Optimal Anytime Constrained Simulated Annealing for Constrained Global Optimization*

Benjamin W. Wah and Yi Xin Chen

Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
1308 West Main Street
Urbana, IL 61801, USA
{wah, chen}@manip.crhc.uiuc.edu
URL: http://www.manip.crhc.uiuc.edu

Abstract. In this paper we propose an *optimal* anytime version of *constrained simulated annealing* (CSA) for solving constrained nonlinear programming problems (NLPs). One of the goals of the algorithm is to generate feasible solutions of certain prescribed quality using an average time of the same order of magnitude as that spent by the original CSA with an optimal cooling schedule in generating a solution of similar quality. Here, an *optimal cooling schedule* is one that leads to the shortest average total number of probes when the original CSA with the optimal schedule is run multiple times until it finds a solution. Our second goal is to design an anytime version of CSA that generates gradually improving feasible solutions as more time is spent, eventually finding a *constrained global minimum* (CGM). In our study, we have observed a monotonically non-decreasing function relating the success probability of obtaining a solution and the average completion time of CSA, and an exponential function relating the objective target that CSA is looking for and the average completion time. Based on these observations, we have designed CSA_{AT-ID}, the anytime CSA with iterative deepening that schedules multiple runs of CSA using a set of increasing cooling schedules and a set of improving objective targets. We then prove the optimality of our schedules and demonstrate experimentally the results on four continuous constrained NLPs. CSA_{AT-ID} can be generalized to solving discrete, continuous, and mixed-integer NLPs, since CSA is applicable to solve problems in these three classes. Our approach can also be generalized to other stochastic search algorithms, such as genetic algorithms, and be used to determine the optimal time for each run of such algorithms.

1 Introduction

A large variety of engineering applications can be formulated as constrained *nonlinear programming problems* (NLPs). Examples include production planning,

* Proc. Sixth International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, Sept. 2000

computer integrated manufacturing, chemical control processing, and structure optimization. Some applications that are inherently constrained or have multiple objectives may be formulated as unconstrained mathematical programs due to a lack of good solution methods. Examples include applications in neural-network learning, computer-aided design for VLSI, and digital signal processing. High-quality solutions to these applications are important because they may lead to lower implementation and maintenance costs.

By first transforming multi-objective NLPs into single-objective NLPs, all constrained NLPs can be considered as single-objective NLPs. Without loss of generality, we consider only minimization problems in this paper. A general discrete constrained NLP is formulated as follows:

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g(x) \leq 0 & x = (x_1, x_2, \ldots, x_n) \text{ is a vector} \\ & h(x) = 0 & \quad \text{of discrete variables,} \end{array} \tag{1}$$

where $f(x)$ is a lower-bounded objective function, $h(x) = [h_1(x), \cdots, h_m(x)]^T$ is a set of m equality constraints, and all the discrete variables in x are finite. Both $f(x)$ and $h(x)$ can be either linear or nonlinear, continuous or discrete (*i.e.* discontinuous), and analytic in closed forms or procedural. In particular, we are interested in application problems whose $f(x)$, $g(x)$, and $h(x)$ are non-differentiable. Our general formulation includes both equality and inequality constraints, although it is shown later that inequality constraints can be transformed into equality constraints. The search space (sometimes called solution space) X is the finite set of all possible combinations of discrete variables in x that may or may not satisfy the constraints. Such a space is usually limited by some bounds on the range of variables.

To characterize the solutions sought in discrete space, we define for discrete problems, $\mathcal{N}(x)$, the *neighborhood* [1] of point x in discrete space X, as a *finite* user-defined set of points $\{x' \in X\}$ such that x' is reachable from x in one step, that $x' \in \mathcal{N}(x) \Longleftrightarrow x \in \mathcal{N}(x')$, and that it is possible to reach every other x'' starting from any x in one or more steps through neighboring points. Note that neighboring points may be feasible or infeasible.

Point $x \in X$ is called a discrete *constrained local minimum* (CLM) if it satisfies two conditions: a) x is a feasible point, implying that x satisfies all the constraints $g(x) \leq 0$ and $h(x) = 0$, and b) $f(x) \leq f(x')$, for all $x' \in \mathcal{N}(x)$ where x' is feasible. A special case in which x is a CLM is when x is feasible and all its neighboring points are infeasible.

Point $x \in X$ is called a *constrained global minimum* (CGM) iff a) x is a feasible point, and b) for every feasible point $x' \in X$, $f(x') \geq f(x)$. According to our definitions, a CGM must also be a CLM.

In the next section we formulate the problem that we study in this paper. This is followed by a summary of the constrained simulated annealing algorithm (CSA) in Section 3 and a statistical model on the CSA procedure in Section 4. Finally, we present our proposed anytime CSA with iterative deepening in Section 5 and our experimental results in Section 6.

2 Formulation of the Problem

Constrained simulated annealing (CSA) [14] (see Section 3) has been proposed as a powerful global minimization algorithm that can guarantee asymptotic convergence to a CGM with probability one when applied to solve (1).

One of the difficulties in using CSA, like conventional *simulated annealing* (SA) [8], is to determine an *annealing schedule*, or the way that temperatures are decreased in order to allow a solution of prescribed quality to be found quickly. In general, the asymptotic convergence of CSA to a CGM with probability one was proved with respect to a cooling schedule in which temperatures are decreased in a logarithmic fashion [14], based on the original necessary and sufficient condition of Hajek developed for SA [6]. It requires an infinitely long cooling schedule in order to approach a CGM with probability one.

