic decision with a deterministic one; by deforming the problem structure and gradually reforming it towards the original one. This provides for hill climbing in the real domain while applying a simple downhill search in the virtual sense. Unlike the standard SA algorithm, the number of iterations must be known in advance and it is the only stopping criteria, because the scaling functions parameters are selected based on the number of iterations. **Results:** This method had produced better solutions and the new enhancement to the algorithm improves the overall performance by examining each state more thoroughly through a set of perturbations and thus securing a move towards a better neighborhood, the same set of tests used in the original methods are repeated for comparison. **Conclusion:** The significance of this research comes from eliminating the unpredictability of the stochastic decisions in the standard SA algorithms which might yield less than acceptable solutions in some cases.

**Key words**: Evolutionary, optimization, simulated, annealing, deterministic, algorithms

## INTRODUCTION

Complex problems have been solved by approximation using a set of methodologies; Evolutionary Algorithms (EAs) which refer to a class of computational problem-solving algorithms inspired by the principle of natural selection; Genetic Algorithms (GAs) by Holland in 1975[10], as an adaptation of the well known survival of the fittest principle and the Simulated Annealing (SA) and its variants invented by Kirkpatrick in 1983[11] as an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by Metropolis in 1953[12]. Since then, many variants and closely related works have been introduced, but the SA seems to have been well explored and not much is added lately, while the EA has sibling like the Particle Swarm Algorithms (PSAs). In all cases, the basic idea is to start with a solution or more and evolve towards more fit ones hoping for an optimal or semi-optimal solution. Generally, the SA algorithm is a local search that allows uphill moves with stochastically controlled acceptance. Although it has been used in many fields for years, it is slow by nature and its parameters setting have no rigorous justification.

In an early study, we proposed an algorithm that mimics the SA behavior except in the solution acceptance; it replaces the stochastic acceptance with a deterministic mechanism. The algorithm performs the search as a downhill procedure in the virtual sense that is capable of climbing hills in the real domain. Mapping the problem from the real sense to the virtual sense is carried out through a reversible deformation mechanism that is problem dependent.

A key to this decision mechanism is the reversible deformation; a constructive distortion that provides for real hill climbing through virtual local search. The randomized acceptance of cost increasing configurations in simulated annealing and possibly all other evolutionary algorithms is meant to enhance the exploratory power of the search through, but may fail to achieve its purpose because of its pure randomized decision. The proposed mechanism eliminates this drawback through deterministic decisions. This approach is close to some extent to the threshold acceptance approach, which accepts cost increasing

solutions with some limit above the current cost and this threshold is gradually decreased towards the end. Although it is a general technique, it requires the problem nature to lend itself to the reversible structuring; we applied it to the cell placement problem to show that it outperforms the standard simulated annealing. However, it can be applied to similar optimization problems like the traveling salesman problem, where the distance matrix is scaled up to the largest distance between any two cities to compute the factor matrix.

**Related work:** Research over the last two decades tried to overcome the challenges that faced the SA algorithms and reduce its computational time and increase its chance of producing an optimal solution. Parallel SA and Hybrid SA-GA[1] and others addressed the computational complexity, while other variants addressed the certain aspects related to the settings and schedules. Threshold acceptance algorithms[2] for example accept a solution that increases the cost by an amount less than a threshold; this threshold is decreased by time to limit the rate of acceptance of cost increasing solutions. However, the threshold and its rate of reduction are yet to be tuned and there exists no rules for optimal settings. Another variation is the old bachelor[3] algorithm, which is similar to the threshold acceptance algorithm, except that if a solution is not accepted the threshold is increased. The degraded ceiling algorithm[4] is yet another variant with absolute bound threshold that decreases by time.

