Global Analysis and Discrete Mathematics
Volume 2, Issue 2, pp. 103–114
ISSN: 2476-5341

A Method for Solving Optimal Control Problems Using Genetic Programming

Ali Akbar Bani* · **Maliheh Darbani**

Received: 28 November 2016 / Accepted: 1 June 2017

Abstract This paper deals with a novel method for solving optimal control problems based on genetic programming. This approach produces some trial solutions and seeks the best of them. If the solution cannot be expressed in a closed analytical form then our method produces an approximation with a controlled level of accuracy. Using numerical examples, we will demonstrate how to use the results.

Keywords Genetic programming · Optimal control problems · Grammatical evolution

Mathematics Subject Classification (2010) 49J15

1 Introduction

Solving the issues regarding optimal control is of special complexities. Thus, offering an appropriate and efficient method to solve these issues is of great significance. In this regard, many methods have been suggested so for. However, we offer a new method for solving OCP, which is based on genetic programming.
The GP system is a series of successive operations which choose, the best from

*Corresponding author
Ali Akbar Bani
Department of Mathematics, Gomishan Center, Gorgan Branch, Islamic Azad University, Gomishan, Iran.
E-mail: bani_ali@yahoo.com

Maliheh Darbani
Department of Mathematics, Gomishan Center, Gorgan Branch, Islamic Azad University, Gomishan, Iran.

the candidate 's numerous answers. To create a variety of answers, such genetic agents as crossover or mutation are used. In this method, the proposed answers of close form are not predetermined, but they are made through a dynamic process which are sometimes very complex.
We use the Grammatical evolution for production because it can generate the programs in the desired language, then some genetic operators such as crossover and mutation are applied to new products. The production process is implemented by a grammar expressed in Backus Naur Form. This paper is organized as follows: in section 2 we give a brief presentation of grammatical evolution, in section 3 we describe in detail this algorithm, in sections 4 and 5 we present our examples and experimental results and in section 6 we present our conclusion.

2 Grammatical evolution

Grammatical evolution is an evolutionary automatic programming process that can generate programs in a desired language. The process needs the BNF grammar for production rule. It deals vectors of integers that called chromosomes. Each integer in chromosome indicates a production rule from BNF grammar. GE starts from the first symbol and the program string is then gradually generated by the placement of non-terminal symbols with the right hand of the selected production rule. Algorithm has the following steps:

get an element from the chromosome (with value R)
select the rule according to the scheme

$$Rule = R \ mod \ SC$$

Where SC is the number of rules for the specific non-terminal symbol.The process of replacing non-terminal symbols with the right hand of production rules is continued until either a full program has been generated or the end of chromosome has been reached. In the latter case we can reject the entire chromosome or we can start over (wrapping event) from the first element of the chromosome. In our approach we allow at most two wrapping events to occur.

In Fig.1 is given a small piece of the C programming language grammar. The numbers in parentheses denote the sequence number of the corresponding production rule to be used in the selection procedure described above.

The symbol S in the grammar denotes the start symbol of the grammar. For example, suppose we have the chromosome $x = [2, 6, 0, 4, 8, 9, 16, 1, 3, 1]$. In Table. 1 we show how a valid function is produced from t. The resulting function in the above example is $f(t) = 2t^2 + t$. Further details about grammatical evolution can be found in ([7],[8],[9],[10]).

Fig.1 The grammar of the proposed method

s ::=< expr > (0)
expr ::=< expr >< op >< expr > (0)
|(< expr >) (1)
|func(< expr >) (2)
| < digit > (3)
|t (4)
|u (5)
|x (6)
< op >::= + (0)
| − (1)
| ∗ (2)
|/ (3)
< func >::= t+ (0)
|t- (1)
|t* (2)
|t/ (3)
< digit >::= 0 (0)
|1 (1)
|2 (2)
|3 (3)
|4 (4)
|5 (5)
|6 (6)
|7 (7)
|8 (8)
|9 (9)

3 Description of the algorithm

We have two algorithms. One of them search for finding analytical solution and another search for detecting numerical solution. Note proposed algorithm is established for polynomials. Solve a given optimal control problem has the following phases:

1. Initialization.
2. Fitness evaluation.

Table 1 Example of program construction.

