Genetic Algorithms and Optimal Control Problems

Zbigniew Michalewicz* Jacek B. Krawczyk† Mohammad Kazemi‡
Cezary Z. Janikow§

29th CDC

Abstract

This paper studies the application of the genetic algorithm to discrete-time optimal control problems. Numerical results obtained here are compared with a system for construction and solution of large and complex mathematical programming models, GAMS. While GAMS appears to work well only for linear quadratic optimal control problems or problems with short horizon, the genetic algorithm applies to more general problems equally well.

1 Introduction

This paper studies the numerical optimization of discrete-time dynamic control systems. As it is well known, the task of designing and implementing algorithms for the solution of optimal control problems is a difficult one. The highly touted dynamic programming is a mathematical technique that can be used in variety of contexts, particularly in optimal control (*cf.* [Bertsekas, 1987]). However, this algorithm breaks down on problems of moderate size and complexity, suffering from what is called the "curse of dimensionality" by its developer (see [Bellman, 1957]).

In particular, optimal control problems are quite difficult to deal with numerically. Some numerical dynamic optimization programs available for general users are typically offspring of the static packages [Brooke et al., 1988] and they do not use dynamic-optimization specific methods. Thus the available programes do not make an explicit use of the Hamiltonian, transversality conditions, *etc.* On the other hand, if they did use the dynamic-optimization specific methods, they would be even more difficult to be handled by a layman.

Genetic algorithms, which will be explained in Section 2 of this paper, require little knowledge of the problem itself. Therefore computations based on these algorithms are attractive to users without the numerical optimization background. Genetic algorithms have been quite successfully [Goldberg, 1989], [DeJong, 1985], [Vignaux & Michalewicz, 1989a,b] applied to static optimization problems like wire routing, scheduling, transportation problem, travelling salesman problem, *etc.*, but to the best of authors' knowledge they have not been applied to optimal

*Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA
†Faculty of Commerce and Administration, Quantitative Studies Group, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
‡Department of Mathematics, University of North Carolina, Charlotte, NC 28223, USA
§Department of Computer Science, University of North Carolina, Chapel Hill, NC, 27599, USA

control problems. The aim of the present paper is to make an effort in filling this gap. In our study, to better evaluate the performance of genetic algorithm, we use a version (called *Student Version*) of a computational package for construction and solution of large and complex mathematical programming models, called GAMS ([Brooke et al., 1987]). In the rest of the paper we will refer to this system as GAMS only.

The remainder of this paper is organized as follows. Section 2 gives an overview of genetic algorithms. In Section 3 two simple optimal control problems are formulated, and solved analytically so that the reference points for comparisons are available. In Section 4 the results of application of the genetic algorithm to the control problems are presented. In Section 5 the analytical solutions of the test problems are presented, and the genetic algorithm performance is compared with that of GAMS. Section 6 provides some concluding remarks.

2 Genetic Algorithms

Genetic algorithms ([Davis, 1987], [De Jong, 1985], [Goldberg, 1985], [Holland, 1975]) are a class of probabilistic algorithms which begin with a population of randomly generated candidates and "evolve" towards a solution by applying "genetic" operators, modelled on genetic processes occurring in nature. As stated in Davis (1987):

> "... the metaphor underlying genetic algorithms is that of natural evolution. In evolution, the problem each species faces is one of searching for beneficial adaptations to a complicated and changing environment. The 'knowledge' that each species has gained is embodied in the makeup of the chromosomes of its members. The operations that alter this chromosomal makeup are applied when parents reproduce; among them are random mutation, inversion of chromosomal material, and crossover—exchange of chromosomal material between two parents' chromosomes."

Let us consider the problem of finding a minimum of a function $f(x)$.

The optimization problem may include constraints. In such a case the problem can be handled through minor modifications of the algorithm, such as usual Lagrange multipliers method or other methods (see [Michalewicz & Schell, 1990]).

