A Tool for Solving Differential Games with Co-evolutionary Algorithms

Francisco Gordillo
Escuela Superior de Ingenieros
Camino Descubrimientos s/n
41092 Sevilla, Spain

Ismael Alcalá
Escuela Superior de Ingenieros
Camino Descubrimientos s/n
41092 Sevilla, Spain

Javier Aracil
Escuela Superior de Ingenieros
Camino Descubrimientos s/n
41092 Sevilla, Spain

Abstract

Game theory is concerned with optimization problems involving several players with conflicting interests. Differential games are an interesting area inside this field in which the problem is formulated by means of dynamical systems. In spite that the theoretical solution of differential games is well known, faced with large-scale, complex systems, analytical or, even, numerical methods are not usually suitable. Genetic algorithms appear to be useful for solving such problems when dealing with nonlinear, large-scale systems. This paper shows how co-evolutionary algorithms can be applied to solve differential games and presents a computer tool to formulate and solve these problems. The differential equations which define the system under study are written in Vensim, a visual modeling tool. The usefulness of the presented tool is shown by means of two examples.

1 INTRODUCTION

This paper deals with the use of genetic algorithms for the resolution of differential games. Differential game theory is an extension of traditional game theory which considers problems involving several players with conflicting interests [Isaacs, 1965]. Differential games consider dynamical systems as the arena of the game. Differential games have application in many diverse fields, such as missile guidance-avoidance [Isaacs, 1965], arm-race problems [Moriarty, 1984] and competitive environments. Another important application is for worst-case designs. In this case, player 1 is the designer while player 2 is not a real player but is nature: it is desired the action u_1 to be optimal faced against the worst possible set of disturbances. In this way, it is possible to replace a stochastic problem by a deterministic one. Differential games have also application in control theory [Vincent and Grantham, 1997].

This paper presents how genetic algorithms in its co-evolutionary variant can be used to solve this kind of problems. Firstly, a computer tool for solving differential games is presented. With this tool the problem can be formulated with Vensim, a visual modeling tool which allows to build and simulate dynamic systems [Ventana Systems, Inc, 1997]. It is a popular software package used in the discipline known as *System Dynamics*˜[Aracil and Gordillo, 1997, Roberts et al., 1983]. Secondly, the usefulness of the tool is shown by means of two simple examples. A more complex application of the tool is described in [Gordillo et al.,].

The paper is organized as follows. Section 2 shows the formulation of differential games. In Sect. 3 co-evolutionary genetic algorithms are presented. The implementation of them in the computer tool is shown in Sect. 4. Section 5 presents the examples of application. The paper closes with a section of conclusions.

2 DIFFERENTIAL GAMES

A differential game with two players can be formulated as follows. Given a dynamical system

$$\dot{x} = f(x, u_1(t), u_2(t), t), \qquad x(t_0) = x_0 \qquad (1)$$

where $x \in \mathbb{R}^n$, $u_1, u_2 \in \mathbb{R}$ and $\dot{x}$ means derivative of x with respect time. The signals $u_1(t)$ and $u_2(t)$ can be considered as the control signals (the actions) which two players can perform to modify the system behavior.

Consider also two performance criteria

$$J_1 \quad = \quad S_1(x(t_f), t_f) + \int_{t_0}^{t_f} L_1(x, u, v, t)dt \qquad (2)$$

$$J_2 = S_2(x(t_f), t_f) + \int_{t_0}^{t_f} L_2(x, u, v, t) dt \quad (3)$$

The objective of player 1 is to find $u_1(t)$ such that J_1 is minimum. Likewise, player 2 must select $u_2(t)$ such that J_2 is minimum.

In fact, other kind of criteria can be considered but this formulation is the usual for analytically solvable problems. Likewise, differential games with more than two players can also be formulated.

The solution of the problem is known in some cases. In particular, when the problem is zero-sum, that is, $J_1 = -J_2$ the ordinary differential equations which define the solution are well known [Isaacs, 1965]. Nevertheless, the class of problems considered in this paper need not to fulfill this condition. Furthermore, when the system equation is other than some very simple examples, the resolution of the differential equations is not a trivial task.

3 CO-EVOLUTIONARY ALGORITHMS

Genetic algorithms are general-purpose search methods based on natural selection, which can be used to solve optimization problems. They are especially useful when other traditional methods fail since genetic algorithms are not very demanding in properties such as continuity or differentiabilility.