String	Choromosome	Operation
<expr>	2,6,0,4,8,9,16,1,3,1	2 mod 7=2
fun(<expr>)	6,0,4,8,9,16,1,3,1	6 mod 4=2
t*(<expr>)	0,4,8,9,16,1,3,1	0 mod 7=0
t*(<expr>,<op>,<expr>)	4,8,9,16,1,3,1	4 mod 7=4
t*(t,<op>,<expr>)	8,9,16,1,3,1	8 mod 4=2
t*(t+<expr>)	9,16,1,3,1	9 mod 7=2
t*(t+<func>(<expr>))	16,1,3,1	16 mod 4=0
t*(t+t+(<expr>))	1,3,1	1 mod 7=1
t*(t+t+(<expr>))	3,1	3 mod 7=3
t*(t+t+(<digit>))	1	1 mod 10=1
t*(t+t+1)	1	1 mod 10=1

3. Genetic operations.
4. Termination control.

3.1 Initialization

At this stage, some values are given to crossover rate and mutation rate. Based on the values, a fraction of chromosome are subject to changes specific to mutation and crossover and then go to the next generation.

3.2 Fitness evaluation

Our optimal control problem (OCP) is expressed in the following form minimize

$$I(x(t), u(t)) = \int_0^{t_f} f_0(t, x(t), u(t)) \tag{1}$$

subject to the first-order dynamic constraints

$$\dot{x} = g(t, x(t), u(t)) \tag{2}$$

$$x(0) = x_0 \qquad , \qquad x(t_f) = x_f \tag{3}$$

where x(t) is the trajectory, u(t) is the control, t is the independent variable, t_f is the terminal time.
As noted we provide two algorithms for solving this problem. The first algorithm searches an analytical solution and the second seeks a numerical solution for OCP.

3.2.1 First algorithm

Our main goal is to find a control function $u(.)$ and a state function $x(.)$ so that the conditions of (2)-(3) are met and (1) is minimised. The steps for the fitness evaluation of the chromosome i are the following:

a) Construct the models $\hat{u}_i(t)$ and $\hat{x}_i(t)$ corresponding whatever is expressed in last section, so that the differential equation is true

$$\hat{\dot{x}}_i(t) = g(t, \hat{x}_i(t), \hat{u}_i(t))$$

b) Calculate the quantity

$$E_i = \lambda((\hat{x}_i(t_f) - x_f)^2 + (\hat{x}_i(t_0) - x_0)^2)$$

where λ is a positive number.

c) Calculate the quantity

$$I_i = \int_0^{t_f} (f_0(t, \hat{x}_i(t), \hat{u}_i(t)))^2$$

d) Finally, the fitness of the chromosome i is given by:

$$F_i = E_i + I_i$$

3.2.2 second algorithm

In this case as before we find a control function $u(.)$ such that the corresponding state $x(.)$ satisfying (2)-(3) and minimize (1). To do this first, we split the interval $[0, t_f]$ to N subinterval $[t_{i-1}, t_i]$, $i = 1, 2, \ldots, N$. Since we mind to find an approximate optimal control for problem (1)-(2), we focus on finding an optimal control function $u(.)$ for the problem by genetic programming. If $u(t)$ be a control function then by a numerical method as Euler method or Rung-kutta, we can discover trajectory corresponding $u(t)$ from (2) with initial condition $x(0) = x_0$. The steps for the fitness evaluation of the chromosome i are the following:

a) Construct the model $\hat{u}_i(t)$ as before.

b)Solve the differential equation

$$\hat{\dot{x}}_i(t) = g(t, \hat{x}_i(t), \hat{u}_i(t))$$

by a numerical method as Euler method or Rung-kutta.

c) Calculate the quantity

$$E_i = \lambda((\hat{x}_i(t_f) - x_f)^2 + (\hat{x}_i(t_0) - x_0)^2)$$

where λ is a positive number.

d) Calculate the quantity

$$I_i = \int_0^{t_f} (f_0(t, \hat{x}_i(t), \hat{u}_i(t)))^2$$

by a numerical method as Trapezoidal method or Simpson.

e) Finally, the fitness of the chromosome i is given by:

$$F_i = E_i + I_i$$

We will run the referred process on each of the two algorithms to all individuals in the population and we arrange them in descending order according to their fitness value. Accordingly, we apply the genetic operators on people and the new generation create. This procedure repeat until the termination criteria occur.

3.3 Genetic operations

The genetic operations used are crossover and the mutation. The crossover allow new individuals to be created.The individuals participating in the crossover operation are selected proportionate to fitness and via tournament selection i.e. the individual with the best fitness in the group is selected the others are disearded.