In practice, asymptotic convergence can never be exploited since any algorithm must terminate in finite time. There are two ways to complete CSA in finite time. The first approach uses an infinitely long logarithmically decreasing cooling schedule but terminates CSA in finite time. This is not desirable because CSA will most likely not have converged to any feasible solution when terminated at high temperatures.

The second approach is to design a cooling schedule that can complete in prescribed finite time. In this paper we use the following *geometric cooling schedule* with *cooling rate* α:

$$T_{j+1} = \alpha \times T_j, \qquad j = 0, \cdots, N_\alpha - 1, \tag{2}$$

where $\alpha < 1$, j measures the number of probes in CSA (assuming one probe is made at each temperature and all probes are independent), and N_α is the total number of probes in the schedule. A probe here is a neighboring point examined by CSA, independent of whether CSA accepts it or not. We use the number of probes expended to measure overhead because it is closely related to execution time. Given $T_0 > T_{N_\alpha} > 0$ and α, we can determine N_α, the *length of a cooling schedule*, as:

$$N_\alpha = log_\alpha \frac{T_{N_\alpha}}{T_0}. \tag{3}$$

Note that the actual number of probes in a successful run may be less than N_α, as a run is terminated as soon as a desirable solution is found. However, it should be very close to N_α, as solutions are generally found when temperatures are low.

The effect of using a finite α is that CSA will converge to a CGM with probability *less than one*. When CSA uses a finite cooling schedule N_α, we are interested in its *reachability probability* $P_R(N_\alpha)$, or the probability that it will find a CGM in any of its previous probes when it stops. Let p_j be the probability that CSA finds a CGM in its j^{th} probe, then $P_R(N_\alpha)$ when it stops is:

$$P_R(N_\alpha) = 1 - \prod_{j=1}^{N_\alpha} (1 - p_j). \tag{4}$$

Table 1. An example illustrating trade-offs between the expected total number of probes in multiple runs of CSA to find a CGM, the cooling rate used in each run, and the probability of success in each run. The optimal cooling rate at $\alpha = 0.574$ leads to the minimum average total number of probes to find a CGM. Note that the probability of success is not the highest in one run using the optimal cooling rate. (The problem solved is defined in (6). Each cooling schedule is run 200 times using $f' = 200$.)

α	cooling rate in one run	0.139	0.281	0.429	**0.574**	0.701	0.862	0.961	0.990
N_α	avg. cooling schedule	99.8	148.0	207.5	**296.0**	434.5	798.0	2414.0	6963.5
T_α	avg. CPU time per run	0.026	0.036	0.050	**0.074**	0.11	0.18	0.54	1.58
$P_R(N_\alpha)$	succ. prob. of one run	1%	10%	25%	**40%**	55%	70%	85%	95%
$\frac{1}{P_R(N_\alpha)}$	avg. runs to find sol'n	100	10	4	**2.5**	1.82	1.43	1.18	1.05
$\frac{N_\alpha}{P_R(N_\alpha)}$	avg. probes to find sol'n	9980	1480	830	**740**	790	1140	2840	7330
$\frac{T_\alpha}{P_R(N_\alpha)}$	avg. time to find sol'n	2.6	0.36	0.20	**0.19**	0.20	0.25	0.64	1.7

Reachability can be maintained by keeping the best solution found at any time and by reporting the best solution when CSA stops.

Although the exact value of $P_R(N_\alpha)$ is hard to estimate and control, we can always improve the chance of hitting a CGM by running CSA multiple times, each using a finite cooling schedule. Given $P_R(N_\alpha)$ for each run of CSA and that all runs are independent, the expected number of runs to find a solution is $\frac{1}{P_R(N_\alpha)}$ and the expected total number of probes is:

$$\begin{array}{c}\text{Expected total number of}\\ \text{probes to find a CGM}\end{array} = \sum_{j=1}^{\infty} P_R(N_\alpha)(1 - P_R(N_\alpha))^{j-1} N_\alpha j = \frac{N_\alpha}{P_R(N_\alpha)} \quad (5)$$

Table 1 illustrates trade-offs between N_α and $P_R(N_\alpha)$ in solving a constrained NLP with a 10-dimensional Rastrigin function as its objective:

$$\begin{array}{ll} \text{minimize} & f(x) = F\left(10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)), 200\right) \\ \text{subject to} & |(x_i - 4.2)(x_i + 3.2)| \leq 0.1 \qquad \text{for } n = 10, \end{array} \quad (6)$$

where F is the transformation function defined later in (11). A run of CSA is successful if it finds a feasible point with objective value less than or equal to 200 in this run, and the probability to hit a CGM is calculated by the percentage of successful runs over 200 independent runs.

Table 1 shows that $P_R(N_\alpha)$ increases towards one when α is increased. A long cooling schedule is generally undesirable because the expected number of probes in (5) is large, even though the success probability in one run of CSA approaches one. On the other hand, if the schedule is too short, then the success probability in one run of CSA is low, leading to a large expected number of probes in (5). An optimal schedule is one in which CSA is run multiple times and the expected total number of problems in (5) is the smallest.