In all cases, finding the temperature values and thresholds are tedious and tests have shown that a variant may perform well on certain class of problems but not as a general procedure. However, the statistical promise of finding the globally optimal solution is considered an important feature of SA; this ensures a uniform sampling of the search space, which is reassuring when little is known about the nature of the space. Attempts to speed up SA, such as Simulated Quenching (SQ), usually trade this promise off with the gain in efficiency[5]. Many researchers have found it very attractive to take advantage of the ease of coding and implementing SA, utilizing its ability to handle quite complex cost functions and constraints. However, the long time of execution of standard Boltzmann-type SA has many times driven these projects to utilize a temperature schedule too fast to satisfy the sufficiency conditions required to establish a true (weak) ergodic search[6].

A logarithmic temperature schedule is consistent with the Boltzmann algorithm, e.g., the temperature schedule is taken to be with expediency the only reason given. While perhaps someday less stringent necessary conditions may be developed for the Boltzmann algorithm, this is not now the state of affairs[7].

Guofang and colleagues[8] used an adaptive simulated annealing, in which they considered the characters of different circuits to be placed and revealed its advantages in placement results and time performance when compared with the traditional simulated annealing algorithm.

The primary criticism to the simulated annealing is that it is too slow, another criticisms is that the algorithm is too broadly based on physical intuition and is too short on mathematical rigor, as a matter of fact some researchers gave their own calculations to demonstrate that SA could be a very poor algorithm to search for global optima in some instances. The other criticisms may be considered by some to be more subjective, but they are likely no more extreme than the use of SQ to solve for global optima under the protective umbrella of SA.

The threshold based solutions have parameters that are hard to select or control and a major drawback of ignoring the problem details; for example in a cell placement problem the large and small cells are dealt with equally, while the small cells are easier to move and should be given this advantage

As a way out, we propose an algorithm that eliminates the temperature based probabilistic acceptance of negative transitions and uses a deterministic mechanism for such decisions instead. However, unlike the threshold acceptance and the degraded ceiling algorithms, it is executed as a downhill search and accepts only positive transitions in the virtual domain. The algorithm starts by scaling up all the dimensions in a process called deformation, which emulates a melting space and scales down using a certain schedule, a process called reformation; which brings the problems dimensions back to normal, mimicking the freezing point.

The RDA algorithm has shown better performance compared to the standard SA algorithm[9]. However, a new enhancement towards even better permanence is devised and tested using the same benchmark, using higher performance machines.

## MATERIALS AND METHODS

**Implementation:** The new algorithm depends on deforming the structure at the beginning and reforming it during the search process. The proposed algorithm is applicable to a wide range of problems, but we will discuss its implementation with cell placement problem of the VLSI design as a case study.

There are three parameters to be defined here; the number of iterations N, the deformation array; set of factors $\alpha_{wi}$ and $\alpha_{di}$ for each cell and the maximum cell dimensions $w_{max}$ and $d_{max}$. N is typically a large number that represents the number of iterations to complete the search and $N_s = \beta*N$ is the number of iterations during which the scaling down completes, where the rest is carried out in local search mode. The largest dimensions are used to deform the structure by scaling up the width and depth of each cell as starting dimension.

The $i^{th}$ cell dimensions in the $j^{th}$ iterations are:

$$w_{ij} = (\alpha_{wi}*\alpha_{wi}* \ldots *\alpha_{wi})*w_{max} = (\alpha_{wi})^{j}*w_{max} \quad (1)$$

$$d_{ij} = (\alpha_{di}*\alpha_{di}* \ldots *\alpha_{di})*d_{max} = (\alpha_{di})^{j}*d_{max} \quad (2)$$

Hence, the dimension scaling factors for each cell are computed as follows:

$$\alpha_{wi} = (w_{i0}/w_{max})^{1/\beta*N} \quad (3)$$

$$\alpha_{di} = (d_{i0}/d_{max})^{1/\beta*N} \quad (4)$$

The outline below shows the standard SA steps, without the details of stopping criteria and perturbation methods that make transition from one solution to a neighboring solution.

