Annealing a Genetic Algorithm over Constraints

Susan E. Carlson

Department of Mechanical, Aerospace
and Nuclear Engineering
Thornton Hall, McCormick Road
University of Virginia
Charlottesville, Virginia 22903-2442
e-mail: sec3a@virginia.edu

R. Shonkwiler

School of Mathematics
Georgia Institute of Technology
Atlanta, GA. 30332
e-mail: shenk@math.gatech.edu

Abstract

A class of variable fitness Genetic Algorithms is studied as a technique for use on constrained optimization problems. Fitness is taken as the product of the objective with an “attenuation” factor which is 1 for feasible solutions but some variable fraction of 1 for infeasible ones. It is shown that this technique leads to algorithms which converge in probability to globally optimal, feasible solutions.

An application of the technique is made to a problem of engineering interest with excellent results: the Ground Water Treatment problem for unconfined aquifers.

1 Introduction

Genetic algorithms are widely used for unconstrained optimization problems with general agreement about the fundamentals of the implementation. But their application to constrained optimization problems remains unsettled. However, many optimization problems, especially those in engineering design, are highly constrained. For such problems, it is necessary to find global optima not violating any constraint. In general, constraints may be linear or non-linear, and equality or inequality in type.

Two main approaches have been tried for overcoming the difficulties of handling constraints in genetic algorithms. One is to allow only feasible solutions in the population, and the other is to apply a penalty to those solutions that violate constraints. Unfortunately, neither has proved to perform satisfactorily in general.

The first approach, excluding infeasible solutions, can be arranged either by discarding them as they arise or

by using specialized operators that maintain feasibility. This technique is very effective on certain specialized problems. If the fraction of infeasible solutions is small, then discarding them as they appear is efficient. But, if infeasible solutions arise too frequently, then the algorithm may spend significant amounts of time looking for those few solutions that do not violate constraints.

On the other hand, by incorporating constraints directly into the genetic operators, feasibility can be maintained. This approach is possible, for example, in the case of linear equality constraints. However, for many constrained problems, it is too hard, costly, or even impossible to maintain feasibility.

The most prevalent technique for coping with infeasible solutions is to penalize a population member for constraint violation. In this way, penalty functions artificially create an unconstrained optimization problem. Traditionally, the weighting of a penalty for a particular problem constraint is based on judgment. Often, the algorithm must be rerun several times before a combination of penalties is found that allows infeasible solutions to die and feasible solutions to flourish. If a set of penalties is too harsh, then the few solutions found that do not violate constraints quickly dominate the mating pool and yield sub-optimal solutions. A penalty that is too lenient can allow infeasible solutions to flourish as they can have higher fitness values than feasible solutions.

The main difficulty then in applying penalty functions is that they are problem dependent. In this paper, we present an approach to the application of penalty functions which is problem independent and generally applicable. In this proposed approach, a variable penalty function is used which is related to the temperature schedule used in simulated annealing algorithms. The penalty is very lenient at the start of the algorithm, but grows progressively harsher as the algorithm runs.

In the early stages of the algorithm, infeasible solutions can act as a “bridge” across isolated clusters of feasible solutions as an aid to searching for the globally optimal solution. On the other hand, because the penalty varies over a run, penalty values which are in tune for the problem will eventually occur resulting in dramatic progress toward the optimum, see Figures 3-5. That there is such a special penalty value is shown clearly in these figures.

The method of penalty variation proposed here is compatible with the IIP (independent, identical processing) parallelization technique and is superlinearly sped-up under parallel execution, [10].

2 A Variable Fitness GA

Let f denote the objective function, to be maximized, which we will assume is non-negative valued throughout its domain D . We will take the fitness function ϕ of the GA to be the product of f and an *attenuation factor* $\alpha(\cdot, \cdot)$ which depends on two parameters, M and T ,

$$\phi(x) = \alpha(M, T)f(x). \quad (1)$$

The first, $M \geq 0$, measures the extent of constraint violation and is zero in the absence of any violation. The second parameter, referred to as *temperature* $T > 0$, is a function of the running time of the algorithm; T tends to 0 (or small values) as execution proceeds. When the GA begins, we want the penalty for constraint violation to be small, or, in terms of attenuation, we want $\alpha \approx 1$, in order that the algorithm be able to utilize infeasible states as needed to find a global maximum. But toward the end of execution we want α to be zero or nearly zero since infeasible solutions are unacceptable. A function which has these properties is

$$\alpha(M, T) = e^{-M/T}. \quad (2)$$

If no constraint is violated, independent of the value of T , then $\alpha = 1$ and fitness is the unattenuated objective value. On the other hand, when T is large (relative to a non-zero M) then $\alpha \approx 1$. But as $T \rightarrow 0$, then $\alpha \rightarrow 0$ as well and hence, by equation (1), fitness tends to zero too. Thus infeasible solutions should be excluded from the GA populations at the end of a run.

