Decision-Theoretic Simulated Annealing

Todd W. Neller and **Christopher J. La Pilla**
Gettysburg College
Department of Computer Science
Gettysburg, PA 17325, USA
tneller@gettysburg.edu

Abstract

The choice of a good annealing schedule is necessary for good performance of simulated annealing for combinatorial optimization problems. In this paper, we pose the simulated annealing task decision-theoretically for the first time, allowing the user to explicitly define utilities of time and solution quality. We then demonstrate the application of reinforcement learning techniques towards approximately optimal annealing control, using traveling salesman, clustered traveling salesman, and scheduling problems. Although many means of automating control of annealing temperatures have been proposed, our techniques requires no domain-specific knowledge of problems and provides a natural means of expressing time versus quality tradeoffs. Finally, we discuss alternate abstractions for future decision-theoretic variants.

Introduction

Simulated annealing (SA) is a technique for combinatorial optimization that is inspired by analogy to the cooling and freezing of liquids. A slowly cooled metal reaches a more orderly, lower-energy state than one that is cooled rapidly (i.e. "quenched"). With SA, search in a state space varies with a "temperature" parameter from a random walk to a hill-descending stochastic local optimization. For further background reading on SA, we recommend (Kirkpatrick, Gelatt, & Vecchi 1983) and (Press *et al.* 1992).

The basic SA algorithm can be described as follows:

- Pick an initial state s.
- While cooling (i.e. reducing) the temperature T according to a given schedule:
  1. Generate a next state s'.
  2. Compute $\Delta E = E(s') - E(s)$, the change in energy.
  3. If $\Delta E < 0$, accept the next state ($s \leftarrow s'$).
  4. Otherwise, accept the next state with probability $e^{-\Delta E/kT}$, where k is Boltzmann's constant[1].
- Return the state s^* that minimized E.

[1]In practice, Boltzmann's constant is often omitted from the implementation.

To apply SA, the user must make four design decisions (Press *et al.* 1992): (1) the state representation, (2) the next state generation function, (3) the energy (or objective) function, and (4) the *cooling* or *annealing schedule*. The successful application of SA depends on good choices for each of these. Although (1)-(3) present interesting challenges of their own, we here focus on (4), the annealing schedule. In the case where the temperature is controlled dynamically, we refer to this as annealing control.

In their paper (Kirkpatrick, Gelatt, & Vecchi 1983), Kirkpatrick et al. note the importance of the annealing schedule. They show that different combinatorial optimization problems call for different annealing schedules that change the rate of cooling in specific temperature ranges. However, in practice, many annealing schedules are chosen through arcane, unpublished, and largely ad hoc techniques. The parameters of such schedules (e.g. starting temperature, decay rate, etc.) are typically chosen empirically through limited human experimentation, e.g. (Press *et al.* 1992, p. 446). Whether to avoid such suboptimal human experimentation biases or just to simply avoid the need for such tedious experimentation, there has been much research towards automated annealing control.

For example, Otten and van Ginneken (Otten & van Ginneken 1989) developed a theory for annealing in the framework of Markov chains. Unfortunately, the application of their theory requires the user to know both the size of the state space and the number of global optima. Not only is the state space size difficult to estimate for many problems, but one rarely expects to have knowledge of the number of global optima. Techniques such as this require too much knowledge of the state space.

Many means of controlling annealing, e.g. (Aarts & van Laarhoven 1985), (Huang, Romeo, & Sangiovanni-Vincentelli 1986), (Lam & Delosme 1988), and (Boyan 1998), provide ways to speed cooling and trade off the quality of the result for computational time. This tradeoff of time versus quality of result is fundamental to the practical application of SA. However, such work has been built on a foundation of theoretical work on "optimal" annealing where such optimality is solely concerned with the *quality* (i.e. low energy) of the result and not with the *time* of computation. The time-quality tradeoffs of such techniques are achieved by allowing various deviations from such optimal annealing (e.g. deviations from thermal equilibrium). To date, no method has dealt explicitly with the user's utility of time and quality, i.e. *what is optimal to the user*.