Co-evolutionary algorithms are a variant of standard genetic algorithms in which two or more genetic populations evolve interacting each other [Hillis, 1992, Koza, 1992]. In other words, each population tries to optimize its own fitness function but this fitness function depends on the state of the rest of the populations. Co-evolutionary algorithms are usually classified into *competitive* and *cooperative* [de Moura Olivera and Jones, 1998] depending if the different objectives of the populations are opposed or complementary, respectively. We believe that this classification should not be so strict and intermediate situations can be allowed: the fitness functions of the different populations may pursue different objectives but the profit of one population need not to imply losses for the other players. This is the underlining idea of this paper which allows to deal with non-zero sum games by means of co-evolutionary algorithms.

The outline of these algorithms is described in the following. Each population evolves as in standard genetic algorithms with the usual operators: selection, crossover and mutation. Other operators are allowed as usual. The difference lies in the way the fitness measure of each individual is obtained. Since the fitness function depends on the state of the rest of the populations, this measure can not be computed without knowing the value of the individuals of the other populations. The procedure is the following. Consider the case of two populations. In order to evaluate the fitness of individual i of the first population it must encounter all the individuals of the other population[1]. As a consequence of the encounter of individual i of the first population with individual j of the second population, two basic fitness values f_{ij}^1 and f_{ij}^2 are obtained. This basic fitness values are not the fitness of the individuals since other encounters must also be considered. In this way two, matrices $F^1 = [f_{ij}^1]$ and $F^2 = [f_{ij}^2]$ are formed. Then, the fitness associated to individual i of the first population can be computed averaging the values f_{ij}^1 with j from 1 to the number of individuals in the second population. In the same way the fitness associated to individuals of the second population is computed averaging the rows of matrix F^2.

When more populations exist the procedure is similar but more matrices F^i will be formed and they will have more dimensions. In this way, in a game with three players, the element f_{ijk}^n of matrix F^n represents the fitness value for player n resultant of the encounter of individual i of the first population with individual j of the second population and individual k of the third population.

Co-evolutionary algorithms have been applied, among other fields, to manufacturing scheduling optimization [Husband and Mill, 1991], multivariable system identification [de Moura Olivera and Jones, 1998], controller design [Jones and de Moura Oliveira, 1997, Paredis, 1997] and competitive learning [Angeline and Pollack, 1993].

The application of this algorithm to the case of differential games is straightforward. The encounters of two (or more) individuals correspond to a simulation of Eq. (1) with the control signal represented by these individuals. The basic fitness values corresponding to the encounter are the values of J_i (see Eqs. (2) and (3)) resultant of the simulation. The rest of the algorithm remains unchanged.

[1] In some variants of the algorithm the encounters are only with some randomly-selected individuals of the other population.

4 CO-EVOLUTIONARY ALGORITHMS IN VENSIM

In this section, the computer tool for solving differential games is described. A co-evolutionary algorithm has been implemented in conjunction with Vensim to solve differential games with two players (the necessary modifications to deal with more players are described below). With Vensim, the dynamic equations (1) can be easily programmed even with large systems with many nonlinearities. The performance criteria J_1 and J_2 can also be programmed with Vensim.

There exists a version of Vensim called Vensim DLL which is a Dynamic Link Library (DLL). This version allows the programmer to make use of the models developed in Vensim from other applications. This is the strategy used in the co-evolutionary tool. The main part of this tool is written in C and calls Vensim DLL to perform the simulations.

In the following, the way the algorithm has been implemented is explained. Two separated programs execute each one an almost-standard genetic algorithm. A different population evolves in each program.

The individuals of each population represent the control signals u_1 and u_2 respectively. As genetic algorithms work with strings of symbols, a codification must be considered. The most natural way is to discretize the variable t between t_0 and t_f and code the values of the signals in the considered instants as binary numbers. In this way, each individual represents a function of time $u_i(t)$. A more interesting approach, from the practical point of view, is to try to find the optimal laws in closed-loop form $u_i(x)$ instead of open loop control signals $u_i(t)$. The inherent feedback in closed-loop control laws allows to correct unpredictable circumstances such as disturbances, error in the initial conditions or modeling errors.

In any case, the tool allows to use any code for u_1 and u_2 since the genetic algorithm implemented in the programs is a general-purpose one. In the examples of the next section a closed-loop control law is found.

On the other hand, both programs are not completely independent since they are synchronized, that is, they wait each other to perform the evaluation of their populations. When both programs are ready to evaluate their populations, one of them (the client) sends to the other (the server) the information which defines the individuals of the present generation. With this information, the other program is ready to perform the encounters of the, say, n_1 individuals of one population with the, say, n_2 individuals of the other one. In this way $n_1 \times n_2$ encounters must be implemented in each generation. For each encounter, an order is passed to Vensim DLL to perform a simulation with the control signal associated to the corresponding individuals. At the end of each simulation two fitness indexes are obtained which represent the value of J_1 and J_2. In this way matrices F^1 and F^2 are formed and the fitness of each individual of both populations can be computed. Then, the fitness values of the individuals associated to the client are sent to it. Now, both programs can continue separately their algorithms till the next generation.