For a given optimization problem, at each iteration t of a genetic algorithm we will maintain a population of solutions $P(t) = \{x_1^t, \ldots, x_n^t\}$, where x_i^t is a feasible solution, t is an iteration number and n is arbitrarily chosen length of the population. This population would undergo "natural" evolution. In each generation relatively "good" solutions will reproduce; the relatively "bad" solutions will die out, and will be replaced by the offsprings of the former ones. To distinguish between the "good" and "bad" solutions we will use $f(x_i^t)$ which will play a role of the environment (see Figure 1).

During iteration t, the genetic algorithm maintains a population $P(t)$ of some solutions $x_1^t, \ldots, x_n^t$ (the population size n remains fixed for the duration of the computation). Each solution x_i^t is evaluated by computing $f(x_i^t)$, which gives us some measure of "fitness" of the solution (obviously, the lower $f(x_i^t)$, the better). Next, at iteration $t+1$ a new population is formed: we select solutions to reproduce on the basis of their relative fitness, and then the selected solutions are recombined using genetic operators (crossover and mutation) to form a new set of solutions.

```
procedure genetic algorithm
begin
    t = 0
    initialize P(t)
    evaluate P(t)
    while (not termination-condition) do
    begin
        t = t + 1
        select P(t) from P(t − 1)
        recombine P(t)
        evaluate P(t)
    end
end
```

Figure 1: A simple genetic algorithm.

The *crossover* combines the features of two parent structures to form two similar offspring. Crossover operates by swapping corresponding segments of a string of parents. For example, if parents are represented by five-dimensional vectors, say $x_1 = (a_1, b_1, c_1, d_1, e_1)$ and $x_2 = (a_2, b_2, c_2, d_2, e_2)$, then crossing the vectors between the second and the fifth components would produce the offspring $(a_1, b_1, c_2, d_2, e_2)$ and $(a_2, b_2, c_1, d_1, e_1)$.

A *mutation* operator arbitrarily alters one or more components of a selected structure—this increases the variability of the population. Each bit position of each vector in the new population undergoes a random change with the probability equal to the mutation rate, which is kept constant throughout the computation process.

A genetic algorithm to solve a problem must have 5 components:

1. A genetic representation of solutions to the problem;
2. A way to create an initial population of solutions;
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness";
4. Genetic operators that alter the composition of children during reproduction; and
5. Values for the parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, *etc.*).

In our implementation we have used a modified genetic algorithm designed to work on numerical problems. It performed much better than other similar implementations. The details of this algorithm will be presented shortly in a separate publication.

3 Two Optimal Control Problems

Two simple discrete-time optimal control models have been chosen as test problems for the genetic algorithm.

The first is a one-dimensional linear-quadratic model:

$$\min q \cdot x_N^2 + \sum_{k=0}^{N-1}(s \cdot x_k^2 + r \cdot u_k^2) \tag{1}$$

subject to

$$x_{k+1} = a \cdot x_k + b \cdot u_k,\ k = 0, 1, \ldots, N-1, \tag{2}$$

where x_0 is given, a, b, q, s, r are given constants, $x_k \in R$, is the state and $u_k \in R$ is the control of the system.

The second test problem is

$$\max \sum_{k=0}^{N-1} \sqrt{u_k} \tag{3}$$

subject to

$$x_{k+1} = a \cdot x_k - u_k \tag{4}$$

and

$$x_0 = x_N \tag{5}$$

where initial state x_0 is given, a is a constant, and $x_k \in R$ and $u_k \in R^+$ are the state and the (nonnegative) control, respectively.

Let us recall that the value for the optimal performance of (1) subject to (2) is

$$J^* = K_0 x_0^2 \tag{6}$$

where K_k is the solution of the Riccati equation

$$\begin{aligned} &K_k = s + ra^2 K_{k+1}/(r + b^2 K_{k+1}), \text{ and} \\ &K_N = q. \end{aligned} \tag{7}$$

The optimal value J^* of (3) subject to (4) and (5) is (after elementary calulations):

$$J^* = \sqrt{\frac{x_0 \cdot (a^N - 1)^2}{a^{N-1} \cdot (a-1)}} \tag{8}$$

The optimal control and state trajectory can obviously be determined analytically as well.