Definition 1. *An optimal cooling schedule is one that leads to the smallest average total number of probes of multiple runs of CSA in order to find a solution of prescribed quality.*

Table 1 shows that $\frac{N_\alpha}{P_R(N_\alpha)}$ is a convex function with a minimum at $\alpha = 0.574$. That is, the average total number of probes of multiple runs of CSA to find a CGM first decreases and then increases, leading to an optimal cooling rate of 0.574 and an average of 2.5 runs of CSA to find a CGM.

This paper aims at determining an optimal cooling schedule that allows a solution of prescribed quality to be found in the shortest average amount of time. In order to find the optimal cooling schedule, users generally have to experiment by trial and error until a suitable schedule is found. Such tuning is obviously not practical in solving large complex problems. In that case, one is interested in running a single version of the algorithm that can adjust its cooling schedule dynamically in order to find a schedule close to the optimal one. Moreover, one is interested in obtaining improved solutions as more time is spent on the algorithm. Such an algorithm is an *anytime algorithm* because it always reports the best solution found if the search were stopped at any time.

The goals of this paper are two folds. First, we like to design cooling schedules for CSA in such a ways that the average time spent in generating a solution of certain quality is of the same order of magnitude as that of multiple run of the original CSA with an *optimal* cooling schedule. In other words, the new CSA is optimal in terms of average completion time up to an order of magnitude with respect to that of the original CSA with the best cooling schedule. Second, we like to design a set of objective targets that allow an anytime-CSA to generate improved solutions as more time is spent, eventually finding a CGM.

The approach we take in this paper is to first study statistically the performance of CSA. Based on the statistics collected, we propose an exponential model relating the value of objective targets sought by CSA and the average execution time, and a monotonically non-decreasing model relating the success probability of obtaining a solution and the average execution time. These models lead to the design of CSA_{AT-ID}, the anytime CSA with iterative deepening, that schedules multiple runs of CSA using a set of increasing cooling schedules that exploit the convexity of (5) and a set of improving objective targets.

Let $T_{opt}(f_i)$ be the average time taken by the original CSA with an optimal cooling schedule to find a CLM of value f_i or better, and $T_{AT-ID}(f_i)$ be the average time taken by CSA_{AT-ID} to find a CLM of similar quality. Based on the principle of iterative deepening [9], we prove the optimality of CSA_{AT-ID} by showing:

$$T_{AT-ID}(f_i) = O\left(T_{opt}(f_i)\right) \text{ where } i = 0, 1, 2, \cdots \tag{7}$$

Further, CSA_{AT-ID} returns solutions of values $f_0 > \cdots > f^*$ that are gradually improving with time.

There were many past studies on annealing schedules in SA. Schedules studied include logarithmic annealing schedules [6] that are necessary and sufficient for asymptotic convergence, schedules inversely proportional to annealing steps in FSA [13] that are slow when the annealing step is large, simulated quenching scheduling in ASA [7] that is not efficient when the number of variables is large, proportional (or geometric) cooling schedules [8] using a cooling rate between

0.8-0.99 or a rate computed from the initial and final temperatures [11], constant annealing [3], arithmetic annealing [12], polynomial-time cooling [2] adaptive temperature scheduling based on the acceptance ratio of bad moves [16], and non-equilibrium SA (NESA) [4] that operates at a non-equilibrium condition and that reduces temperatures as soon as improved solutions are found.

All the past studies aimed at designing annealing schedules that allow one run of SA to succeed in getting a desirable solution. There was no prior studies that examine trade-offs between multiple runs of SA using different schedules and the improved probability of getting a solution. Our approach in this paper is based on multiple runs of CSA, whose execution times increase in a geometric fashion and whose last run finds a solution to the application problem. Based on iterative deepening [9], the total time of all the runs will be dominated by the last run and will only be a constant factor of the time taken in the last run.

3 Constrained Simulated Annealing

In this section, we summarize our Lagrange-multiplier theory for solving discrete constrained NLPs and the adaptation of SA to look for discrete saddle points.

Consider a discrete equality-constrained NLP:

$$\begin{aligned} &\textit{minimize}_x \quad f(x) \\ &\textit{subject to} \quad h(x) = 0, \end{aligned} \tag{8}$$

where $x = (x_1, \ldots, x_n)$ is a vector of discrete variables, and $f(x)$ and $h(x)$ are analytic in closed forms (but not necessarily differentiable) or procedural. An inequality constraint like $g_j(x) \leq 0$ can be transformed into an equivalent equality constraint $\max(g_j(x), 0) = 0$. Hence, without loss of generality, our theory only considers application problems with equality constraints.

A *generalized discrete Lagrangian function* of (8) is defined as follows:

$$L_d(x, \lambda) = f(x) + \lambda^T H\left(h(x)\right), \tag{9}$$

where H is a continuous transformation function satisfying $H(y) = 0$ iff $y = 0$.

We define a *discrete saddle point* (x^*, λ^*) with the following property:

$$L_d(x^*, \lambda) \leq L_d(x^*, \lambda^*) \leq L_d(x, \lambda^*) \tag{10}$$

for all $x \in \mathcal{N}(x^*)$ and all $\lambda \in R$. Essentially, a saddle point is one in which $L_d(x^*, \lambda)$ is at a local maximum in the λ subspace and at a local minimum in the x subspace. The concept of saddle points is very important in discrete problems because, starting from them, we can derive the first-order necessary and sufficient condition for CLM that lead to global minimization procedures. This is stated formally in the following theorem [15]:

Theorem 1. *First-order necessary and sufficient condition for CLM. A point in the variable space of (8) is a CLM if and only if it satisfies the saddle-point condition (10).*

```
1.  procedure CSA
2.      set initial x = (x, λ) by randomly generating x and by setting λ ← 0;
3.      initialize temperature T_0 to be large enough and cooling rate 0 < α < 1
4.      set N_T (number of probes per temperature);
5.      while stopping condition is not satisfied do
6.          for n ← 1 to N_T do
7.              generate x' from N(x) using G(x, x');
8.              accept x' with probability A_T(x, x')
9.          end_for
10.         reduce temperature by T ← α × T;
11.     end_while
12. end_procedure
```

Fig. 1. CSA: Constrained simulated annealing [15].