**SA algorithm outline:**

1 i = 0;
2 GET($T_m$); Initial temperature
3 GET($S_0$); Random initial solution
4 GET(S'); Neighbor S'$\in$N($S_i$)
5 $\Delta_i$ = C(S')-C($S_i$); Evaluate cost difference
6 $P_x = \exp(-\Delta_i/T_i)$; Acceptance probability
7 $P_y$ = GET(P); Random number (0,1)
8 $P_y<P_x$:$S_{i+1}$ = S'; Accept/reject solution
9 $T_i$ = NEXT($T_i$); Adjust temperature
10 $T_i>T_f$:GOTO4; Stopping criteria

The stopping criterion may be a time budget, a fixed number of iterations, reaching a freezing temperature, or relative improvement below a certain threshold or a certain number of iterations without getting a cost decreasing solution.

Initial and freezing temperatures can be computed by using the initial solution cost along with selected probabilities for acceptance at the beginning and at the end. Let $P_m$ and $P_f$ be the probabilities of accepting a cost increase of 25% using the initial cost as reference, then.

If the initial solution cost is $C_i$ then the melting and freezing temperatures are $T_m$ and $T_f$:

$$P_m = \exp(-0.25*C_i/T_m) \quad (5)$$

$$T_m = -0.25*C_i/\ln(P_m) \quad (6)$$

and:

$$P_f = \exp(-0.25*C_i/T_f) \quad (7)$$

$$T_f = -0.25*C_i/\ln(P_f) \quad (8)$$

For $P_m = 0.999$ and $P_f = 0.001$, we get:

$T_m = 250*C_i$ and $T_f = 0.036*C_i$

The proposed algorithm allows a local search algorithm to behave like a simulated annealing in the sense that it accepts cost increasing solution within limits defined by the scaling function, although it looks like a local search that accepts only cost decreasing solutions. The advantage of this method is it limits the randomization to the neighborhood generation and replaces the temperature schedule with a scaling schedule, which is more predictable in terms of time budget allocation, as it dictates the number of iterations. It also provides more realistic acceptance as it is proportional to the actual cost function, in which some moves are unfairly accepted or rejected.

The algorithm uses two functions DEFORM and REFORM. The first is used only once at the beginning to restructure the problem, while the second is used to gradually bring back the structure to its real form RDA algorithm outline.

**RDA algorithm outline:**

1 i = 0;
2 DEFORM; Scale up
3 GET($S_0$); Random initial solution
4 i<$N_s$: REFORM; Scale down
5 GET(S'); Neighbor S'$\epsilon$N($S_i$)
6 C(S')<C($S_i$):$S_{i+1}$ = S'; Accept if a better one
7 i< N: GOTO 4; Stopping criteria

Typically, the REFORM process scales down the structure slowly enough by selecting large a value for N and to avoid the rare chance of not getting out a trap of local minimum; we reverse the last REFORM step if a certain number of trails carried out in sequence with no

acceptance. The initialization strategy could have a crucial influence on the performance; especially when the search space is disconnected. So, we randomly look for an initial solution that satisfies a certain constraint; area that is less than twice the algebraic sum of the cells in our case.

As the outline of the RDAe shows; rather than moving through the scale down process regardless of the accept/reject decision, we keep trying perturbation until a successful move is achieved. This increases the computational time in a limited number of trials but the overall increase in time is quite small.

**RDAe algorithm outline:**

| | | |
|---|---|---|
| 8 | i = 0; | |
| 9 | DEFORM; | Scale up |
| 10 | GET($S_0$); | Random initial solution |
| 11 | i<$N_s$: REFORM; | Scale down |
| 12 | LOOP; | Forceful Find of |
| | {; | virtually better Solution |
| 13 | GET(S'); | Neighbor S'$\in$ N($S_i$) |
| | } C(S')>C($S_i$); | Loop and accept only a |
| 14 | $S_{i+1}$ = S'; | better or equal one |
| 15 | i<N: GOTO 4; | Stopping criteria |

We used a set of similar workstations in a lab to carry out 10 rounds per test, to compare the standard SA with the proposed RDA and RDAe, in terms few factors:

- Score; a measure of probability of finding a solution with quality exceeding a certain value in a fixed amount of time
- Minimum, maximum and average quality for several runs with nearly same time budget
- Time to find a solution with quality higher than a certain value

Figure 1 shows a sequence of transitions at different scales, it shows how virtual local search path climbs the hill in the real sense, the inner surfaces represent the cost surfaces over time; the doubled solid line on the top represents the initial cost of the sequence after the DEFORM step while the doubled solid line at the bottom represents the real cost surface over the same sequence and the dashed lines in-between represent the cost surface at different scaling levels.

Figure 2 shows a detailed snapshot of the two paths; the real (by standard SA) and the virtual (by RDA and RDAe) with a focus on the transitions between two sets of iterations; from 41-41 and from 42-43.

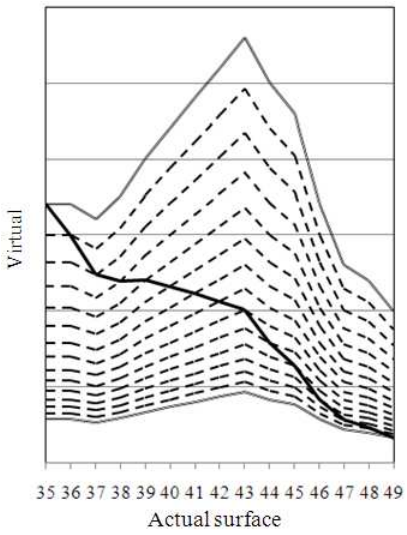


Fig. 1: Path: Virtual Vs actual surface

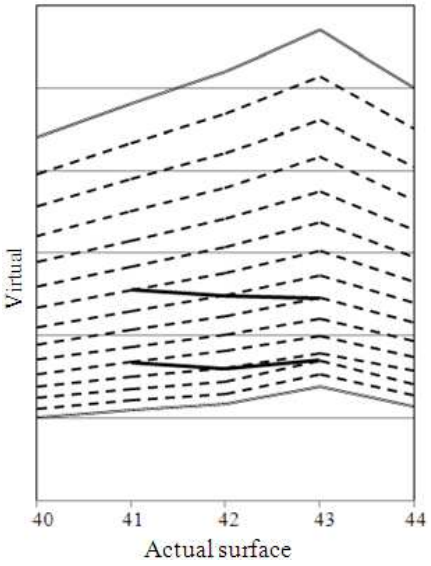


Fig. 2: Path: Virtual Vs actual surface

The first transition is a cost decreasing and hence accepted whether taken early in time (the upper segment) or late in time (the lower segment). On the other hand, the transition from 42-43 would be taken in the early stage and rejected in the late one.

## RESULTS AND DISCUSSION

Figure 3 and 4 show snapshots of the three paths towards the end of the search; RDA and RDAe are always along a cost decreasing path while the simulated running on the real problem.

