Fast Optimization by Demon Algorithms

Ian Wood and Tom Downs
Neural Network Laboratory, Department of Electrical and Computer Engineering
University of Queensland, St. Lucia 4072. Australia
{wood, td}@elec.uq.edu.au

ABSTRACT

We introduce four new general optimization algorithms based on the 'demon' algorithm from statistical physics and the simulated annealing (SA) optimization method. These algorithms use a computationally simpler acceptance function, but can use any SA annealing schedule or move generation function. Computation per trial is significantly reduced. The algorithms are tested on traveling salesman problems including Grotschel's 442-city problem and the results are comparable to those produced using SA. Applications to the Boltzmann machine are considered.

1. Introduction

We present here a number of optimization algorithms based on the simulated annealing (SA) method. These new methods aim to speed up SA by reducing computation time per trial without sacrificing the quality of solutions. The choice of parameters is kept fairly simple, and applicability to other variations of SA is maintained.

The initial motivation for this study came from an interest in improving the speed of the Boltzmann machine - a recurrent neural net model [1] which requires Gibbs sampling of its internal states at a low 'temperature' equilibrium for both its learning and operational phases. Attainment of a low-temperature equilibrium has been achieved in the past via simulated annealing but is slow enough to deter most people from using the model. As a means of improving the speed of the Boltzmann machine one might consider speeding up both the approach to equilibrium and the rate at which sampling can occur.

Fast Gibbs sampling of equilibria is also important in computational statistical physics. One approach due to Creutz [2] is aimed at the 2-D Ising model of atomic spins in a ferromagnetic lattice. Conventionally, this is simulated using the Metropolis algorithm [3], but Creutz found he could use a computationally simpler 'demon' algorithm to achieve similar results in far less time.

In its original form the demon algorithm does not aim to generate low energy states, and hence is not directly useful for optimization. Optimization problems can usually be framed in terms of a cost or energy function which is to be minimized over a space of possible solutions. Here we propose four algorithms which vary the operation of the demon algorithm to encourage it to search for optimal solutions. The methods are tested on 200- and 442-city traveling salesman problems and results on the latter are compared with those reported using other similarly general optimization algorithms.

2. The Metropolis Algorithm

The Metropolis algorithm was invented to allow computer simulation of equilibria in statistical physics. An initial state and a temperature are specified, and a Markov chain of system states is generated. Once equilibrium is reached at the required temperature, associated quantities can be approximated from the chain of states. The algorithm can be stated as follows.

1. choose an initial configuration (state) S
2. choose a temperature $T > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$, where $E(S)$ is the energy of configuration S
   (c) if $\Delta E < 0$ accept new configuration, ie: $S = S'$
   (d) else if $rand[0,1] \leq \exp(-\Delta E/T)$ accept new configuration, ie: $S = S'$
   (e) else reject new configuration
4. until stop_condition

The simulation would normally only be stopped once the user has enough samples to calculate equilibrium properties to a desired accuracy.

The method of choosing the next state is called the generating function. Each new system state or configuration should be a small stochastic perturbation of the current state. For discrete parameters, such as those present in the Boltzmann machine,

the generating function is usually a uniform random distribution over the neighbouring states.

New states are accepted according to an acceptance function which depends on the difference in energy between the current state and the proposed state. The Metropolis algorithm accepts any state transition which will reduce the system energy, and accepts increases stochastically using the function in 3(d).

3. Creutz's Demon Algorithm

Creutz's original demon algorithm can be stated as:

1. choose an initial configuration S
2. choose a demon energy $D > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq D$ accept new configuration and update demon, ie: $S = S',\ D = D - \Delta E$
   (d) else reject new configuration
4. until stop_condition

In this algorithm, the energy lost by the system is given to an artificial variable called a 'demon'. Increases in system energy are only allowed if the demon can provide the necessary energy, which it then loses. As a result, total system energy is a constant : $E(S) + D = C$, for any state in the Markov chain. Temperature is not specified directly, but can be estimated from the chain of states. Its value is clearly governed by the total energy C, which is set at the energy of the initial state plus the initial demon energy.

The acceptance function for Creutz's method is deterministic and computationally simpler than that of the Metropolis algorithm. It replaces an exponentiation and the generation of a random number with a comparison and a subtraction. The sequence of states produced remains stochastic, but derives this from the generating function.

4. Simulated Annealing

Kirkpatrick et al [4] altered the Metropolis algorithm for optimization by making it specifically aim for low energy states, whilst retaining its ability to escape local minima by the occasional acceptance of moves which increase system energy. The only difference between simulated annealing and the Metropolis algorithm is the addition of a scheduling step:

3(f) if quasi-equilibrium reached, reduce temperature according to schedule.