Figure 1 describes CSA [14] that looks for saddle points with the minimum objective value. By carrying out *probabilistic ascents* in the λ subspace with a probability of acceptance governed by a temperature, it looks for local maxima in that subspace. Likewise, by carrying out *probabilistic descents* in the x subspace, it looks for local minima in that subspace. It can be shown that the point where the algorithm stops is a saddle point in the Lagrangian space.

CSA differs from traditional SA that only has probabilistic descents in the x space, and the point where SA stops is a local minimum of the objective function of an unconstrained optimization. By extending the search to saddle points in a Lagrangian space, CSA allows constrained optimization problems to be solved in a similar way as SA in solving unconstrained optimization problems.

Using distribution $G(\mathbf{x}, \mathbf{x}')$ to generate trial point $\mathbf{x}'$ in neighborhood $\mathcal{N}(\mathbf{x})$, a Metropolis acceptance probability $A_T(\mathbf{x}, \mathbf{x}')$, and a logarithmic cooling schedule, CSA has been proven to have asymptotic convergence with probability one to a CGM. This is stated in the following theorem without proof [14].

Theorem 2. *Asymptotic convergence of CSA. The Markov chain modeling CSA converges to a CGM with probability one.*

Although Theorems 1 and 2 were derived for discrete constrained NLPs, it is applicable to continuous and mixed-integer constrained NLPs if all continuous variables were first discretized. Discretization is acceptable in practice because numerical evaluations of continuous variables using digital computers can be considered as discrete approximation of the original variables up to a computer's precision. Intuitively, if discretization is fine enough, the solutions found are fairly good approximations to the original solutions. Due to space limitations, we do not discuss the accuracy of solutions found in discretized problems [17]. In the rest of this paper, we apply CSA to solve constrained NLPs, assuming that continuous variables in continuous and mixed-integer NLPs are first discretized.

4 Performance Modeling of CSA

The performance of a CSA procedure to solve a given application problem from a random starting point can be measured by the probability that it will find a solution of a prescribed quality when it stops and the average time it takes to find the solution. There are many parameters that will affect how CSA performs, such as neighborhood size, generation probability, probability of accepting a point generated, initial temperature, cooling schedule, and relaxation of objective function. In this section, we focus on the relationship among objective targets, cooling schedules, and probabilities of finding a desirable solution.

4.1 Relaxation of objective target

One way to improve the chance of finding a solution by CSA is to look for CLM instead of CGM. An approach to achieve this is stop CSA whenever it finds a CLM of a prescribed quality. This approach is not desirable in general because CSA may only find a CLM when its temperatures are low, leading to little difference in times between finding CLM and CGM. Further, it is necessary to prove the asymptotic convergence of the relaxed CSA procedure.

A second approach that we adopt in this paper is to modify the constrained NLP in such a way that a CLM of value smaller than f' in the original NLP is considered a CGM in the relaxed NLP. Since the CSA procedure is unchanged, its asymptotic convergence behavior remains the same. The relaxed NLP is obtained by transforming the *objective target* of the original NLP:

$$F(f(x), f') = \begin{cases} f' & \text{if } f(x) \leq f' \\ f(x) & \text{if } f(x) > f' \end{cases}. \tag{11}$$

Assuming that f^* is the value of the CGM in the original NLP, it follows that the value of the CGM of the relaxed NLP is f^* if $f' \leq f^*$ and is f' if $f' > f^*$. Moreover, since the relaxed problem is a valid NLP solvable by CSA, CSA will converge asymptotically to a CGM of the relaxed NLP with probability one.

As a relaxed objective function leads to a possibly larger pool of solution points, we expect CSA to have a higher chance of hitting one of these points during its search. This property will be exploited in CSA_{AT-ID} in Section 5.2.

4.2 Exponential model relating f' and N_α for fixed $P_R(N_\alpha)$

In order to develop CSA_{AT-ID} that dynamically controls its objective targets, we need to know the relationship between f', the degree of objective relaxation, and N_α, the number of probes in one run of CSA, for a fixed $P_R(N_\alpha)$. In this section we find this relationship by studying the statistical behavior in evaluating four continuous NLPs by CSA.

Figure 2 shows a 3-D graph relating the parameters in solving (6), in which $P_R(N_\alpha)$ was obtained by running CSA 200 times for each combination of N_α and f'. It shows an exponentially decreasing relationship between f' and N_α at

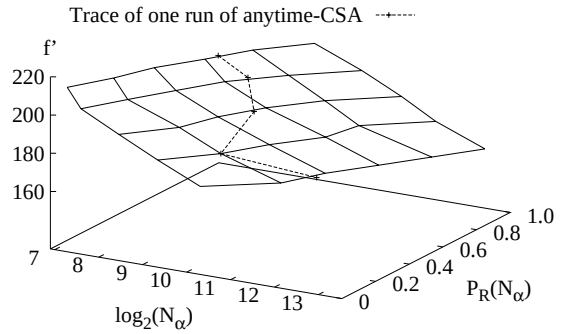


Fig. 2. A 3-D graph showing an exponentially decreasing relationship between f' and N_α and a monotonically non-decreasing relationship between $P_R(N_\alpha)$ and N_α when CSA is applied to solve (6). The dotted line shows the trace taken in a run of CSA_{AT-ID}.