In this paper, we introduce a means of controlling annealing that requires no knowledge of the state space, and allows the user to be explicit about the time-versus-quality tradeoff. This is accomplished by first posing SA decision-theoretically as an optimal control problem, and then applying simple reinforcement learning techniques (Sutton & Barto 1998). Annealing control is learned through annealing experience. Our experiments demonstrate the practicality of this technique through successful application to traveling salesman problems and course scheduling problems.

Related Work

Whereas we will discuss the application of machine learning to the design of annealing schedules, machine learning has also been applied to the separate problems of initial state selection and next state generation. In principle, these techniques could be used in conjunction with our own.

Boyan's STAGE algorithm (Boyan & Moore 1998) is a multi-restart stochastic local search algorithm that intelligently chooses initial states for restarts. Given that each initial state and Markovian stochastic local search leads to a final state with its associated object-level utility, STAGE learns a function approximation for the mapping of initial states to final state utilities, using both the given objective function and additional relevant weighted search features. STAGE then hill-climbs this function approximation to choose the next initial state that is expected to maximize the final state utility for the next stochastic local search.

In his master's thesis (Su, Buntine, & Newton 1997), Su is concerned with combinatorial optimization problems with computationally expensive energy functions, e.g. the VLSI cell-based layout placement problem. Su uses regression to compute a function approximator that efficiently predicts the energy of a state. With each iteration of annealing, many possible next states are sampled and evaluated with this efficient approximator. The sampled neighboring state with the minimal expected energy is chosen as the next state for the iteration.

In the section "A Simple Parameter-Learning Agent", we refer to the agent task as *parameter selection* because of its relation to work on the *algorithm selection problem* as defined in (Rice 1976). In particular, the work of (Lagoudakis & Littman 2000) is closely related to our own in that reinforcement learning techniques are applied to the dynamic recursive selection of algorithms for sorting and order statistic selection problems. In contrast, we learn control of an algorithm parameter dynamically within an iterative architecture. Work in this area increasingly provides evidence of the benefits of learning metalevel algorithm selection and control.

Decision-Theoretic Simulated Annealing

A decision-theoretic controller for simulated annealing maps sensory inputs and utility gains or losses to control outputs so as to maximize the expected utility of the process. These sensory inputs may come at varying intervals in the process, and consist of information about the process. Control outputs dictate how the process should proceed until the controller functions next.

In pursuing optimal control, the two primary considerations are (1) the utility of the quality of the result of SA (i.e. the value of the state found), and (2) the utility of the time taken to compute such a result. Put simply, each iteration of simulated annealing offers a potential gain of solution quality with a definite loss of time.

The measure of result quality, called the object-level (Horvitz 1988) or intrinsic (Russell & Wefald 1991) utility function, should be considered distinct from the state energy function of SA. While object-level utility may be a function of energy, the two functions serve different purposes. A result of SA may not have any object-level utility until its energy crosses a certain threshold. The energy function, by contrast, can and should provide guidance in searching the state space. We denote the energy and object-level utility of the search state s to be $E(s)$ and $U_O(s)$ respectively.

We must also measure the utility of time. The time cost or inference-related utility, denoted $U_I(t)$, represents the cost of computation where t is elapsed SA time. Let s_t be the current search state of an SA process at time t. Let $s_t^* = \text{argmin}_{s_{t'}, t' \in [0,t]} E(s_{t'})$ denote the minimal energy state visited up to time t. Then the *net utility* $U(s_t^*, t)$ of an SA process at time t is $U_O(s_t^*) - U_I(t)$. If we compare the net utility of an SA process at two different times t_1 and t_2, then the net utility of the SA process from t_1 to t_2 is $U(s_{t_2}^*, t_2) - U(s_{t_1}^*, t_1)$.

The annealing agent may choose between iterations how to proceed with annealing. We call each of the annealing agent control signals an *annealing action*. An annealing action may dictate a temperature control policy and the duration (i.e. number of iterations) of that policy. Another important annealing action is to *terminate* the SA process. An annealing agent monitors the state of the SA process and directs its progress or termination.

Optimal control of annealing would then be an agent mapping annealing process information to annealing control actions that maximize expected utility. Thus, learning optimal control of annealing entails some form of search among mappings from annealing states to annealing actions for one that approximates optimal control. For this we apply policy evaluation and improvement techniques of reinforcement learning (Sutton & Barto 1998).