Kirkpatrick's original annealing schedule was to set

$$T(n) = \alpha\ T(n-1) \qquad (1)$$

$\alpha \in (0,1)$ where n is the number of times annealing has been applied. This negative exponential (or geometric) schedule is quite commonly used in applications and has produced good results ([4], [5]).

Simulated annealing is seen to consist of three procedures: a move generating function, a move acceptance function and an annealing schedule. Many variations on the original generating function (eg:[6]) and annealing schedule (eg:[5]) have been suggested. Far less effort seems to have gone into the acceptance function. Other papers that attempt a similar simplification of the acceptance function include [7], [8], [9].

5. Demon Algorithms for Optimization

Here we have altered Creutz's algorithm to guide us from an initial state towards lower energy states. This is done by gradually removing energy from the demon in the following ways:

- 'annealing' the demon value, much as Kirkpatrick et al [4] and others have annealed the temperature in simulated annealing.
- imposing a fairly low upper bound on the demon, to truncate its value regularly.

The two above methods can each be improved by introducing a stochastic demon value, which is normally distributed around a mean. The demon mean then operates in a similar manner to the demon value in the deterministic demon methods. The stochastic demon will occasionally take on high values allowing the system to escape from local minima that it might otherwise have been heavily delayed or trapped by. However, the additional randomness increases the computational cost of the methods.

The procedure for the bounded demon algorithm is shown below:

Bounded Demon Algorithm

1. choose an initial configuration S
2. choose an initial demon energy $D = D_0 > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq D$ accept new configuration and update demon, ie: $S = S', D = D - \Delta E$
   (d) else reject new configuration
   (e) if $D > D_0$, $D = D_0$ - enforce demon upper bound
4. until stop_condition

The annealed demon algorithm replaces step 3(e) with an annealing step:

3(e) if quasi-equilibrium reached, reduce demon according to schedule, eg: $D = \alpha * D$

A randomized version of each of the bounded and annealed demon algorithms can be obtained by replacing D, the demon energy, with D_m in steps 2 & 3(e) and adding a step before 3(c) to generate a demon value from a distribution centered around this mean:

3(b)(ii) $D = D_m$ + noise value.

3(c) is also changed slightly, so that although the demon value is checked for acceptance, its mean is updated:

3(c) if $\Delta E \leq D$ accept new configuration and update demon mean, ie: $S = S', D_m = D_m - \Delta E$.

The additive noise has mean 0 and variance specified by the user as a fraction of the Dm_0 value. This adds a stochastic element to the acceptance calculation and allows rare large increases in energy, ruled out by the deterministic algorithms.

It should be possible to combine these algorithms with any of the alternative generating functions (eg: FSA [6]) or annealing schedules (eg: polynomial [5]) that have been proposed.

6. Computational Complexity

Table: 1: Algorithm Complexity - Acceptance Function

Algorithm	Operations	Time
Metropolis Alg	m,e,r	24
Creutz' Demon	a,c	2
SA	m,e,r	24
Bounded Demon	a,c	2
R Bounded Demon	a,c,3m,e,r	32
Annealed Demon	a,c	2
R Annealed Demon	a,c,3m,e,r	32
Greedy	c	1
TA	c	1

Table 1 compare the computational complexity of the algorithms in the worst case - the acceptance of a new configuration which increases the system energy. Operations are classed as a (addition and subtraction), c (compare), m (multiplication and division), e (exponentiation) and r (random number generation). Relative computation time for these calculations is approximately: a,c - 1; m - 3; r - 5; e - 16. The generating function is common to all these algorithms and uses $2 \times r + 3 \times a$ for a time of 13 units.

The annealing operation is only performed on average every N trials, where N is likely to be over 100.

It requires a multiplication in negative exponential schedules and an addition in linear schedules. The generating function requires two random numbers to be generated in all cases.

7. Traveling Salesman Problems

As a test of the capabilities of the new algorithm, we chose the Traveling Salesman Problem (TSP). For large numbers of cities, this class of problems is recognized as being difficult to solve using general combinatorial optimization algorithms [10]. It has been widely studied and published results exist for many optimization techniques on a range of TSP instances. Also, global optima are known for some large problem instances which allows a more absolute evaluation of the performance of the various algorithms.

The algorithms tested included the four demon algorithms, as well as standard simulated annealing [4] and a greedy algorithm which only accepts improvements in the cost function.

All algorithms were allowed to run to a maximum of 10^7 trials, and simulations were stopped before that if the number of consecutive rejected moves exceeded 50,000. Only a little effort was made to choose suitable values for the 'user-defined' parameters.