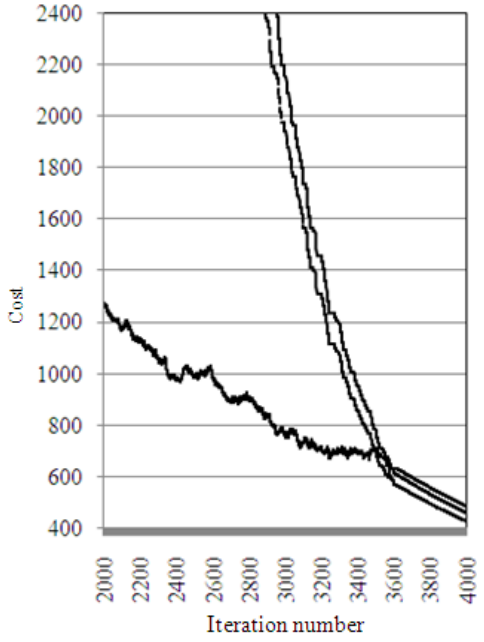


Fig. 3: Behaviors, cost Vs iteration snapshot

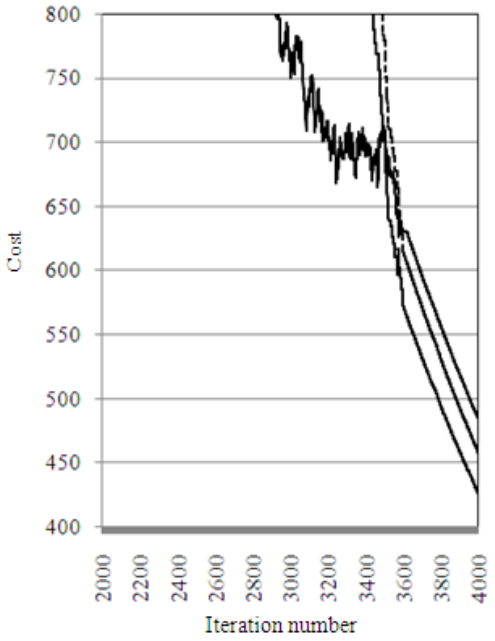


Fig. 4: Behaviors, cost Vs iteration snapshot

To compare the performance of the RDAe with the RDA and the standard SA, we measured the time required to perform 100 trials on each and selected the number of iterations such that they execute to completion in nearly the same amount of time. We used faster computers in this research compared to the previous one[9] to carry out the measurements and hence the results are nearly scaled down in time by around 30% and the statistical nature of the moves prevents the exact reproductions of the old results too.

Table 1: Performance, waste and time (ami 33, 857)

| | SA | RDA | RDAe |
|---|---|---|---|
| Min (%) | 9.50 | 5.60 | 5.30 |
| Max (%) | 11.50 | 6.20 | 6.10 |
| Mean (%) | 10.30 | 5.80 | 5.50 |
| Time (h) | 2.95 | 3.06 | 3.17 |

Table 2: Performance, waste and time (ami 49, 1598)

| | SA | RDA | RDAe |
|---|---|---|---|
| Min (%) | 13.6 | 6.30 | 5.90 |
| Max (%) | 14.4 | 7.60 | 7.10 |
| Mean (%) | 13.8 | 6.90 | 6.50 |
| Time (h) | 7.08 | 7.32 | 7.48 |

Table 3: Performance, score Vs tolerance (ami 15, 400)

| | Score of 10 rounds (20 min) | | |
|---|---|---|---|
| Target (%) | SA | RDA | RDAe |
| 0 | 1 | 2 | 3 |
| 2 | 2 | 3 | 3 |
| 4 | 3 | 4 | 5 |
| 6 | 3 | 5 | 6 |
| 8 | 4 | 5 | 7 |
| 10 | 5 | 6 | 7 |
| 12 | 5 | 7 | 7 |
| 14 | 5 | 7 | 8 |
| 16 | 6 | 8 | 8 |

Table 4: Performance, time Vs tolerance (ami 15, 400)

| | Average run time (min) | | |
|---|---|---|---|
| Target (%) | SA | RDA | RDAe |
| 0 | 32.6 | 35.7 | 37.5 |
| 2 | 30.5 | 33.5 | 35.3 |
| 4 | 29.5 | 32.6 | 33.8 |
| 6 | 27.9 | 28.2 | 30.8 |
| 8 | 27.3 | 27.6 | 30.0 |
| 10 | 26.4 | 26.9 | 29.3 |
| 12 | 25.8 | 26.5 | 28.5 |
| 14 | 25.2 | 26.3 | 27.8 |
| 16 | 24.5 | 25.3 | 26.1 |

Using the benchmark test of 33 cells and 10 rounds per algorithm, Table 1 shows that the RDAe beats the standard in the min, max and mean dead area, the worst case is even better than the best case of the SA and the time is only slightly more. And the enhanced version RDAe is slightly better than RDA in almost all cases.