The crossover operation produces two offspring .The two offspring is usually different from their two parents and different from each other. In that opration for each couple of new chromosomes two parents are selected ,we cut these parent-chromosomes at a randomly chosen point and we exchange the right-hand-side sub-chromosome [6].

The next genetic operator used is the mutation. Mutation is used very sparingly in work. The mutation operation in an asexual operation in that it operates on only one individual. It begins by randomly selecting a string from population and then randomly selecting a number between 1 and L (length of string) as the mutation point. Then the single character at the selected mutation point is changed.

3.4 Termination control

The genetic operators are used to produce new generations until either a chromosome is found in the population with the best fitness or the maximum number of frequency happens for producing new generation.

4 Examples

We define an error function as $e(t_f) = \hat{x}(t_f) - x_f$ on $[0, t_f]$, where $\hat{x}(t_f)$ and x_f are the exact and the approximate solution obtained from each repetition GP for the example, respectively. In recent two examples we apply the first algorithm to obtain approximate solutions of some OCP.

4.1 Example

Consider the following optimal control problem [2] which is minimization of the functional

$$I(x,u) = \int_0^1 (x(t)^2 + u(t)^2)dt$$

$$\text{subject to :} \qquad \dot{x} = u(t)$$

$$x(0) = 1 \qquad , \qquad x(1) = 0.6481$$

After solving with proposed method, we obtain the following results:

$$\hat{u}(t) = -\frac{1055}{1000}t^2 + 2t - 1 \qquad , \qquad \hat{x}(t) = -\frac{1055}{3000}t^3 + t^2 - t + 1$$

The final value $x(1) = 0.6483$, the optimal value $I^* = 0.7695$ and the error function $e(1) = 2.0000 \times 10^{-4}$. The exact objective value is $I(x,u) = 0.7616$. The obtained approximate and exact optimal control and trajectory have been shown in Fig 2.

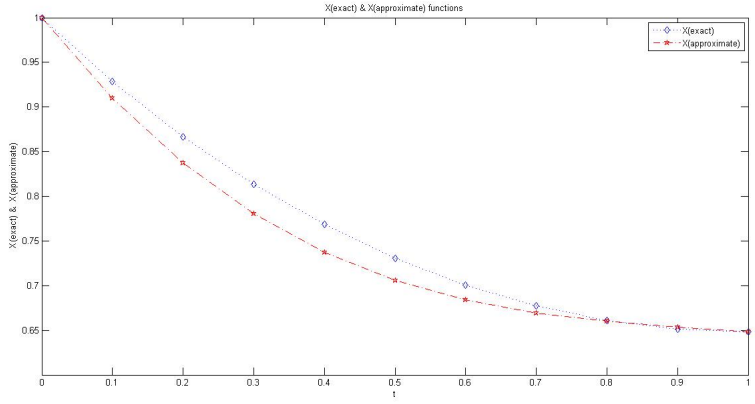

Figure 2. Comparison between the exact and approximate results of X(t).

4.2 Example

In this example a system of optimal control problem is considered [5] as follows:

$$I(x,u) = \int_0^1 u(t)^2 dt$$

$$\text{subject to :} \qquad \dot{x} = y(t),$$

$$\dot{y} = u(t).$$

Initial and final conditions are:

$$x(0) = 0 \quad , \quad y(0) = 0$$

$$x(1) = 0.1 \quad , \quad y(1) = 0.3$$

After solving with proposed method, we obtain the following results:

$$\hat{u}(t) = t - \frac{6}{10}t^2 \quad , \quad \hat{y}(t) = \frac{1}{2}t^2 - \frac{6}{30}t^3 \quad , \quad \hat{x}(t) = \frac{1}{6}t^3 - \frac{6}{120}t^4$$

The final amounts and errors will be as follows :

$$\hat{y}(1) = 0.3000 \quad , \quad e_y(1) = 0.000$$

$$\hat{x}(1) = 0.1167 \quad , \quad e_x(1) = 8.3 \times 10^{-2}$$

The obtained objective value is $I^* = 0.1053$. The obtained approximate and exact optimal control and trajectory have been shown in Fig 3.