As a representative example, we show results on one instance of a 200-city TSP, averaged over 5 runs each starting from a random initial tour. The city coordinates were chosen from a uniform random distribution over a 10×10 grid. The move generation rule used was uniform 2-opt [11], or segment reversal.

A paper by Dueck and Scheuer [7] using an algorithm resembling the annealed demon algorithm contained the results of detailed testing on Grotschel's 442-city problem [12]. This paper also quotes results on these two TSPs by Rossier et al [13] using exhaustive Lin-2-opt and simulated annealing, and Muhlenbein et al [14] using genetic algorithms.

The data for this problem and many others is available in the TSPLIB archive at [15].

8. Dueck and Scheuer's work

Dueck and Scheuer's [7] main algorithm is known as Threshold Accepting (TA). It uses a threshold term that operates somewhat like the demon parameter in the demon algorithms. Choosing an initial value for the threshold is somewhat different because the threshold does not dynamically reduce. The threshold is reduced over time via a linear or hand-optimized annealing schedule and quasi-equilibrium is not considered. Given these differences, the algorithm pseudo-code is identical to that of the annealed demon algorithm, except that in 3(c), the threshold or demon value is not altered:

3(c) if $\Delta E \leq D$ accept new configuration, ie: $S = S'$.

The principal differences between this algorithm and the annealed demon algorithm are:

- the threshold does not absorb and release energy, unlike the demon.
- the only annealing schedules considered are linear or a problem-specific variation of this.
- the upper bound on individual energy increases is fixed
- unlimited hill-climbing is possible, allowing eventual escape from any deep local minima, as well as unconstrained wandering.

Of the demon algorithms, only the two randomized algorithms allow the possibility of unlimited hill-climbing.

The demon value in the the bounded demon algorithms plays much the same role as the threshold in TA. However the average demon value is usually much lower than the demon upper bound, so this bound can be set much higher than the threshold in TA while maintaining a similar rate of energy reduction. The demon value's variation allows the algorithm to occasionally accept state transitions involving much larger increases in energy than can be allowed under TA.

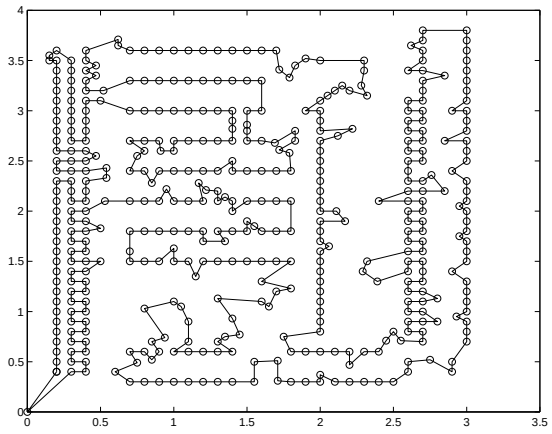

Fig. 1: An optimal solution to Grotschel's 442-city TSP

9. Results

All of the new algorithms require the choice of an initial demon value. This choice is quite important for the bounded demon algorithms since it is also an annealing control. It is much less important for the annealed demon algorithm, which has separate parameters (such as α in eqn. 1) for annealing.

However, we have generally found that all the algorithms were less sensitive to the choice of these parameters than simulated annealing was to the choice of α. Other parameters, such as those involved in the determining of quasi-equilibrium, were chosen as advised in [5].

9.1. Random scatter 200-city problem

Table: 2: 200-city TSP results

Algorithm	Average	Best	Trials
SA	106.28	104.10	1.09 M
Annealed Demon	104.71	103.52	4.22 M
Bounded Demon	106.15	105.02	10 M
R Annealed Demon	103.75	102.95	7 M
R Bounded Demon	105.66	105.30	10 M
Greedy	115.20	112.85	0.15 M

The fact that some of the algorithms terminated well short of the maximum number of trials is some cause for concern, since it is expected that more trials would lead to better results. The following tests showed that a more careful choice of parameters could result in runs of any desired length, and correspondingly better results.

9.2. Grotschel's 442 city problem

Fig 1 shows an optimal solution to Grotschel's 442-city problem [12]. The optimal tour has a length of 50.78 units. Results on this problem from [13] using simulated annealing and from [14] using genetic algorithms are summarized in Table 3. Both report only their best results. Note that the Lin-2-opt procedure was allowed to run until no further improvement was possible.

Rossier et al [13] introduced a 'Distance' heuristic for the problem, which requires that the two cities chosen for consideration of a 2-opt move must lie within a .45 radius of each other. This is the maximum distance to a neighbour for any of the 442 cities in Grotschel's problem. Cities in this problem have on average around 20 neighbours within this radius, and examination of the optimal solution (fig. 1) shows that only one distance along the route exceeds .45, indicating the likely usefulness of this problem-specific heuristic.