The communication between both programs is implemented with the TCP/IP protocol. In this way, they can be executed in the same or in different computers. In this last case, an easy improvement would allow to save time: each program would evaluate the half of matrices F^1 and F^2. With this enhancement, which has not been programmed yet, the program would acquire parallel computation capabilities. Another easy improvement of the tool would be the incorporation of more players which would be implemented with more genetic-algorithm programs.

The tool includes a graphical interface which allows the user to easily configure the differential game. The user can

- specify the name of the Vensim model which defines the dynamical system and the performance criterion. The last one can be much more general than the type given in (2) and (3).
- specify the configuration of each genetic algorithm:
  - Performance criterion:
    - Name of the Vensim variable.
    - If it is desired to optimize the final value of the variable or its integral in the simulation time.
    - If it is desired to maximize or minimize.
  - Unknown control signal:
    - Name of the Vensim variable.
    - Maximum and minimum values bounds.
    - Number of values considered in the interval.
  - Parameters which define the genetic algorithm: number of individuals, maximum number of evaluations, crossover and mutation probabilities.
- Configure the output of the program.

5 EXAMPLES

In this section, two examples of application of the tool are exposed involving a pursuit game in two dimensions. These examples are included only in order to illustrate the use of the presented tool. More refinements must be included in these examples to acquire more practical value. A more complex example of application of the tool can be found in [Gordillo et al.,]. Another application of genetic algorithms (with a different approach) to a similar example can be found in [Sheppard and Salzberg, 1995].

In the first example, one of the players (the pursuer) tries to catch the other player (the evader). This is a typical example of differential game. In the second example, a new pursuer is added in such a way that the two pursuers act as an only player. The two-pursuer problem has no known analytical solution [Imado and Ishihara, 1993].

In both examples, the dynamics of each player are:

$$\begin{aligned} \frac{d^2x_1^i}{dt^2} &= u_1^i \\ \frac{d^2x_2^i}{dt^2} &= u_2^i \end{aligned}$$

where $\mathbf{r^i} = (x_1^i, x_2^i)$ are the coordinates of the player i ($i = P$ for the pursuer and $i = E$ for the evader) and $\mathbf{a^i} = (u_1^i, u_2^i)$ is its acceleration which acts as the control signal. Furthermore, two restrictions have been considered for each player:

$$\begin{aligned} \|\mathbf{a^i}\| &\leq a_{max}^i \\ \|\dot{\mathbf{x}}^\mathbf{i}\| &\leq v_{max}^i \end{aligned}$$

where a_{max}^i and v_{max}^i are predefined constants.

5.1 EXAMPLE 1: SINGLE-PURSUER GAME

In the case of one pursuer and one evader, the objective of the pursuer is to minimize

$$J_P = \int_0^{t_c} L_P dt = \int_0^{t_c} t\|\mathbf{r^i} - \mathbf{r^j}\| dt \qquad (4)$$

where t_c is the time in which the pursuer catches the evader. This index is similar to the ITAE criterion used in control theory. The multiplication by time in the definition of L penalizes in smaller amount early errors than late errors. In this way the pursuer attempts to catch the evader as soon as possible. The final time for the simulations t_f has been chosen equal to 30.

On the other hand, the evader tries to maximize the same functional, in other words, $J_E = -J_P$.

The parameters which define the restrictions on the movement of the players are:

$$\begin{aligned} a_{max}^P &= 1 & a_{max}^E &= 2 \\ v_{max}^P &= 2 & v_{max}^E &= 1 \end{aligned}$$

The objective of the differential game is to find optimal closed-loop control laws for both players. It is assumed that these control laws have the form:

$$\begin{aligned} u_1^i(t) = & \quad K_{r_{11}}^i(x_1^P(t) - x_1^E(t)) \\ & + K_{r_{12}}^i(x_2^P(t) - x_2^E(t)) \\ & + K_{v_{11}}^i(\dot{x}_1^P(t) - \dot{x}_1^E(t)) \\ & + K_{v_{12}}^i(\dot{x}_2^P(t) - \dot{x}_2^E(t)) \qquad (5) \\ u_2^i(t) = & \quad K_{r_{21}}^i(x_1^P(t) - x_1^E(t)) \\ & + K_{r_{22}}^i(x_2^P(t) - x_2^E(t)) \\ & + K_{v_{21}}^i(\dot{x}_1^P(t) - \dot{x}_1^E(t)) \\ & + K_{v_{22}}^i(\dot{x}_2^P(t) - \dot{x}_2^E(t)) \qquad (6) \end{aligned}$$

with $i = P, E$.

