A RELATIONSHIP BETWEEN THE METROPOLIS ALGORITHM AND THE TWO–MEMBERED EVOLUTION STRATEGY

L. Vincent Edmondson

Central Missouri State University

1. Introduction. A significant amount of research has been done during the past two decades in the area of nature–inspired heuristic algorithms. These algorithms are designed to be robust problem–solving techniques which are typically applied to difficult optimization problems (such as those found in the class of problems labeled NP- complete). The two most common "natural" heuristic algorithms are simulated annealing and genetic algorithms. This paper briefly reviews the mechanics of the algorithms and then establishes a relationship between the Metropolis algorithm [1] from simulated annealing and a special form of a genetic algorithm known as the two–membered evolution strategy.

2. Simulated Annealing and the Metropolis Algorithm. Simulated annealing is modeled after the actual annealing process in condensed matter physics. In brief, annealing is the process in which the temperature of a solid in a heat bath is increased to a point at which the particles of the solid move freely with respect to one another, followed by a slow cooling of the heat bath. If the cooling is slow enough, then the particles line themselves up and reach a state with minimum energy.

If a system is in thermal equilibrium at a given temperature T, then its energy is probabilistically distributed among all different energy states E according to the Boltzmann probability distribution

$$\mathrm{Prob}(E) \sim \exp\left(\frac{-E}{kT}\right),$$

where k is the Boltzmann constant. This means that, for any temperature T, there is a nonzero probability that the current local minimum is not the global minimum. The net effect of this is that the system can perform hillclimbing in an attempt to move from a local minimum to a better (possibly global) minimum [2,3].

The following pseudo–code form of the Metropolis algorithm incorporates the aforementioned hillclimbing strategy.

1. Generate a solution x_1 to the minimization problem and evaluate the objective function at x_1 to obtain E_1. ("Solution" simply means a valid answer to the problem and it does not imply optimality.)

2. Randomly perturb x_1 to obtain x_2 and evaluate the objective function at x_2 to obtain E_2.
3. Calculate the probability p that x_2 will become the incumbent solution.

$$p = \exp\left[\frac{-(E_2 - E_1)}{kT}\right]$$

   If $p > 1$, then $p \leftarrow 1$.
4. Determine if x_2 will become the incumbent solution. Assume that random $[0, 1)$ generates a uniformly–distributed random number in the range $[0, 1)$.
   If $p >$ random $[0, 1)$ then $x_1 \leftarrow x_2$ and $E_1 \leftarrow E_2$.
5. Determine if the algorithm should stop.
   If (termination criterion is not met) then goto step 2
   else stop with "optimal" solution x_1.

Examination of step 4 shows that the solution x_2 will always replace x_1 (and, hence, become the incumbent solution) whenever $E_2 \leq E_1$. This indicates that the solution at x_2 is better than the solution at x_1. There is also a chance that x_2 will replace x_1 as the incumbent solution when $E_2 > E_1$ (this is known as "hillclimbing").

Some possible termination criteria are having reached a maximum number of iterations or having successfully replaced the incumbent solution a maximum number of times. Clearly, these maximum numbers must be determined prior to the start of the algorithm.

For any particular invocation of the Metropolis algorithm, the temperature T maintains a constant value. The simulated annealing algorithm is a series of Metropolis algorithms with different (decreasing) values of T.

It is important to note, for the purposes of this paper, that the Metropolis algorithm always keeps a single solution as the incumbent. The perturbed solution will always unseat the incumbent if it is better, and it will sometimes unseat the incumbent if it is worse (this is hillclimbing).

3. Genetic Algorithms and the Two-Membered Evolution Strategy. Genetic algorithms are randomized, population–based search procedures which utilize the Darwinian notion of "survival of the fittest." These algorithms were independently developed by Holland [4] at the University of Michigan and by Rechenberg and Schwefel [5] in Germany. The German versions are known as evolution strategies [ESs] and will be the focus of this section.

The general process of the two–membered ES, denoted (1+1)–ES, is to start with the single population member, mutate it (change it in some fashion prescribed by the mutation operator) to create a single offspring, and then select the better of the two to become the parent for the next generation. The "betterness" quality of an individual arises from the objective function evaluation. If the objective function is to be minimized, then the individual with the smallest function value becomes the parent.

Schwefel [6] describes the (1+1)–ES algorithm with the following 8-tuple:

$$(1+1) - ES = (P^0, m, s, c_d, c_i, f, g, t)$$

where

P^0	$=$	$(x^0, \sigma^0) \in I$	population, $I = \mathbb{R}^n \times \mathbb{R}^n$
m	:	$I \to I$	mutation operator
s	:	$I \times I \to I$	selection operator
c_d, c_i	$\in$	$\mathbb{R}$	step–size control
f	:	$\mathbb{R}^n \to \mathbb{R}$	objective function
g	:	$\mathbb{R}^n \to \mathbb{R}$	constraint functions
t	:	$I \times I \to \{0, 1\}$	termination criterion

At any given time/generation r, P^r represents the parent and $m(P^r)$ is the child (mutated parent). Although the mutation operator can be generalized, it was originally defined in such a way that x'^r (the child) was the addition of the n–element vector x^r (the parent) and an n–element vector of independent, normally–distributed random numbers with zero mean and standard deviation σ^r. Assuming a minimization problem, the parent in generation $r+1$ would be the same as in generation r unless $f(x'^r) \leq f(x^r)$. The step–size controls were used to modify σ^r so that a successful mutation occurred approximately one–fifth of the time. The termination criterion could be defined in numerous ways, including the use of a maximum number of generations or a maximum CPU time.