Experiments and Observations

We describe two annealing agents; a parameter-learning agent demonstrates that simple reinforcement learning techniques can be applied to existing annealing schedules to find approximately optimal parameter settings, subject to a user-defined utility measure. We then describe a more complex annealing control agent that demonstrates how reinforcement learning may be used to obtain problem-specific, adaptive annealing schedules.

First, we present an overview of the annealing problems used to evaluate our learning agents.

Annealing Problems

Three combinatorial optimization problems were used for our experiments: the traveling salesman problem, the clustered traveling salesmen problem, and a scheduling problem. Both of the traveling salesmen problems and Lin and Kernighan's next state generation function are described in (Kirkpatrick, Gelatt, & Vecchi 1983). In both cases, 400 cities were distributed on a 350-by-350 grid.

Each randomly generated instance of our simple course-scheduling problem consists of 2500 students, 500 courses, and 12 time slots. Each student is enrolled in 5 unique random courses. Each state of the problem is represented as a list containing the time slot for each course. Next states are generated by randomly changing the time slot of one course. The energy function $E(s)$ is defined as the total number of course conflicts, i.e. the total number of class pairs in each student's enrollment which occur in the same time slot.

For all of our tests, we define our measure of the object-level utility of a state $U_O(s)$ such that changes in energy at low energy states are weighted more heavily than changes at high energy states: $U_O(s_t) = -log_{10}(E(s_t))$. Our time cost function $U_I(t) = ct$, where c is a fixed cost per iteration and t is the number of iterations completed. This allows us to compare results generated on different machines.

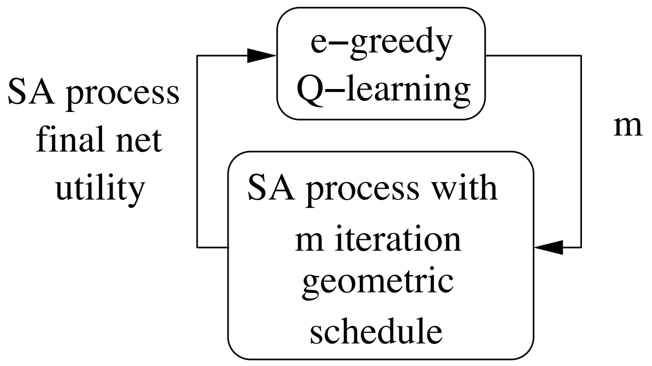
In previous work (Neller & Hettlinger 2003), we have shown that a preliminary version of a decision-theoretic annealing agent was capable of learning a rapid-cooling schedule, i.e. "simulated quenching". For these experiments, we wished to demonstrate learning of a non-extreme behavior which is neither quenching, nor close to thermal equilibrium. We found that for all problems, an iteration cost $c = 2.2 \times 10^{-8}$ achieved this objective.