Thus, the control laws are defined with the value of the parameters

$$(K_{r_{11}}^P, K_{r_{12}}^P, K_{r_{21}}^P, K_{r_{22}}^P, K_{v_{11}}^P, K_{v_{12}}^P, K_{v_{21}}^P, K_{v_{22}}^P)$$

for the pursuer and

$$(K_{r_{11}}^E, K_{r_{12}}^E, K_{r_{21}}^E, K_{r_{22}}^E, K_{v_{11}}^E, K_{v_{12}}^E, K_{v_{21}}^E, K_{v_{22}}^E)$$

for the evader. Hence, each individual of both populations of the co-evolutionary algorithm represents eight real parameters.

This problem has been introduced in the tool described in the previous section. For each encounter, a simulation has been performed, corresponding to the following initial conditions:

$$\begin{aligned} \mathbf{r^P} &= (0,0) & \mathbf{v^P} &= (0,2) \\ \mathbf{r^E} &= (10,0) & \mathbf{v^E} &= (0,1) \end{aligned}$$

The results of one execution of the tool are shown in Tab. 1. In order to test the validity of the solution some simulations have been performed. The first one, shown in Fig. 1, corresponds to the best control laws and the same initial conditions of the simulation which defines the fitness of the individuals. In this figure the behavior of the pursuer is represented by stars and the one of the evader by circles. Figure 2 shows the evolution of J_P during this simulation. Figure 3 shows a simulation with different initial conditions:

$$\begin{aligned} \mathbf{r^P} &= (0,0) & \mathbf{v^P} &= (0,2) \\ \mathbf{r^E} &= (15,0) & \mathbf{v^E} &= (-1,0) \end{aligned}$$

Table 1: Parameters Obtained for Example 1

$K^P_{r_{11}}$	-1.56	$K^E_{r_{11}}$	-0.70
$K^P_{r_{12}}$	-1.60	$K^E_{r_{12}}$	-0.59
$K^P_{r_{21}}$	0.06	$K^E_{r_{21}}$	0.04
$K^P_{r_{22}}$	-0.27	$K^E_{r_{22}}$	-0.22
$K^P_{v_{11}}$	-1.64	$K^E_{v_{11}}$	-1.82
$K^P_{v_{12}}$	1.04	$K^E_{v_{12}}$	1.34
$K^P_{v_{21}}$	-0.02	$K^E_{v_{21}}$	1.30
$K^P_{v_{22}}$	-2.00	$K^E_{v_{22}}$	-2.00

The behavior of both players in both simulations is quite coherent. As it can be seen, the evader is able to escape. This is due to the fact that the constraints in the movement of the pursuer are very restrictive. Concretely, a^P_{max} is quite small compared with a^E_{max}. The simulation of Fig. 4 corresponds to the same conditions of Fig. 1 but with $a^P_{max} = 1.5$. In this case, the pursuer catches the evader.

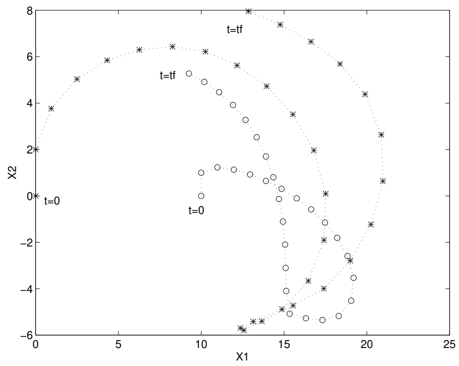

Figure 1: Example 1: Best Control Laws with Training Initial Conditions

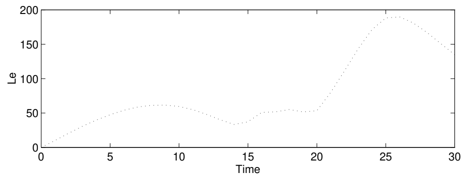

Figure 2: Evolution of L_P for the Case of Fig. 1

5.2 EXAMPLE 2: TWO-PURSUER GAME

In the second example two pursuers attempt to catch a single evader. The evader tries to remain far from both pursuers at the same time. This problem has no known analytical solution even when looking for open-loop control laws [Imado and Ishihara, 1993]. Here,