Table 2. The averages and standard deviations of coefficient of determination R^2 on linear fits of f' and $log_2(N_\alpha)$ for fixed $P_R(N_\alpha)$.

Benchmark	Mean(R^2)	Std. Dev.(R^2)
G1 [10]	0.9389	0.0384
G2 [10]	0.9532	0.0091
Rastrigin	0.9474	0.0397
Problem 5.2 [5]	0.9461	0.0342

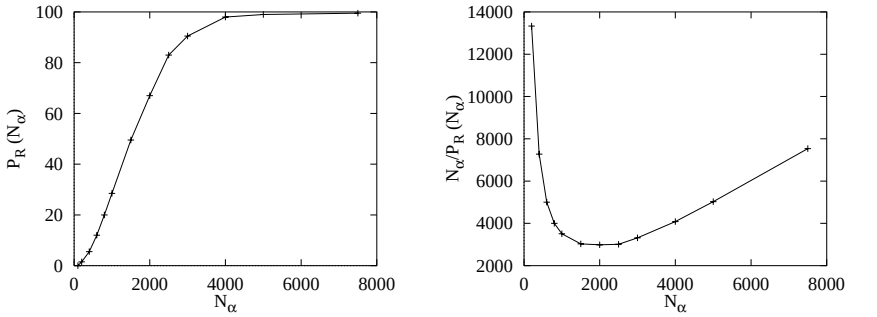
fixed $P_R(N_\alpha)$ and a monotonically non-decreasing relationship between $P_R(N_\alpha)$ and N_α at fixed f'. These observations lead to the following exponential model:

$$N_\alpha = ke^{-af'} \quad \text{for fixed } P_R(N_\alpha) \text{ and positive real constants } a \text{ and } k. \quad (12)$$

To verify statistically our proposed model, we performed experiments on several benchmarks of different complexities: G1, G2 [10], Rastrigin (6), and Floudas and Pardalos' Problem 5.2 [5]. For each problem, we collected statistics on f' and N_α at various $P_R(N_\alpha)$, regressed a linear function on f' and $log_2(N_\alpha)$ to find a best fit, and calculated the coefficient of determination R^2 of the fit. Table 2 summarizes the average and standard deviation of R^2 of the linear fit for each test problem, where R^2 very close to 1 shows a good fit. Since R^2 has averages very close to one and has small standard deviations, f' is verified to be exponential with respect to N_α at fixed $P_R(N_\alpha)$.

4.3 Sufficient conditions for the existence of $N_{\alpha_{opt}}$

In order for $N_{\alpha_{opt}}$ to exist at fixed f', $\frac{N_\alpha}{P_R(N_\alpha)}$ in (5) must have an absolute minimum in $(0, \infty)$. Such a minimum exists if $P_R(N_\alpha)$ satisfies the following sufficient conditions: a) $P_R(0) = 0$ and $\lim_{N_\alpha \to \infty} P_R(N_\alpha) = 1$, and b) $P_R''(0) > 0$. We do not show the proof of these conditions due to space limitation.



a) $P_R(N_\alpha)$ satisfies the two sufficient conditions b) Absolute minimum in $\frac{N_\alpha}{P_R(N_\alpha)}$

Fig. 3. An example showing the existence of an absolute minimum in $\frac{N_\alpha}{P_R(N_\alpha)}$ when CSA was applied to solve (6) with $f' = 180$. ($N_{\alpha_{opt}} \approx 2000$.)

We collected statistics on $P_R(N_\alpha)$ and N_α at various f' for each of the four test problems studied in Section 4.2. The results indicate that $P_R(N_\alpha)$ satisfies the two sufficient conditions, implying that $\frac{N_\alpha}{P_R(N_\alpha)}$ has an absolute minimum in $(0, \infty)$. In other words, each of these problems has an optimal cooling schedule $N_{\alpha_{opt}}$ that minimizes $\frac{N_\alpha}{P_R(N_\alpha)}$ at fixed f'. Figure 3 illustrates the existence of such an optimal schedule in applying CSA to solve (6) with $f' = 180$. The experimental results also show that $P_R(N_\alpha)$ is monotonically nondecreasing.

Note that there is an exponential relationship between $P_R(N_\alpha)$ and N_α in part of the range of $P_R(N_\alpha)$ (say between 0.2 and 0.8) in the problems tested. We do not exploit this relationship because it is not required by the iterative deepening strategy studied in the next section. Further, the relationship is not satisfied when $P_R(N_\alpha)$ approaches 0 or 1.

It is interesting to point out that the second sufficient condition is not satisfied when searching with random probing. In this case, $P_R(N_\alpha) = 1 - (1 - \frac{1}{S})^{N_\alpha}$, and $P_R^{''}(0) = -ln^2(1 - \frac{1}{S}) < 0$, where S is the number of states in the search space. Hence, $\frac{N_\alpha}{P_R(N_\alpha)}$ at fixed f' does not have an absolute minimum of N_α in $(0, \infty)$.