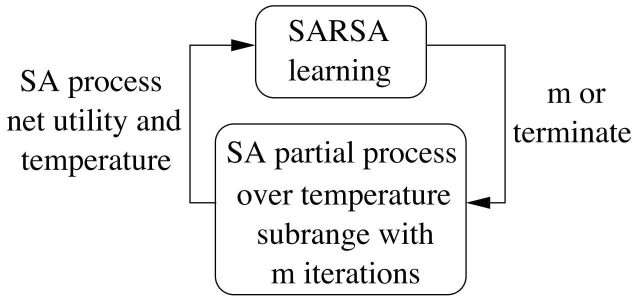
A Simple Parameter-Learning Agent

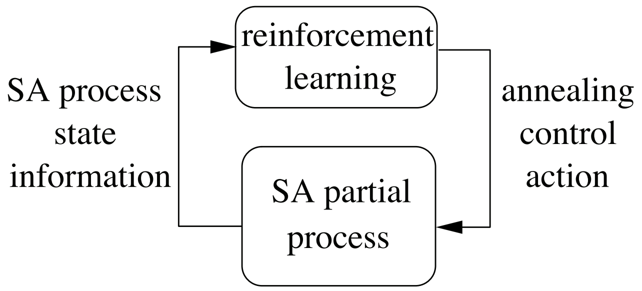
In this section, we present a parameter-learning agent (Figure 1(a)) which is both simple to implement and aids in understanding the more complex dynamic control agent of the next section. Our agent treats an annealing schedule or autonomous annealing controller as a black box with parameters. We assume a reasonable, finite set of parameter choices A with $|A| = n$. The agent attempts to find the optimal parameter values through experimentation.[2] There are many different action selection approaches for balancing explorative versus exploitative actions (Sutton & Barto 1998). Since our purpose is pedagogical, we will describe a simple ϵ-greedy agent.[3]

[2]We can view this experimental task as an *n-armed bandit problem* with conditional dependencies. Numerous studies, e.g. (Horowitz 1973; Meyer & Shi 1995; Anderson 2001; Gans, Knox, & Croson 2004), have shown that human experimentation on such problems is suboptimal in predictable ways. This further reinforces the usefulness of automated approaches.

[3]Our softmax agent had faster convergence to optimal behavior, but this would require us to have two separate applications of Boltzmann distributions with two temperature parameters.


e–greedy Q–learning
SA process final net utility
m
SA process with m iteration geometric schedule
(a) Simple parameter-learning agent


SARSA learning
SA process net utility and temperature
m or terminate
SA partial process over temperature subrange with m iterations
(b) Annealing control agent



(c) General DTSA agent

Figure 1: Metalevel learning of annealing control

Our agent learns parameter selection over random representative sets of problems. Let $Q_t(a)$ be the average decision-theoretic net utility of all complete trials at learning iteration t using parameter choice $a \in A$. When no trials have been performed with a, we define $Q_t(a) = 0$.

Each iteration of learning proceeds as follows. With probability ϵ, we uniformly, randomly select a parameter $a \in A$. Otherwise, we select the a having the highest expected net utility (i.e. the "greedy" action), that is, we select $\mathrm{argmax}_{a \in A} Q_t(a)$. We run the algorithm with the selected a, and incrementally update our average $Q_t(a)$ with the experienced net utility.

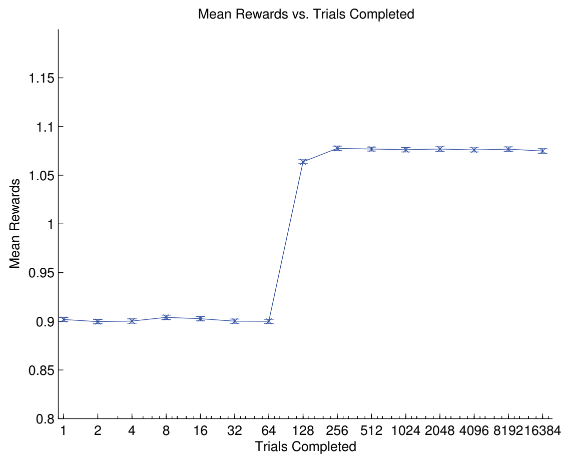
Our simple ϵ-greedy agent used $\epsilon = .1$. No effort was made to tune ϵ in order to show that this metalevel control did not itself need metalevel control.

For each annealing trial, the parameter to be selected is m, the number of annealing iterations. Parameter set A consists of 9 choices for m, equally-spaced logarithmically between 10^5 and 10^7 (inclusive). The annealing schedule was a simple geometric schedule, where α was computed to take the temperature from a fixed T_0 and T_{m-1} in m iterations. The starting temperature T_0 was derived using the method described in (White 1984; Huang, Romeo, & Sangiovanni-Vincentelli 1986), where one performs random walks on initial states, computing the standard deviation σ of the ΔE's. Then, an initial temperature of $T_0 = 20\sigma$ is sufficiently high to accept virtually all ΔE's, and serves as a reasonable starting temperature. The final temperature was derived similarly, such that any positive ΔE for a problem would be almost certainly rejected. Starting and ending temperatures for the traveling salesman problems were 2×10^4 and 1×10^{-4} respectively; for the scheduling problem, starting and ending temperatures were 1×10^3 and 1×10^{-3} respectively.