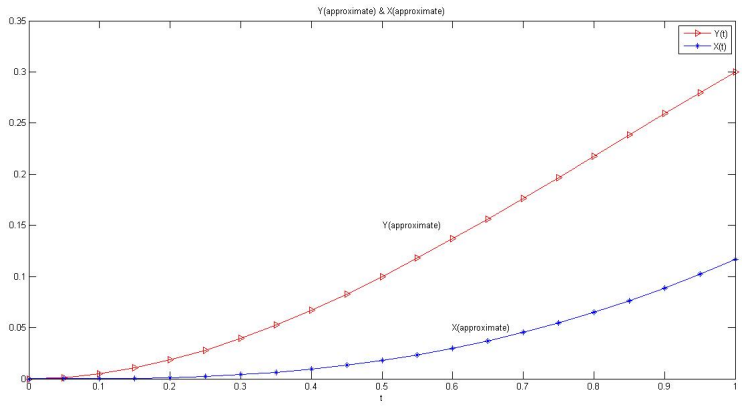

Figure 3. The approximate Y(t) and X(t) functions.

In following two examples we apply the second algorithm to obtain approximate solutions of some OCP.

4.3 Example

Consider the following optimal control problem [4] which is minimization of the functional

$$I(x,u) = \int_0^1 u(t)^2 dt$$
$$\text{subject to :} \qquad \dot{x} = x^2(t) + u(t)$$
$$x(0) = 0 \qquad , \qquad x(1) = 0.5$$

After solving with second method, we obtain the following results:

$$\hat{u}(t) = \frac{3116}{1000}t^4 - t^2 - t \qquad , \qquad \hat{x}(1) = 0.5007 \qquad , \qquad e(1) = 7.000 \times 10^{-4}$$

The obtained objective value is $I^* = 0.1053$. The obtained approximate and exact optimal control and trajectory have been shown in Fig 4.

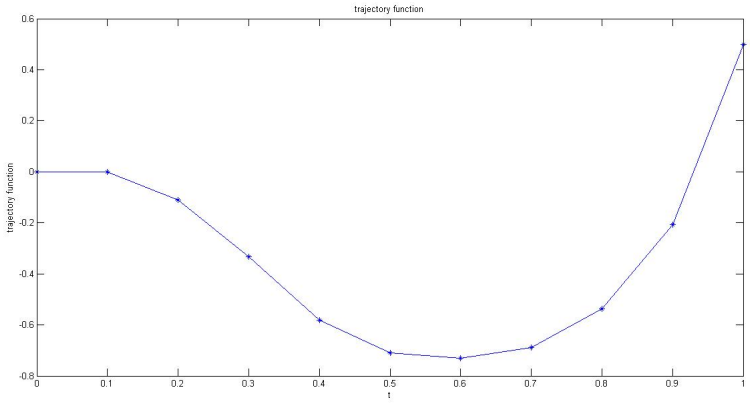

Figure 4. The approximate trajectory function.

4.4 Example

Consider the following OCP which is minimization of the functional

$$I(x,u) = \int_0^1 u(t)^2 dt$$
$$\text{subject to :} \qquad \dot{x} = \frac{1}{2}x^2(t)sin(x(t)) + u(t)$$
$$x(0) = 0 \qquad , \qquad x(1) = 0.5$$

After solving with second method, we obtain the following results:

$$\hat{u}(t) = -\frac{4003}{3002.532}t^2 \qquad , \qquad \hat{x}(1) = 0.5000 \qquad , \qquad e(1) = 0.000$$

The obtained objective value is $I^* = 0.3555$. The obtained approximate and exact optimal control and trajectory have been shown in Fig 5.

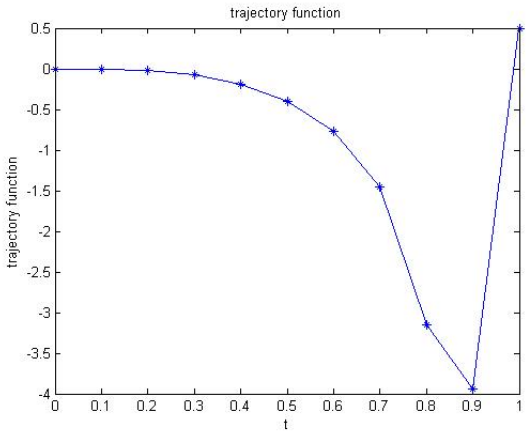

Figure 5. The approximate trajectory function.