5 Anytime CSA with Iterative Deepening

We propose in this section CSA_{AT-ID} with two components. In the first component discussed in Section 5.1, we design a set of cooling schedules for multiple runs of the original CSA so that (7) is satisfied; that is, the average total number of probes to find a CLM of value f' or better is of the same order of magnitude as $T_{opt}(f')$. In the second component presented in Section 5.2, we design a schedule to decrease objective target f' in CSA_{AT-ID} that allows it to find f^* using an average total number of probes of the same order of magnitude as $T_{opt}(f^*)$.

CSA_{AT-ID} in Figure 4 first finds low-quality feasible solutions in relatively small amounts of time. It then tightens its requirement gradually, tries to find a solution at each quality level, and outputs the best solution when it stops.

```
1.  procedure anytime-CSA
2.      set initial target of solution quality at f' = ∞;
3.      set initial cooling rate α = α_0;
4.      set K = number of CSA runs at fixed α and f (typically K = 3);
5.      repeat /* Lines 5-15 are for gradually improving solutions of value f' */
6.          repeat /* Lines 6-12 are for generating a solution of quality f' */
7.              for i ← 1 to K do
8.                  evaluate CSA with transformed objective F(f(x), f') and α;
9.                  if CSA succeeded then goto 13; end_if
10.             end_for
11.             increase cooling schedule N_α ← ρ × N_α (typically ρ = 2);
12.         until number of probes in target f' exceeded 10 times
                the number of probes in previous target level;
13.         if (f' == ∞) then f' = f_0; end_if /* f_0 is first fesasible solution found */
14.         reduce target level f' ← f' − c;
15.     until no better solution was found in two successive decreases of f';
16. end_procedure
```

Fig. 4. CSA_{AT-ID}: Anytime-CSA procedure with iterative deepening. The only problem-dependent run-time information used is f_0.

It is important to point out that CSA_{AT-ID} does not use regression at run time in order to find the values of parameters of (12). One reason is that these problem-dependent parameters are hard to estimate. Rather, CSA_{AT-ID} exploits the exponential relationship between f' and N_α and the monotonically nondecreasing relationship between $P_R(N_\alpha)$ and N_α in order to derive a set of CSA runs with different parameters. The only run-time information used in CSA_{AT-ID} is f_0, the value of the first feasible solution found with an initial objective target of $f' = \infty$.

5.1 Finding a solution using increasing cooling schedules

Lines 6-12 in Figure 4 are used to evaluate CSA using a set of cooling schedules, each involving multiple runs of CSA, in order to carry out iterative deepening [9] and to achieve geometric growth in the number of probes in successive schedules. By choosing an appropriate number of runs under each cooling schedule, we like to show that the total average overhead over all the schedules is dominated by that of the last schedule and is of the same order of magnitude as the average overhead of multiple run of the original CSA with the best cooling schedule.

Our approach in Lines 6-12 of Figure 4 starts with an objective target $f' = \infty$ and a cooling rate $\alpha = \alpha_0$, corresponding to a fast cooling schedule $N_0 = N_{\alpha_0}$. We propose to use a set of geometrically increasing cooling schedules:

$$N_i = \rho^i N_0, \qquad i = 0, 1, \ldots \tag{13}$$

where N_0 is the (fast) initial cooling schedule. Under each cooling schedule, CSA is run multiple times for a maximum of K times but stops immediately when a solution is found. For iterative deepening to work, $\rho > 1$.

Let $P_R(N_i)$ be the reachability probabilities of one run of CSA under cooling schedule N_i. Let $\beta(f')$ be the expected total number of probes taken by Lines 6-12 in Figure 4 to find a solution with objective target f' starting from schedule N_0, and $B_{opt}(f')$ be the expected total number of probes taken by the original CSA with optimal $N_{\alpha_{opt}}$ to find a solution of similar quality. According to (5),

$$B_{opt}(f') = \frac{N_{\alpha_{opt}}}{P_R(N_{\alpha_{opt}})} \tag{14}$$

The following theorem shows the sufficient conditions in order for $\beta(f')$ to be of the same order of magnitude as $B_{opt}(f')$. Due to space limitation, we do not show the proof here.

Theorem 3. *$\beta(f') = O(B_{opt}(f'))$ if*

a) $P_R(N_\alpha)$ is monotonically non-decreasing for N_α in $(0, \infty)$;
b) $P_R(0) = 0$, and $\lim_{N_\alpha \to \infty} P_R(N_\alpha) = 1$;
c) $P_R''(0) > 0$; and
d) $(1 - P_R(N_{\alpha_{opt}}))^K \rho < 1$.

The proof is not shown due to space limitations. Typically, $\rho = 2$, and in all the benchmarks tested, $P_R(N_{\alpha_{opt}}) \geq 0.25$. Substituting these values into the last condition in Theorem 3 yields $K > 2.4$. In our experiments, we have used $K = 3$. Since a maximum of three runs of CSA are done under each cooling schedule, $B_{opt}(f')$ is of the same order of magnitude as *one* run of CSA with the optimal cooling schedule.