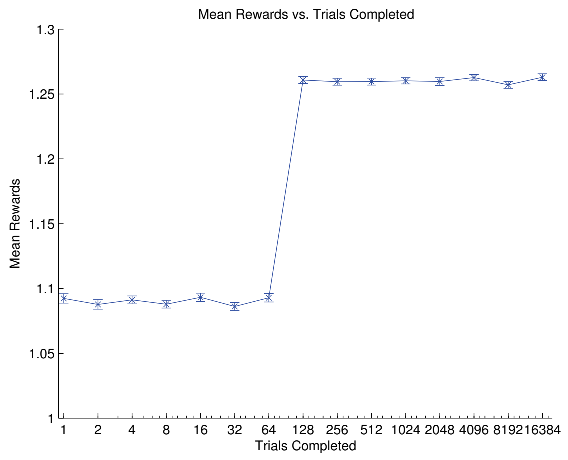
For each experiment, learning occurs over 16384 (2^{14}) simulated annealing trials with randomly generated instances of a given problem. After each trial that is a power of 2, a greedy policy is logged based on the current $Q_t(a)$ estimates. These policies were then later tested with a randomly generated benchmark set of 100 problems. All figures show the tested utility means and 90% confidence intervals from bootstrapping with 10000 resamples. The progress of ϵ-greedy learning is shown in Figure 2.

In each case, we see that there is an initial period of experimentation after which the quality greatly increases to the optimal setting. This jump occurred between 64 and 128 iterations for traveling salesman problems, and between 32 and 64 iterations for the scheduling problem. After the jump, the policy changed slightly (except in the case of the clustered TSP) and fixed on the optimal selection for the remainder of trials, which is 10^6 iterations for the clustered TSP and 5.6×10^5 for the other problems. Peak mean utilities for the TSP, clustered TSP, and scheduling problems were 1.078, 1.263, and 0.5798 respectively.

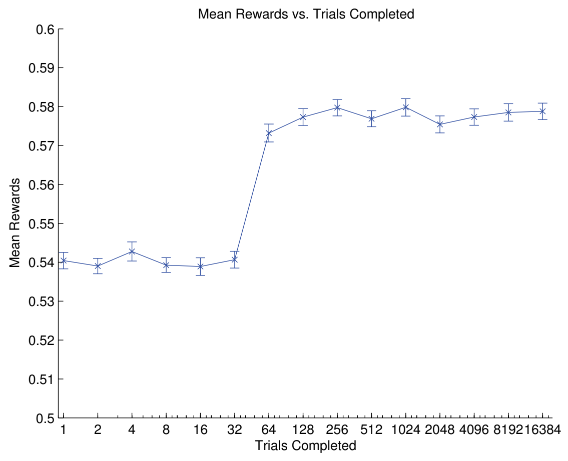
We do not claim that such ϵ-greedy action selection is optimal sampling for this number of trials. Rather, we view this as one of an array of reasonable choices for online learning that serves as a simple and valuable wrapper for annealing schedule black boxes which require parameter tuning.



(a) Traveling salesman problem



(b) Clustered TSP



(c) Scheduling problem

Figure 2: ϵ-greedy learning of geometric schedules

This is significant from a software engineering standpoint, as adding very little, simple code can obviate the need for a tedious and bias-prone tuning task.

If one desires a faster convergence, there are many options. In our experiences, softmax action selection provides faster convergence but greater policy instability without parametric adjustment through learning. Optimistic initial values of $Q_0(a) = 1.5$ cause all choices to be tested in the first 9 trials, leading to convergence to the optimal parameter selection within 32 iterations for the TSP and 16 iterations for the other problems. However, this approach leverages domain knowledge. Indeed, there are many better sampling techniques, but we have opted for a naïve, simple approach for generality and ease of exposition.

Annealing Control Agent

The simple agent of the previous section treated the annealing process as a black box with parameter tuning as control. The choice was between a number of geometric schedules through a given temperature range. In this section, we show the viability of dynamic control of annealing temperature.