An overview of the main iteration loop of the algorithm used in the applications is presented next along with an explanation of its steps. A proof that it is possible to lower the temperature in such a way that converges to a globally optimal solution follows similar to the techniques used in [3] and [5].

1. select an initial population

2. note the solution having the best fitness
3. note the solution having the best objective among feasible solutions
4. initialize temperature
5. Loop
 6. perform GA mutation/mating on the population
 7. update temperature
 8. recalculate attenuation factors and update fitnesses for infeasible solutions
 9. update the best fitness seen so far
 10. update the best objective among feasible solutions seen so far
 11. check for an exit condition
 12. cull the population back to original size
13. EndLoop
14. report the best feasible solution

Step 1. The initial population is chosen at random meaning every possible population is equally likely to be selected.

Steps 2/3. We keep track of the best fitness value and the best feasible objective value to occur over the course of the run for two reasons. First, these could be lost from the population pool in the culling step, (Step 12) during the normal course of the run and consequently go unreported as an output of the run. Second, the progression of these values can be used as part of a mathematical criteria to terminate the algorithm, see Step 11.

Step 4. We are not aware of definitive results about selecting an initial temperature. If there were to be infinitely many iterations, then any positive starting temperature is permissible. Of course this is impossible, a run has only finitely many iterations to work with. Consequently the final temperature will be bounded away from zero, and the initial temperature matters. We give some empirical results on the effect of the starting temperature for the two applications reported. As a rule of thumb, we take the starting temperature to be on the same order of magnitude as the mean constraint violation $\bar{M}$.

Step 5. This is the generation by generation loop of mating, mutation, and replacement common to all GA's.

Step 6. Much has been written about GA mutation and mating operations, we have our favorites. Whatever the details, the population should be expanded in size at this point of the algorithm to include every newly generated solution. Thus if solution X is used as a source for mutations and solution Y is the result, then both X and Y are kept at this point of the algorithm.

Step 7. A standard temperature updating schedule is reciprocal logarithmic, i.e. $O(1/\log(t))$, which can be shown to give convergence to an optimal solution as iteration count t tends to infinity. However, recently it has been shown [1] that under the imposition of a fixed number of iterations, harmonic cooling, $O(1/t)$, is better. However, we prefer *root-2* cooling in which $T = O(1/\sqrt{t})$. The question then becomes, what should the final temperature be? Like the question of starting temperature, we do not know a canonical answer, but in the applications we show some empirical results. As a rule of thumb, we take the final temperature to be on the order of one one-hundredth of the mean constraint violation $\bar{M}$. Alternatively, we can compare the progression of the best observed fitness with the best observed feasible solution during a run. When these two match, it means the penalty imposition has driven the algorithm out of the infeasible region whereupon the temperature is sufficiently low. This is the desirable condition. If runs are ending with a difference between the best fitness solution and best feasible solution, then the final temperature should be taken lower.

Steps 8/9. Since fitness is dependent upon temperature, which is changing over iterations (possibly on each iteration), the fitness of each solution of the population must be updated to reflect this. However only the attenuation factor must be recalculated; by the form of the fitness function, an objective recalculation is not necessary. The same holds for the best observed fitness solution. As a result, the best observed fitness may decrease occasionally but only if it is represented by an infeasible solution.

Step 10. In the culling step, it can happen that the best performer of the population goes away. Hence we keep track of the best solution seen over the entire run and report this at the end of the run. At first it might be supposed undesirable that a top performing solution should be culled out, but in fact it gives the GA a chance to explore a different region of the search space D . An often reported phenomena in the GA

literature is the problem of “niching” in which the opposite happens; a top performer, which unfortunately is not the globally optimal solution, tends to dominate the population with its progeny. When niching occurs, further processing is useless. Although it is not strictly necessary to keep track of the best observed fitness as well as the best observed feasible solution, we find that it is useful to do so. As discussed in Steps 7 and 11, the best fitness helps set temperatures and helps decide when to terminate the algorithm. Also, the best fitness solution often only slightly violates constraints and lends insight to the overall optimization problem.

Step 11. Again, a theory for terminating the search is lacking. Several criteria have been proposed, see [4,11]. We use the following: (1) either the pre-determined number of allowed iterations has been reached or (2) there has been no improvement over a certain, pre-determined, succession of iterations. The latter is determined by the best fitness and best objective updating. Each time a new best fitness or a new best feasible solution is found, the *no-improvement* statistic is reset. The rationale for the no-improvement condition is to shut down the search if it has hopelessly niched. From the graphs provided in the applications to follow, it is easy to see how frequently the algorithm makes improvements.