Dueck and Scheuer [7] use this heuristic extensively, and with good results. We show results from our own simulations of the demon algorithms and simulated annealing along with those from [7] (threshold accepting) with the 'Distance' heuristic and without on the 442-city problem. Each line contains the average and best results over 25 random starting tours.

The SA and demon algorithm simulations were done with fairly careful selection of the user parameters. For instance, the negative exponential schedule (eqn. 1) is governed by the parameter α. Although many texts, eg: [5], suggest choosing α in the range .85 .. .99., we found that values around

Table: 3: Simulated Annealing and Genetic Algorithms

Algorithm	Best result	Trials
Lin-2-opt	57.30	unknown
SA Standard	53.30	2 M
SA Distance	51.765	2 M
Genetic Algorithm	51.21	unknown

Table: 4: Algorithms using 2-opt - 2 M trials

Algorithm	Average	Best
Simulated Annealing	54.35	53.34
Bounded Demon	53.69	52.28
R Bounded Demon	53.97	53.48
Annealed Demon	54.73	53.62
R Annealed Demon	54.44	53.16
Greedy 0.8 M	57.20	55.74
Threshold Accepting	52.96	51.94

Table: 5: Algorithms using 2-opt and Distance - 2 M trials

Algorithm	Average	Best
Simulated Annealing	51.72	51.23
Bounded Demon	52.24	51.60
R Bounded Demon	52.36	51.77
Annealed Demon	51.74	51.19
R Annealed Demon	51.75	51.26
Threshold Accepting	51.51	50.97

.9996 were much more successful for SA at 2 M trials. These reduced the system temperature from an initial value, commonly 20, to a final value of .01. This was low enough to ensure the rejection of most energy increasing moves, meaning the system was unlikely to evolve further.

10. Discussion

Dueck and Scheuer [7] admit that their 'annealing' schedule was optimized for Grotschel's TSP. The values chosen are not far removed from a linear annealing schedule, which would require only one parameter - the initial value. However, they choose 30 values for the threshold, each apparently held for 1/30 th of the total number of trials. This can be considered as the choosing of 30 parameters. They report obtaining similar results using a linear schedule, but no details are given.

Our exponential annealing schedule requires two parameters - the initial demon value and α (eqn. 1). A linear schedule is also possible, but has not yet been tried. We believe that an annealing schedule requiring any more parameter choices places an unnecessary burden on the user.

11. Conclusion

We have presented here four new optimization algorithms which differ from simulated annealing in their acceptance function. The bounded and annealed demon algorithms use a deterministic acceptance rule, which varies dynamically, and is computationally simpler than simulated annealing. Results on Grotschel's 442-city TSP are comparable with SA and related algorithms over the same number of trials. We now intend to apply one of the optimizing demon algorithms to the Boltzmann machine in place of simulated annealing to speed up its equilibration.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985. [Reprinted in Anderson.Rosenfeld.88].

[2] M. Creutz, "Microcanonical Monte Carlo simulation," *Physical Review Letters*, vol. 50, no. 19, pp. 1411–1414, 1983.

[3] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[4] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[5] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*. Chichester: Wiley, 1989.

[6] H. Szu and R. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, pp. 157–162, 1987.

[7] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, pp. 161–175, 1990.

[8] P. Moscato and J. Fontanari, "Stochastic versus deterministic update in simulated annealing," *Physics Letters A*, vol. 146, no. 4, pp. 204–208, 1990.

[9] H. Guo, M. Zuckermann, R. Harris, and M. Grant, "A fast algorithm for simulated annealing," *Physica Scripta*, vol. T38, pp. 40–44, 1991.

[10] G. Reinelt, *The Traveling Salesman - Computational Solutions for TSP Applications*. Berlin: Springer-Verlag, 1995.

[11] S. Lin, "Computer solutions of the traveling salesman problem," *The Bell System Technical Journal*, vol. 44, pp. 2245–2269, 1965.

[12] M. Grötshel, Preprint no. 38, Polyhedrische Kombinatorik and Schnittebenverfahren, Universitat Augsburg, Germany, 1984.

[13] Y. Rossier, R. Troyon, and T. Liebling, "Probabilistic exchange algorithms and euclidean traveling salesman problems," *OR Spektrum*, vol. 8, pp. 151–164, 1986.

[14] H. Muhlenbein, M. Gorges-Schleuter, and O. Kramer, "Evolution algorithms in combinatorial optimization," *Parallel Computing*, vol. 7, pp. 65–85, 1988.

[15] G. Reinelt, "TSPLIB.", http://softlib.rice.edu/softlib/tsplib/, June 1995.