5.2 Anytime search using decreasing objective targets

After finding a solution of quality f' using Lines 6-12 in Figure 4, Line 14 adjusts f' to a new objective target so that better solutions will be found if more time is allowed. (If this were the first time that a feasible solution was found, then Line 13 updates f' to f_0, the value of first feasible solution with an initial objective target of $f' = \infty$.) Based on the exponential model in (12) and the principle of iterative deepening [9], the average number of probes to find a solution of value f' grows geometrically if f' is decreased using the following *linear schedule*:

$$f_{j+1} = f_j - c, \text{ where } c \text{ is a positive constant.} \tag{15}$$

In our experiments, we estimate c to be 10% of f_0.

Let $\gamma(f_n)$ be the expected total number of probes CSA_{AT-ID} takes to find f_n, using objective targets $f_0, f_1, \ldots, f_n$. The following theorem proves the relative complexities of $\gamma(f_n)$ and $B_{opt}(f_n)$.

Theorem 4. $\gamma(f_n) = O(B_{opt}(f_n))$.

Table 3. Comparison between $\overline{T}_{AT-ID}$ and $\overline{T}_{CSA}$ in solving four constrained NLPs with transformed objective $F(f(x), f')$. $\overline{N}_{\alpha_{succ}}$ is the average length of the cooling schedule when a desirable solution was found using Lines 6-12 of CSA_{AT-ID}. $N_{\alpha_{opt}}$ is the length of the optimal cooling schedule for the target objective.

Problem ID	f^*	Target f'	CSA_{AT-ID}			CSA			$\frac{\overline{T}_{AT-ID}}{\overline{T}_{CSA}}$
			N_0	$\overline{N}_{\alpha_{succ}}$	$\overline{T}_{AT-ID}$	$N_{\alpha_{opt}}$	$P_R(N_{\alpha_{opt}})$	$\overline{T}_{CSA}$	
G2	-0.8036	-0.803	220	4640	9.176	5800	0.56	7.375	1.244
Rastrigin	162.6630	163.0	200	4560	3.964	3400	0.43	2.010	1.972
5.2	1.5670	1.7	2310	108100	595.662	136400	0.55	208.211	2.861
7.3	0.9705	1.3	450	515840	1105.948	829440	0.60	555.916	1.989

Proof. According to (12), and (14),

$$B_{opt}(f_i) = \frac{N_{\alpha_{opt}}}{P_R(N_{\alpha_{opt}})} = \Theta(e^{-af_i}). \tag{16}$$

Hence, using the result in Theorem 3,

$$\begin{aligned}\gamma(f_n) &= \sum_{i=0}^{n} \beta(f_i) = \sum_{i=0}^{n} O(B_{opt}(f_i)) = \sum_{i=0}^{n} O(e^{-af_i}) \\ &= \sum_{i=0}^{n} O(e^{-a(f_0 - ic)}) = O(e^{-a(f_0 - nc)}) = O(B_{opt}(f_n))\end{aligned} \tag{17}$$

The theorem shows that, despite finding solutions of intermediate quality defined by a linear sequence of improving objective targets, the overall complexity is dominated by that in finding solutions to the last objective target f_n. In particular, we have established (7) by showing that $T_{AT-ID}(f^*) = O(T_{opt}(f^*))$.

6 Experimental Results

We tested CSA_{AT-ID} on four continuous constrained NLPs of different sizes and degrees of difficulty. G2 [10] and Rastrigin (6) are relatively easy NLPs with multiple feasible regions. In particular, (6) is characterized by a large number of deep infeasible local minima in the objective function. Finally, Floudas and Pardalos' Problems 5.2 and 7.3 [5] are large and difficult NLPs with many equality constraints. Although our experiments were on continuous NLPs, similar performance is expected for discrete and mixed-integer constrained NLPs.

Table 3 compares $\bar{T}_{AT-ID}$, the average time taken by Lines 6-12 in Figure 4 to obtain a CGM to the four benchmark NLPs with transformed objective $F(f(x), f')$, and $\bar{T}_{CSA}$, the average time of CSA with an optimal cooling schedule $N_{\alpha_{opt}}$ for the objective target. The results verify the analysis in Section 5.1 and show that the two averages are related by a (small) constant factor.

Figure 5 compares the anytime behavior of CSA_{AT-ID} and the original CSA in terms of solution quality and execution time. The anytime performance of the original CSA was found by running CSA using the same cooling schedule