Step 12. As the last step of the cycle, the population size is cut back to its original, pre-mutation/mating level. As with GA mutation and mating, there is a vast literature on how to do this. We use the “Russian roulette” method see [2].

3 Ground Water Management Problem

3.1 Introduction

The contamination of ground water supplies poses widespread and important environmental problems. One solution strategy is “pump and treat” in which wells are installed in the affected area, contaminated ground water pumped out, treated and returned to the environment. The associated engineering problem is deciding how many wells, w , to utilize, where to locate them, (x_j, y_j) , $j = 1, \dots, w$, and how much to extract, Q_j , $j = 1, \dots, w$, from each. These are the decision variables. To assess the effectiveness of the method, several survey stations (wells) called *elements* are used to determine whether ground water is escaping the treatment. Mathematically speaking, imagine a region Ω bounded by a simple closed curve Γ and let $\mathbf{N}$ denote the unit outward pointing normal vector to Γ . Then, in order that no water escape treatment,

the flow vector $\mathbf{v}$ at each point on the boundary must satisfy

$$\mathbf{v} \cdot \mathbf{N} < 0. \quad (3)$$

Fig. 1, Velocity Field for Optimal Well Locations (44 and 45)

Of course, many choices of decision variables lead to favorable modifications of the groundwater flow so that equation (3) is satisfied. It therefore becomes a question of how it can be done in the most cost effective way, for example, with minimum extraction. When posed as an optimization problem, equation (3) constitutes a system of constraints, one for each element, perhaps m in number.

In addition, to velocity constraints, there are also *head* constraints. Extraction at a given well reduces the head at that well, this is called *drawdown* S . But the head cannot be reduced below some limiting value. Thus the drawdown S_j at well j , which depends jointly on the several extraction rates Q_i , $i = 1, \dots, w$, cannot exceed a value which depends on the undisturbed head at that well, h_i^{ud} .

In this study, we fix the number of wells and their locations, although we have used more wells than is necessary, and optimize on the extraction rates. Mathematically, the problem to be optimized is this:

$$\begin{aligned} & \text{minimize } \sum_{j=1}^w Q_j & (4) \\ \text{subject to: } & S_j \leq S_j^{max}, \quad j = 1, \dots, w & (5) \\ & \text{and } v_i \cdot \mathbf{N}_i < 0, \quad i = 1, \dots, m. & (6) \end{aligned}$$

In this, the S_j^{max} , $j = 1, \dots, w$, are the given head constraints and, as above, $\mathbf{N}_i$ is the unit outward normal to the containment curve Γ (at the i th element).

3.2 The constrained optimization problem

Genetic Algorithm solutions have been proposed for fixed well locations in an unconfined aquifer using

Fig. 2, Piezometric Head Distribution for Optimal Well Locations

time-fixed penalty functions, [6]. However, as noted above, the construction of such a penalty function requires some experience with the problem in order to tune its parameters. Further, we will show in this example, besides obviating the need for tuning, better final results are attainable with a time varying penalty approach. Under the assumptions mentioned above, the linearized form of this aquifer problem is in fact a linear program; consequently it is possible to calculate the exact solution by deterministic methods. However, by allowing the number of wells and their locations to vary, the mathematical nature of the problem changes and it is no longer a linear program. In this work however, we test the behavior of our varying penalty technique on the fixed location problem. The optimization problem is solved for a confined aquifer region and incorporates the capture segment method of Ratzlaff [9].

The particular problem we solve is a rectangular region Ω subdivided into an 8 by 11 grid of nodes (a 7 by 10 array of rectangles), each of which is bisected to give 140 triangular finite element regions. We have chosen 30 survey elements located on equally spaced nodes along the periphery of the region, see Figure 1.

One way to incorporate a variable number of extraction wells is to establish more wells than are necessary for an optimal solution. Then, at optimality, the unneeded ones will have 0 pumping rates. In this spirit then, we assume an 8 well solution even though we know the optimal solution requires only two. However, as mentioned above, in this study we fix their locations. The optimal solution is that wells number 44 and 45 pump at the rate 15.96 while the others should not pump at all, for a total extraction rate of 31.92. This solution is consistent with the constraints; the optimal piezometric head distribution is shown in Figure 2 and the corresponding velocity field is shown in Figure 1.

3.3 Implementing a variable penalty GA

In this problem we want to minimize the combined extraction rates, so our first step is to convert this minimization problem into a maximization one. Since the pumping rates can never exceed the total head available at each well, we take the GA objective function to be the difference

$$f(\text{decision variables}) = \text{BASE} - \sum_{j=1}^w Q_j \quad (7)$$

where BASE is the sum of the available heads at each well. When pump rates are high, f will be low and conversely. Since the locations are fixed, the decision variables here are exactly the w extraction rates.