For easy illustration, we choose to formulate the control problem with a slight and important difference from the simple agent. We subdivide the temperature range into subranges, allowing the controller agent to dynamically choose from a number of geometric schedules for each temperature subrange when it transitions from one to another. Thus, the agent dynamically chooses monotonic piecewise-geometric schedules (Figure 1(b)).

For our experiments, the temperature ranges used earlier were logarithmically partitioned into 10 subranges. For each subrange, the agent could, as before, choose the number of iterations it would take to decay the temperature geometrically from the highest to the lowest temperature of the range. Since there were ten partitions, these choices were logarithmically distributed between 10^4 and 10^6 (inclusive), so that the agent's most extreme fast and slow annealing schedules were identical to those of the simple agent.

The learning algorithm was the tabular SARSA learning algorithm of (Rummery & Niranjam 1994), described in (Sutton & Barto 1998), with $\alpha = 0.1$ and $\gamma = 1$. Our evaluation of this agent's learning, shown in Figure 3, is the same as before except for one significant difference. During periodic greedy evaluations of the policy, we allowed annealing to self-terminate if the expected utility of future iterations was negative. This ability to self-terminate at the "point of diminishing returns" is an interesting advantage of such dynamic control.

The peak mean utility for the clustered TSP was 1.271. While the added flexibility did not significantly increase the overall utility of annealing, we do note that convergence to near optimal annealing was significantly faster, achieving approximately optimal net utility after only a few trials. This faster convergence was observed consistently across multiple learning trials.

Again, we do not suggest that this agent represents the optimal means of controlling annealing. Rather, this combination of decision-theory and reinforcement learning is the first of a new class of algorithms we believe merits further

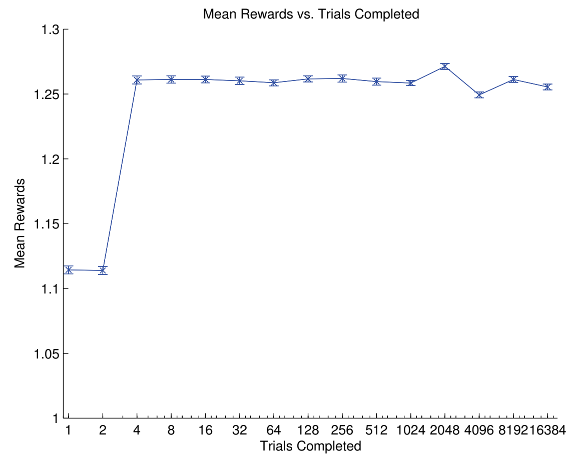


Figure 3: ϵ-greedy learning of piecewise geometric temperature control for the clustered TSP

investigation. We next consider degrees of design freedom for this class of decision-theoretic simulated annealing algorithms.

Future Work

The general annealing agent (Figure 1(c)) is essentially an adaptive mapping from annealing process information to annealing actions. Over time, the agent seeks to maximize its expected utility. We have made simple choices for states and actions for our experiments. In this section, we consider the many possibilities for future research.

States: One could represent the SA process as a Markov decision process (MDP) based on each search state s_t. To condition our decisions on actual states of the search space requires knowledge of the state space (i.e. domain-specific knowledge) that would make our technique less general. We have instead opted, as with other automatic annealing controllers, to approximate optimal control by use of an approximating MDP. In general, one can base the approximating MDP on an abstraction of the current state of the SA process itself, considering only the most important domain-independent features such as time t, temperature T, energy $E(s_t)$, specific heat (Kirkpatrick, Gelatt, & Vecchi 1983), or various Markov chain statistics. We have simply used the temperature as an abstraction of the process. One important open research question is which domain-independent annealing process features are most significant for decision-theoretic annealing control.

Actions: At intervals during annealing, the annealing agent may choose whether and how to proceed with annealing. This includes both an annealing policy and the duration of that policy. For example, an agent may decide to perform 1000 iterations of annealing at a specific fixed temperature. It may instead, as we have described, decide to perform 100 iterations with a fixed temperature decay rate. The agent may also decide to raise the temperature for a period (i.e. simulated tempering), or may decide to terminate the SA process.

While it is possible to have the annealing agent choose a temperature for each iteration of simulated annealing, this is unnecessary and introduces significant computational overhead. This is decision-making at a fine-grained extreme where all annealing schedules can be expressed. At the same time, this yields a vast space of potential SA state-action mappings to consider, posing problems for learning.