The value $N = 45$ is chosen as the largest horizon for which a comparative numerical solution from GAMS was still achievable.

Problem (3) subject (4) and (5) will be solved for the following values of N: $N = 2$, $N = 4$, $N = 10$, $N = 20$, and $N = 45$.

In the sequel, the problem (1) subject to (2) will be solved for the following sets of the parameters:

Case	N	x_0	s	r	q	a	b
I	45	100	1	1	1	1	1
II	45	100	10	1	1	1	1
III	45	100	1000	1	1	1	1
IV	45	100	1	10	1	1	1
V	45	100	1	1000	1	1	1
VI	45	100	1	1	0	1	1
VII	45	100	1	1	1000	1	1
VIII	45	100	1	1	1	0.01	1
IX	45	100	1	1	1	1	0.01
X	45	100	1	1	1	1	100

Table 1. Ten test cases.

	Generations							Factor
Case	1	100	1,000	10,000	20,000	30,000	40,000	
I	17807.4	3.27985	1.74689	1.61866	1.61825	1.61804	1.61803	10^4
II	13670.4	5.33177	1.45968	1.11349	1.09205	1.09165	1.09163	10^5
III	17023.8	2.87485	1.07974	1.00968	1.00126	1.00104	1.00103	10^7
IV	15077.3	8.64310	3.75530	3.71846	3.70812	3.70165	3.70160	10^4
V	5956.43	12.2559	2.89769	2.87727	2.87646	2.87570	2.87569	10^5
VI	16657.7	5.07047	2.05314	1.61869	1.61830	1.61806	1.61806	10^4
VII	2680666	19.2684	7.02566	1.63464	1.62412	1.61888	1.61882	10^4
VIII	116.982	67.1758	1.92764	1.00009	1.00005	1.00005	1.00005	10^4
IX	7.18263	4.42849	4.37093	4.31504	4.31024	4.31004	4.31004	10^5
X	9870352	138132	16096.0	1.38244	1.00041	1.00010	1.00010	10^4

Table 2. Genetic Algorithm for problem (1)–(2).

4 Experiments and Results

In this section we present the results of the modified genetic algorithm for optimal control problems.

For problem (1)–(2), each element of the population is a real vector (initialized randomly); during the evolution process the best vectors reproduce (crossover, mutation: see Section 2) and create offspring. Because of the lack of constraints, every offspring is an admissible element of the population.

For each case, we have repeated 3 separate runs of 40,000 generations, and the best of those are reported in Table 2, along with intermediate results at some generation intervals. For example, the values in culumn "10,000" indicates the partial result after 10,000 generations, while running 40,000. It is important to note that such values are worse than those obtained while running only 10,000 generation, due to the nature of some genetic operators.

	Generations						
N	1	100	1,000	10,000	20,000	30,000	40,000
2	6.3310	6.3317	6.3317	6.3317	6.3317	6.3317	6.331738
4	12.6848	12.7127	12.7206	12.7210	12.7210	12.7210	12.721038
8	25.4601	25.6772	25.9024	25.9057	25.9057	25.9057	25.905710
10	32.1981	32.5010	32.8152	32.8209	32.8209	32.8209	32.820943
20	65.3884	68.6257	73.1167	73.2372	73.2376	73.2376	73.237668
45	167.1348	251.3241	277.3990	279.0657	279.2612	279.2676	279.271421

Table 3. Genetic Algorithm for problem (3)–(5).

In the next section we will compare these results with the exact solutions and solutions obtained from the computational package GAMS.