5 Experimental results

For the first algorithm, each chromosome is split uniformly in 2 parts. Each part of the chromosome represents the solution of the corresponding for control function and trajectory function. We applied 80% for the crossover rate and 5% for mutation rate. We investigated the matter of these two parameters by performance some experiments. Each experiment was run 25 times. The population size was set to 1000 and the length of each chromosome to 40 . The size of the population is a serious parameter. A size of subminiature weakens the method's effectiveness. A whacking size presents the method slowly. So the choice of population size is a critical task. We have done a lot of testing and found that values in the interval $[500, 2000]$ are proper. The length of the chromosomes usually depends on the problem to be solved. In these experiments the maximum number of generations allowed was set to 500 and the preset fitness target for the termination criteria was 10^{-6}. In table 2 we list the results of the proposed method for the above examples. Under deadings MIN, MAX, AVG we list the minimum, maximum and average of generations in the set of 25 experiments.

Table 2. Results of Method for Examples.

OCP	MIN	MAX	AVG
$Exam4.1$	28	290	150
$Exam4.2$	25	300	175
$Exam4.3$	32	260	148
$Exam4.4$	15	275	136

In Fig. 6 we plot the proposed solutions of the example 1.

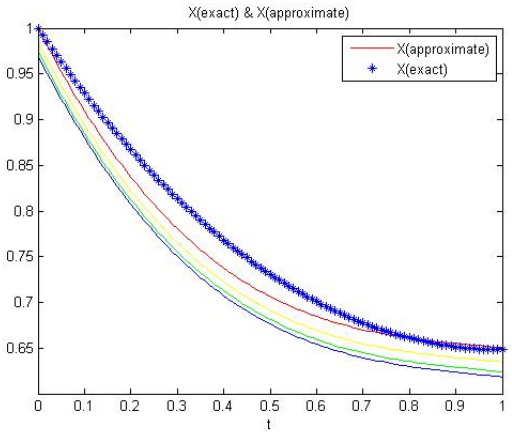

Figure 6. Candidate solutions of example 1.

6 Conclusion

We expressed a novel scheme based on genetic programming for solving OCP. Producing analytical solutions are the approximate or even the exact solutions is the advantage of the proposed method. Furthermore, if the exact solution could not be represented in a closed analytical form, the proposed method produced an approximation with a controlled level of accuracy.

References

1. A. Brabazon, M.O. Neill, A grammar model for foreign-exchange trading, in Proceedings of the International conference on Artificial Intelligence, volume II, H. R. Arabnia et al. (eds.), CSREA Press, 492–498, 23–26 June (2003).
2. DN. Burghes, A. Graham, Introduction to Control Theory, Including Optimal Control, E. Horwood, Halsted Press, New Yourk (1980).
3. J.J. Collins, C. Ryan, Automatic generation of robot behaviors using grammatical evolution, in Proc. of AROB 2000, the Fifth International Symposium on Artificial Life and Robotics (2000).
4. O.S. Fard and A.H. Borzabadi, Optimal control problem, quasi-assignment problem and genetic algorithm, Proc. World. Acad. Sci. Eng. Tech., 21, 70–73 (2007).
5. M. Keyanpour, M. Azizsefat, Numerical solution of optimal control problems by an iterative scheme, Advanced Modeling and Optimization, 13(1), 25–37 (2011).
6. J.R. Koza, Genetic Programming: On the Programming of Computer by Means of Natural Selection, MIT Press, Cambridge, MA (1992).
7. M.O. Neill, Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution, PhD Thesis, University Of Limerick, Ireland, August (2001).
8. M.O. Neill, C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, Volume 4 of Genetic programming, Kluwer Academic Publishers (2003).
9. M.O. Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evolutionary Computation, 5, 349–358 (2001).

10. C. Ryan, J.J. Collins, and M.O. Neill, Evolving programs for an arbitrary language, in Proceedings of the First European Workshop on Genetic Programming, volume 1391 of LNCS, W. Banzhaf, Ri. Poli, M. Schoenauer and T.C. Fogarty, (eds.), Springer-Verlag, 83–95, Paris, 14–15 April (1998).
11. M.O. Neill, J.J. Collins, and C. Ryan, Automatic generation of caching algorithms, in Evolutionary Algorithms in Engineering and Computer Science, Kaisa Miettinen, Marko M. Mkel, Pekka Neittaanmki, and Jacques Periaux (eds.), 127–134, Jyvskyl, Finland, 30 May-3 June (1999).
12. C. Ryan, M.O. Neill, and J.J. Collins, Grammatical evolution: Solving trigonometric identities, in Proceedings of Mendel 1998: 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets., Technical University of Brno, Faculty of Mechanical Engineering, 111–119, Brno, Czech Republic, June 24–26 (1998).
13. I.G. Tsoulos, I.E. Lagaris, Solving differential equations with genetic programming, Genetic Programming and Evolvable Machines, 7(1), 33–54 (2006).