Another important open research question is which annealing control actions are most beneficial, and which conditions should trigger the controller's intervention.

Conclusions

The most important contribution of this work is that we have posed the simulated annealing control problem decision-theoretically for the first time. Given the extreme practical importance of the time versus quality tradeoff, it is surprising that the simulated annealing literature has not previously framed this problem decision-theoretically.

We have also introduced both simple and complex agents for learning parameter selection and dynamic control of annealing respectively. One important simple application of this work is to add a layer of metalevel control to existing optimization algorithms in order to tune parameters for optimal balance of time versus quality.

However, we have shown that beyond adding to existing algorithms, we can develop entirely new algorithms for dynamic control of temperature throughout the annealing process. The unexplored possibilities are exciting, as a single algorithm could flexibly learn to hill-climb, anneal, temper, or terminate as appropriate for maximizing utility.

References

Aarts, E., and van Laarhoven, P. 1985. A new polynomial time cooling schedule. In *Proc. ICCAD-95: IEEE International Conference on Computer Aided Design*, 206–208.

Anderson, C. M. 2001. *Behavioral Models of Strategies in Multi-Armed Bandit Problems*. Ph.D. Dissertation, California Institute of Technology, Pasedena, CA.

Boyan, J., and Moore, A. 1998. Learning evaluation functions for global optimization and boolean satisfiability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 3–10.

Boyan, J. A. 1998. *Learning Evaluation Functions for Global Optimization*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburg, PA, USA. Available as Carnegie Mellon Tech. Report CMU-CS-98-152.

Gans, N.; Knox, G.; and Croson, R. 2004. Simple models of discrete choice and their performance in bandit experiments. working paper, The Wharton School, University of Pennsylvania.

Horowitz, A. V. 1973. *Experimental study of the two-armed bandit problem.* Ph.D. Dissertation, University of North Carolina, Chapel Hill, NC. Unpublished.

Horvitz, E. J. 1988. Reasoning under varying and uncertain resource constraints. In *Proc. National Conference on Artificial Intelligence (AAAI-88)*, 111–116.

Huang, M. D.; Romeo, F.; and Sangiovanni-Vincentelli, A. 1986. An efficient general cooling schedule for simulated annealing. In *Proc. ICCAD-86: IEEE Int. Conf. on Computer Aided Design*, 381–394.

Kirkpatrick, S.; Gelatt, C.; and Vecchi, M. 1983. Optimization by simulated annealing. *Science* 220:671–680.

Lagoudakis, M. G., and Littman, M. L. 2000. Algorithm selection using reinforcement learning. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, 511–518. Morgan Kaufmann.

Lam, J., and Delosme, J.-M. 1988. Performance of a new annealing schedule. In *Proc. 25th ACM/IEEE Design Automation Conference*, 306–311.

Meyer, R. J., and Shi, Y. 1995. Sequential choice under ambiguity: Intuitive solutions to the armed-bandit problem. *Management Science* 41(5):817–834.

Neller, T. W., and Hettlinger, D. C. 2003. Learning annealing schedules for channel routing. In *Proceedings of the International Conference on VLSI*, 298–302.

Otten, R., and van Ginneken, L. 1989. *The Annealing Algorithm.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 1992. *Numerical Recipes in C: the art of scientific computing - 2nd Ed.* Cambridge: Cambridge University Press.

Rice, J. R. 1976. The algorithm selection problem. In *Advances in Computers*, volume 15. Academic Press. 65–118.

Rummery, G., and Niranjam, M. 1994. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, Cambridge, England.

Russell, S., and Wefald, E. 1991. *Do the Right Thing: studies in limited rationality.* Cambridge, MA: MIT Press.

Su, L.; Buntine, W.; and Newton, R. 1997. Learning as applied to simulated annealing. Master's thesis, University of California, Berkeley, CA.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: an introduction.* Cambridge, MA: MIT Press.

White, S. 1984. Concept of scale in simulated annealing. In *Proc. ICCD-84: IEEE International Conference on Computer Design*, 646–651.