Problem (3)–(5) differs from the first problem in the sense that not every randomly initialized vector $\langle u_0, \ldots, u_{N-1} \rangle$ of positive real numbers generates an admissible sequence x_k (see condition (4)) such that $x_0 = x_N$, for given a and x_0. In our version of genetic algorithm, we have generated a random sequence of $u_0, \ldots, u_{N-2}$, and have set $u_{N-1} = a \cdot x_{N-1} - x_N$. For negative u_{N-1}, we have discarded the sequence and repeated the initialization process: this happened in less than 10% of cases.

The same difficulty occured during the reproduction process. An offspring (after some genetic operations) need not satisfy the constraint: $x_0 = x_N$. In such a case we replaced the last component of the offspring vector u using the formula: $u_{N-1} = a \cdot x_{N-1} - x_N$. Again, if u_{N-1} turns out to be negative, we do not introduce such offspring into new population (again, the number of such cases did not exceed 10%).

Table 3 summarizes our results. Problem (3)–(5) is solved for the following values of N: $N = 2$, $N = 4$, $N = 10$, $N = 20$, and $N = 45$. The population size is fixed at 70; we present also the intermediate values after the first, 100th, 1000th, 10,000th, 20,000th and 30,000th generations.

It appears that in this case, 10,000 generations is sufficient: the improvement in the next 30,000 generations is insignificant.

5 Genetic Algorithms Versus Other Methods

In this section we compare the above results with the exact solutions as well as those obtained from the computational package GAMS.

Exact solutions of the test problems for the values of the parameters specified in Table 1 have been obtained using formulae (6) and (7).

To highliht the performance and competitiveness of the genetic algorithm, the same test problems were solved using GAMS. The comparison may be regarded as not totally fair for the genetic algorithm, since GAMS is based on search methods particularly appropriate for linear-quadratic problems. Thus the problem (1)–(2) must be an easy case for this package. On the

	Exact solution	Genetic Algorithm		GAMS	
Case	value	value	D	value	D
I	16180.3399	16180.3939	0.000%	16180.3399	0.000%
II	109160.7978	109163.0278	0.002%	109160.7978	0.000%
III	10009990.0200	10010391.3989	0.004%	10009990.0200	0.000%
IV	37015.6212	37016.0806	0.001%	37015.6212	0.000%
V	287569.3725	287569.7389	0.000%	287569.3725	0.000%
VI	16180.3399	16180.6166	0.002%	16180.3399	0.000%
VII	16180.3399	16188.2394	0.048%	16180.3399	0.000%
VIII	10000.5000	10000.5000	0.000%	10000.5000	0.000%
IX	431004.0987	431004.4092	0.000%	431004.0987	0.000%
X	10000.9999	10001.0045	0.000%	10000.9999	0.000%

Table 4. Comparison of solutions for the linear-quadratic model.

N	Exact solution	GAMS		GAMS+		Genetic Alg	
		value	D	value	D	value	D
2	6.331738	4.3693	30.99%	6.3316	0.00%	6.3317	0.000%
4	12.721038	5.9050	53.58%	12.7210	0.00%	12.7210	0.000%
8	25.905710	*		18.8604	27.20%	25.9057	0.000%
10	32.820943	*		22.9416	30.10%	32.8209	0.000%
20	73.237681	*		*		73.2376	0.000%
45	279.275275	*		*		279.2714	0.001%

Table 5. Comparison of solutions for the linear-quadratic model. The symbol "*" means that the GAMS failed to find any reasonable value and gave a warning: "The final point is not close to an optimum".

other hand, if for these test problems the genetic algorithm proved to be competitive, or close to, there would be an indication that it should behave satisfactorily in general. Tables 4 and 5 summarise the results, where columns D refer to the percentage of the relative error.

As shown above the performance of GAMS for the linear-quadratic problem is perfect. However, this was not at all the case for the second test problem.

To begin with, none of the GAMS solutions was identical with the analitical one. The difference between the solutions were increasing with the optimization horizon as shown below (Table 5), and for $N > 4$ the system failed to find any value.