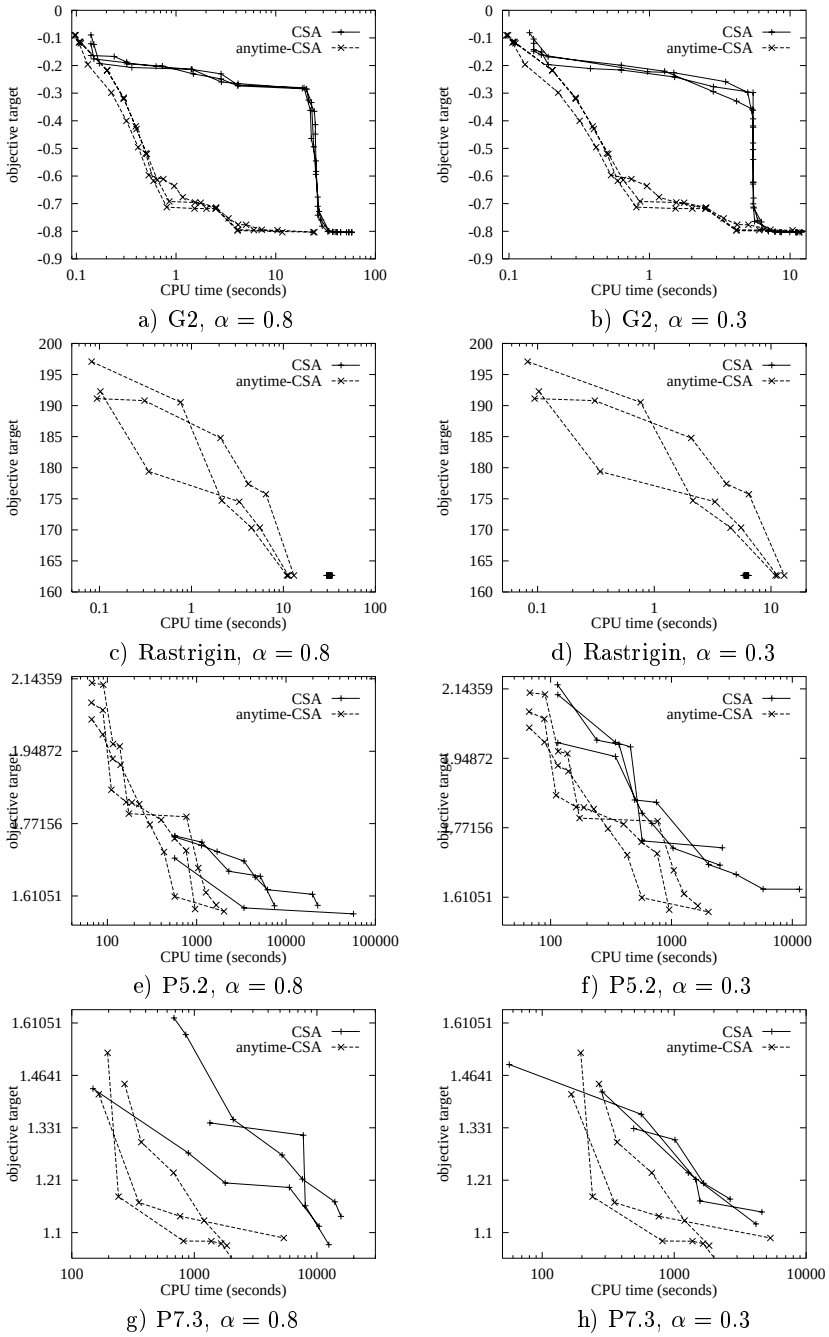


Fig. 5. Performance comparison of CSA_{AT-ID} and CSA in solving four continuous constrained minimization NLPs. CPU times were measured on Pentium 500-MHz computers running Solaris 2.7. α is the cooling rate of the original CSA.

multiple times until a CGM was found. Without knowing its optimal schedule, we tried two geometric schedules with $\alpha = 0.3$ and $\alpha = 0.8$, respectively. CSA and CSA_{AT-ID} were each ran from three random starting points.

In general, the results show that CSA_{AT-ID} performs substantially better than the original CSA as an anytime algorithm. When compared against a given amount of time, CSA_{AT-ID} found much better suboptimal solutions than CSA. When compared against solutions of the same quality, CSA_{AT-ID} took between one to two orders less CPU time than CSA.

References

1. E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* J. Wiley and Sons, 1989.
2. E. H. L. Aarts. A new polynomial-time cooling schedule. *Proc. of the IEEE Int. Conf. on CAD-85*, pages 206–208, 1985.
3. J. Bernasconi. Low autocorrelation binary sequences: Statistical mechanics and configuration space analysis. *J. Physique*, 48(4):559–567, 1987.
4. M. F. Cardoso, R. L. Salcedo, and S. F. de Azevedo. Non-equilibrium simulated annealing: A faster approach to combinatorial minimization. *Industrial Eng. Chemical Research*, 33:1908–1918, 1994.
5. C. A. Floudas and P. M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455 of *Lecture Notes in Computer Science.* Springer-Verlag, 1990.
6. B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.
7. L. Ingber and N. Rosen. Genetic algorithms and very fast simulated re-annealing: A comparison. *J. Math. Comput. Modelling*, 16:87–100, 1992.
8. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
9. R. Korf. Heuristics as invariants and its application to learning. *Machine Learning*, pages 100–103, 1985.
10. Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.
11. C. N. Potts and L. N. V. Wassenhove. Single machine tardiness sequencing heuristics. *IIE Transactions*, 23:346–354, 1991.
12. S. Rees and B. C. Ball. Citeria for an optimal simulated annealing schedule for problems of the traveling salesman type. *J. Physics*, 20(5):1239–1249, 1987.
13. H. Szu and R. Hartley. Fast simulated annealing. *Phys. Lett. A*, 122(3-4):157–162, 1987.
14. B. W. Wah and T. Wang. Simulated annealing with asymptotic convergence for nonlinear constrained global optimization. *Principles and Practice of Constraint Programming*, pages 461–475, Oct. 1999.
15. B. W. Wah and Z. Wu. The theory of discrete Lagrange multipliers for nonlinear discrete optimization. *Principles and Practice of Constraint Programming*, pages 28–42, Oct. 1999.
16. Y. Wang, W. Yan, and G. Zhang. Adaptive simulated annealing for optimal design of electromagnetic devices. *IEEE Trans. on Magnetics*, 32(3):1214–1217, 1996.
17. Z. Wu. *The Theory and Applications of Nonlinear Constrained Optimization using Lagrange Multipliers.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, Aug. 2000.