As discussed in Section 2, for the variable penalty feature, we take fitness ϕ to be the product of the attenuation factor $\alpha = e^{-M/T}$ times the GA objective

$$\phi = \alpha f. \quad (8)$$

To calculate M , we simply add the values $v_i \cdot N_i$ for those i giving a positive result and the values $S_j^{max} - S_j$ for those j giving a positive result,

$$M = \sum_{i=1}^m [v_i \cdot N_i]^+ + \sum_{j=1}^n [S_j^{max} - S_j]^+ \quad (9)$$

where $[\cdot]^+$ means the positive part.

3.4 Results

Fig. 3 Convergence, root-2 cooling, $T_{\text{start}} = 1000$

We show convergence profiles for various cooling systems. All results presented here are averages over 5 runs for each experiment. Runs were terminated after 20,000 iterations (which required about 3 minutes on a workstation). For each experiment, the individual convergence profiles of the several runs were quite similar and variance was small. The sole exception to

this was the fixed high temperature experiments. In their case, about one-half of the runs did not find any feasible solution after 20,000 iterations.

Fig. 4 Convergence, root-2 cooling, $T_{\text{start}} = 10000$

Before performing any experiments, we calculated 100 random fitness values to approximate the average constraint violation. Of the 100 values, the average was 3,710.12; all 100 violated the constraints. Next, root-2 cooling was tried with a starting temperature of 1000. Ending temperature was about 8. Average convergence is shown in Figure 5. The algorithm came within 1 of the global minimum by iteration 10000. At this point the temperature was about 10. After that not much improvement was made, the final value was .9 above the global minimum.

Fig. 5 Convergence, logarithmic cooling, $T_{\text{start}} = 1000$

Then root-2 cooling was tried with a starting temperature of 10,000 and an ending temperature of 70. As seen in the convergence profile in Figure 6, convergence to within 1 of the global minimum occurred at 7000 iterations. This corresponded to a temperature of about 119. The final objective value came within

about .3 of the goal.

Fig. 6 Convergence, fixed temperatures.

Logarithmic cooling was tried with a starting temperature of 1000 and a resulting ending temperature of 100. As seen in Figure 7, convergence to within 1 came at about 4000 iterations, when the temperature was 120. The final value was just 0.15 above the global minimum.

Finally runs were made at the fixed temperatures of 10000, 1000, and 100. The higher temperatures failed to come close to the optimum. At a temperature of 100, convergence to within 1 occurred after approximately 1000 iterations, but almost no further improvement was seen. See Figure 8. This shows that once penalties are tuned to the problem, the fixed penalty method does reasonably well.

4 Conclusions

The variable fitness Genetic Algorithm was found to be an effective method of solving constrained optimization problems. This approach does not require experimentation or an a priori knowledge about the problem to assign penalty function weights.

References

1. Azencott, R., *Simulated Annealing, Parallelization Techniques*, John Wiley and Sons, New York (1992).
2. Carlson, Susan E., Shonkwiler, R., Ingram, M., "Component Selection Using Genetic Algorithms," *Advances in Design Automation 1993 Proceedings*. Vol. 1, ASME, New York, (1993) 471-476.
3. Catoni, O., "Sharp Large Deviations Estimates for Simulated Annealing Algorithms," *Ann. Inst.*

Henri Poincaré, Probabilités et Statistiques, **27**, 3(1991) 291-383.

4. Dorea, C.C.Y., "Stopping rules for a random optimization method," *SIAM J Control and Optimization*, **28**, 4(1990) 841-850.
5. Hajek, B., "Cooling Schedules for Optimal Annealing," *Math of Operations Research*, **13** (Feb. 1988).
6. McKinney, Daene C. and Lin, Min-Der, *Genetic Algorithm Solution of Groundwater models*, Vol. 30, No. 6, pp. 1897-1906, June 1994.
7. Michalewicz, Z. and Attia, N., "Evolutionary Optimization of Constrained Problems," Preprint (1995).
8. Powell, David and Skolnick, M., "Using Genetic Algorithms in Engineering Design Optimization with Non-Linear Constraints," *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, (1993), 426-431.
9. Ratzlaff, S. A., Aral, M. M., Al-Khaziyal, F., "Optimal design of groundwater capture systems using segmented velocity-direction constraints," *Groundwater*, **30**(4), (1992) 607-612.
10. Shonkwiler, R. and Van Vleck, E., "Parallel Speed-up of Monte Carlo Methods for Global Optimization," *J. of Complexity* **10**(1994) 64-95
11. Zuckerman, D., "Optimal stopping in a continuous search model," *J. Appl. Prob*, **23** 514-518(1986).