Again, for the purposes of this paper, it is important to note that in the (1+1)–ES algorithm a single solution is maintained as the incumbent. This incumbent is perturbed each generation and then a selection operator chooses the incumbent for the next generation.

4. Relationship Between the Metropolis Algorithm and (1+1)–ES. The following theorem establishes a relationship between the Metropolis algorithm and the (1+1)-ES algorithm.

Theorem. The Metropolis algorithm is a special case of the two–membered evolution strategy.

Proof. To prove this theorem, it is sufficient to show that the Metropolis algorithm can be defined with the same 8–tuple used for the (1+1)–ES algorithm.

$$\text{Metropolis algorithm} = (P^0, m, s, c_d, c_i, f, g, t).$$

P^0 represents the initial solution. In general, the value of σ^r is arbitrary (unless the mutation operator requires a standard deviation).

The Metropolis algorithm does not specify a particular perturbation method. Therefore, the mutation operator m can be defined in whatever manner is consistent with the perturbation method required by the specific instantiation of the Metropolis algorithm under consideration.

The selection operator s must be defined so that

$$P^{r+1} = \begin{cases} x'^r & \text{if } \exp\left[\frac{-(f(x'^r)-f(x^r))}{kT}\right] > \text{ random } [0,1) \\ x^r & \text{otherwise.} \end{cases}$$

The values of c_d and c_i are arbitrary (unless σ^r needs to be modified so that the mutation success rate can be held approximately constant).

The choice of algorithm will have no impact on the objective function f or the constraint functions g. It is assumed that the mutation operator will generate perturbations that satisfy all constraint functions.

The Metropolis algorithm does not specify a particular termination criterion. Therefore, t can be defined in whatever manner is consistent with the termination criterion required by the specific instantiation of the Metropolis algorithm under consideration.

Remark. This theorem shows that, at a fundamental algorithmic level, the annealing process is a simplistic form of evolution.

5. Example. Here is a simple example using the Metropolis algorithm. Suppose that the following distance matrix is given for the traveling salesperson problem.

city	A	B	C	D	E
A	–	5	9	2	12
B	5	–	6	11	4
C	9	6	–	7	9
D	2	11	7	–	11
E	12	4	9	11	–

Suppose that x_1 is the tour A–B–C–D–E. The associated objective function E_1 is $5+6+7+11+12=41$. Now suppose that x_1 is perturbed by inverting the order of the second through fourth cities in the tour, yielding x_2 = A–D–C–B–E. The associated objective function E_2 is $2+7+6+4+12=31$. Without loss of generality, assume that the Boltzmann parameters k and T are 1 and 0.99, respectively. Using step 3, $p \approx 24368$. Since the calculated value for p is greater than 1, it is reset to 1. Therefore, in step 4, x_2 becomes the incumbent solution.

Suppose that the next iteration perturbs the incumbent solution by inverting the order of the first and second cities, giving a tour of D–A–C–B–E with an objective function value of 32. Since $E_2 > E_1$, step 3 will yield a p value that is less than unity. Therefore, x_2 will replace x_1 as the incumbent solution only if p is greater than the random number generated in step 4. This process, known as hillclimbing, is used to allow the algorithm to escape from (possibly non–global) local minima.

The algorithm will terminate after either a predetermined number of iterations has been reached or after a predetermined number of successful reconfigurations has been reached.

Section 4 of this paper established that the (1+1)–ES is equivalent to the Metropolis algorithm when the parameters are chosen appropriately. Based on this equivalence, the (1+1)–ES would yield the same sequence of x–iterates as the Metropolis algorithm. Therefore, it is not necessary to repeat the example for the (1+1)–ES.

6. Conclusion. Randomized search techniques (including simulated annealing and genetic algorithms) have been applied to a wide variety of problems. Goldberg [7] lists genetic algorithm application problems from diverse disciplines such as biology, computer science, engineering, and social science. Aarts and van Laarhoven give a similar list for simulated annealing in [2].

A characteristic of many of these problems is that they are NP–complete. Although neither simulated annealing nor genetic algorithms can guarantee that an optimal solution to a problem will be found (especially for an NP–complete problem), they have been shown to be robust techniques that generally locate a near–optimal solution.

7. Acknowledgment. The author would like to thank an anonymous referee for providing suggestions to improve this paper.

References

1. N. Metropolis et al., "Equation of State Calculations by Fast Computing Machines," *Journal of Chem. Physics,* 21 (1953), 1087–1092.

2. P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications,* Kluwer Academic Publishers, 1987.

3. W. H. Press et al., *Numerical Recipes in C,* Cambridge University Press, 1988.

4. J. H. Holland, *Adaptation in Natural and Artificial Systems,* The University of Michigan Press, 1975.

5. H. P. Schwefel, *Numerical Optimization of Computer Models,* John Wiley & Sons, 1981.

6. H. P. Schwefel, "A Survey of Evolution Strategy," ed. R. K. Belew and L. B. Booker, *Proceedings of the Fourth International Conference on Genetic Algorithms,* Morgan Kaufmann, 1991, 2–9.

7. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison Wesley, 1989.