A comparison between the three algorithms on a larger problem of 49 cells is shown in Table 2 using 10 rounds per algorithm. The results are consistent regardless of the problem size.

Table 3 depicts the performance of the proposed algorithm compared to the others in terms of possibility of achieving its goal; how many of 10 rounds will get a solutions of predefined quality within a fixed time.

Table 4 shows a comparative performance in terms the time to achieve a certain requirement, a dead space less than certain percent of the algebraic sum of cells areas. The RDAe takes little more time due to its continuous use of the REFORM function. The better score of RDAe justifies this extra time.

## CONCLUSION

Standard Simulated Annealing (SA) and many of its variants are quite sensitive to the initial conditions and hence may get stuck at local optima, while the deterministic acceptance of cost increasing solutions of the Reversible Deformation Annealing (RDA) seems to reduce the sensitivity to the initial conditions, hence increase the likelihood of optimality. In addition, the RDAe outperformed the RDA which has proven to be better than the standard SA in every single run in terms of the quality of the solution when run for the same amount of time including the few trials tested on the same initial condition. The RDA has only a small price of increased time due to the iterative down scaling and the RDAe add to that little price a bit more due to the several transitions in a limited number of iterations to enhance the acceptance rate. The results are encouraging to extend to similarly problems that lend themselves to reversible deformation, like traveling salesman and binary knapsack.

## REFERENCES


1. Calaor, A.E., A.Y. Hermosilla and B.O. Corpus Jr., 2002. Parallel hybrid adventures with simulated annealing and genetic algorithms. Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, (PAAN'02), IEEE Computer Society, Washington DC., USA., pp: 33-38. http://doi.ieeecomputersociety.org/10.1109/ISPAN.2002.1004258
2. Dueck, G. and T. Scheuer, 1990. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. J. Comput. Phys., 90: 161-175. http://dx.doi.org/10.1016/0021-9991(90)90201-B
3. Hu, T.C., A.B. Kahng and C.W. Tsao, 1995. Old bachelor acceptance: A new class of non-monotone threshold accepting methods. ORSA Journal on Computing, 7: 417-425. DOI: 10.1287/ijoc.7.4.417
4. Burke, E.K., Y. Bykov, J.P. Newall and S. Petrovic, 2004. A time-predefined local search approach to exam time tabling problems. IIE Trans., 36: 509-528. DOI: 10.1080/07408170490438410
5. Desai, R. and R. Pateil, 1996. Combining simulated annealing and local optimization for efficient global optimization. Proceedings of the 9th Florida AI Research Symposium, June 1996, The Florida Artificial Intelligence Research Society, Key West, FL., pp: 233-237. ISBN: 0-9629-1738-8.
6. Perumal, A.I. and S.P. Rajagopalan, 2007. Adaptive simulated annealing: Useful lesson learned. Int. J. Soft Comput., 2: 572-579.
7. Ingber, L., 1996. Adaptive Simulated Annealing (ASA): Lessons learned. Control Cybernet., 25: 33-54.
8. Guofang Nan, Minqiang Li, Sanlin and Jisong Kou, 2005, Adaptive simulated annealing for standard cell placement. Adv. Neural Comput., 3612: 943-947. DOI: 10.1007/11539902_117
9. Eldos, T. and R, Qasim, 2009. Reversible deformation annealing: A new variant of simulated annealing. Proceeding of the International Conference on Genetic and Evolutionary Methods, July, 13-16, WorldComp, Las Vegas, Nevada, US., pp: 270-274.
10. Holland, J., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbour, ISBN: 10: 0-262-58111-6.
11. Kirkpatrick, S., C.D. Gelatt Jr. and M.P. Vecchi, 1983. Optimization by simulated annealing. Science, 220: 671-680. DOI: 10.1126/science.220.4598.671
12. Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth and A.H. Teller, 1953. Equation of state calculations by fast computing machines. J. Chem. Phys., 21: 1087. DOI: 10.1063/1.1699114