It appears that GAMS is sensitive to non-convectness of the optimizing problem and to the number of variables. Even adding an additional constraint to the problem ($u_{k+1} > 0.1 \cdot u_k$) to to restrict the feasibility set so that the GAMS algorithm does not "lose itself"[1]) has not helped much (see column "GAMS+"). As this column shows, for optimization horizons suffciently long there is no chance to obtain a satisfactory solution from GAMS.

[1]This is "unfair" from the point of view of the genetic algorithm which works without such help.

6 Conclusions

In this paper we have initiated the study of the application of the genetic algorithm to discrete-time optimal control problems. A modified version of the genetic algorithm has been applied to two rather simple classes of optimal control problems. The numerical results are compared with those obtained from a search-based computational package (GAMS). It is shown, that while GAMS appears to work well only for linear-quadratic optimal control problems, or problems with short horizon, the genetic algorithm applies to more general problems and appears to be competitive with search-based methods.

Two main conclusions can be drawn from this paper.

The first conclusion is that genetic algorithms can be successfully applied to optimal control problems. In particular, the results which were obtained are encouraging because:

- the closeness of the numerical solution to the analytical one was satisfying,
- the coding and computation efforts (compared to those of a static-optimization version and, in fact, in general) were reasonable (less that 1000 lines of code in C, and, for 40,000 generations, less than 10 minutes of CPU time on CRAY Y-MP).

The second conclusion is that our program based on a genetic algorithm approach resulted in a more reliable version than a "commercial" computation package (GAMS). While the genetic algorithm gave us results comparable with the analytic solutions for both test models, GAMS failed for one of them.

References

[**Bellman, 1957**] Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

[**Bertsekas, 1987**] Bertsekas, D. P., *Dynamic Programming. Deterministic and Stochastic Models*, Prentice Hall, Englewood Cliffs, N.J., 1987.

[**Brooke et al, 1988**] Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User's Guide*, The Scientific Press, 1988.

[**Davis, 1987**] Davis, L., (editor), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987.

[**De Jong, 1985**] De Jong, K.A., *Genetic Algorithms: A 10 Year Perspective*, Proceedings of an International Conference on genetic Algorithms and Their Applications, Pittsburgh, pp.169–177.

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

[**Grefenstette, 1986**] Grefenstette, J.J. *Optimization of Control Parameters for Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No.1, January/February 1986, pp.122–128.

[**Holland, 1975**] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[**Holland, 1986**] Holland, J., *Escaping Brittleness*, in *Machine Learning II*, ed. R. Michalski, J. Carbonell, T. Mitchel, Morgan Kaufmann Publ., Los Altos, CA.

[**Holland et al., 1986**] Holland, J.H., Holyoak, K.J., Nisbett, R.E., and Thagard, P.R., *Induction*, The MIT Press, 1986,

[**Michalewicz et al., 1989**] Michalewicz, Z., Vignaux, G.A., Groves, L. *Genetic Algorithms for Approximation and Optimization Problems*, Proceedings of the 11th New Zealand Computer Conference, Wellington, August 16–18, 1989, pp.211-223.

[**Michalewicz & Schell, J., 1990**] Michalewicz, Z., and schell, J., *Data Structures + Genetic Operators = Evolution Programs*, submitted for the International Conference on Tools for AI, Washington, November 6–9, 1990.

[**Michalewicz et al., 1990**] Michalewicz, Z., Vignaux, G.A., Hobbs, M., *A Genetic Algorithm for the Nonlinear Transportation Problem*, submitted to the ORSA Journal on Computing.

[**Vignaux & Michalewicz, 1989a**] Vignaux, G.A., Michalewicz, Z., *Genetic Algorithms for the Transportation Problem*, Proc. 4th International Symposium on Methodologies for Intelligent Systems, Charlotte, October 12–14, 1989, pp.252–259.

[**Vignaux & Michalewicz, 1989b**] Vignaux, G.A., Michalewicz, Z., *A Genetic Algorithm for the Linear Transportation Problem*, submitted to IEEE Transactions on Systems, Man, and Cybernetics.