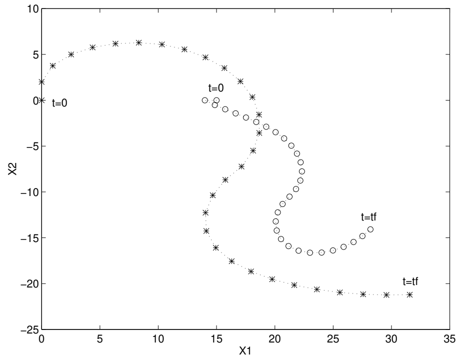

Figure 3: Example 1: Best Control Laws with Test Initial Conditions

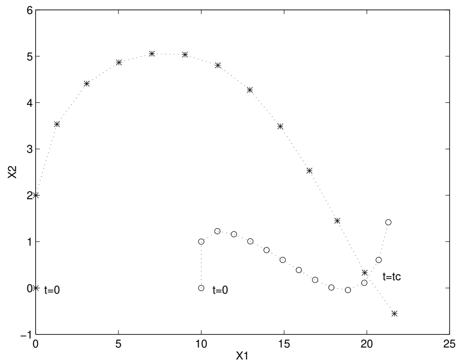

Figure 4: Example 1: Best Control Laws with Training Initial Conditions and $a^P_{max} = 1.5$

the control law, as well as the performance index, for both pursuers are identical, so they act as a single player. Hence, a population of the co-evolutionary algorithm represents both pursuers simultaneously.

The individuals representing the control law of both pursuers attempt to minimize the performance index

$$J_P = \int_0^{t_c} L_P dt = \int_0^{t_c} t \left(\|\mathbf{r^{P_1}} - \mathbf{r^E}\| + \|\mathbf{r^{P_2}} - \mathbf{r^E}\| \right) dt$$

where $\mathbf{r^{P_1}}$ and $\mathbf{r^{P_2}}$ are the positions of both pursuers. Their control law has the same form (5)–(6) and, hence, the same parameters.

On the other hand, the evader tries to maintain both pursuers as far as possible. This can be done maximizing the distance to the nearer pursuer. Thus, a sensible performance index is

$$J_E = \int_0^{t_c} L_E dt = -\int_0^{t_c} t \min_{i=1,2} \|\mathbf{r^E} - \mathbf{r^{P_i}}\| dt$$

The structure of the control law is:

$$\begin{aligned}
u_1^E(t) = \quad & K^E_{r_{11}}(x_1^{P_1}(t) - x_1^E(t)) \\
+ \quad & K^E_{r_{11}}(x_1^{P_2}(t) - x_1^E(t)) \\
+ \quad & K^E_{r_{12}}(x_2^{P_1}(t) - x_2^E(t)) \\
+ \quad & K^E_{r_{12}}(x_2^{P_2}(t) - x_2^E(t)) \\
+ \quad & K^E_{v_{11}}(\dot{x}_1^{P_1}(t) - \dot{x}_1^E(t)) \\
+ \quad & K^E_{v_{11}}(\dot{x}_1^{P_2}(t) - \dot{x}_1^E(t)) \\
+ \quad & K^E_{v_{12}}(\dot{x}_2^{P_1}(t) - \dot{x}_2^E(t)) \\
+ \quad & K^E_{v_{12}}(\dot{x}_2^{P_2}(t) - \dot{x}_2^E(t)) \\
u_2^E(t) = \quad & K^E_{r_{21}}(x_1^{P_1}(t) - x_1^E(t)) \\
+ \quad & K^E_{r_{21}}(x_1^{P_2}(t) - x_1^E(t)) \\
+ \quad & K^E_{r_{22}}(x_2^{P_1}(t) - x_2^E(t)) \\
+ \quad & K^E_{r_{22}}(x_2^{P_2}(t) - x_2^E(t)) \\
+ \quad & K^E_{v_{21}}(\dot{x}_1^{P_1}(t) - \dot{x}_1^E(t)) \\
+ \quad & K^E_{v_{21}}(\dot{x}_1^{P_2}(t) - \dot{x}_1^E(t)) \\
+ \quad & K^E_{v_{22}}(\dot{x}_2^{P_1}(t) - \dot{x}_2^E(t)) \\
+ \quad & K^E_{v_{22}}(\dot{x}_2^{P_2}(t) - \dot{x}_2^E(t))
\end{aligned}$$

Notice that the constants associated to both pursuers are the same. Therefore, each individual of the evader population still represents eight parameters.

The maximum velocities and accelerations are the same as in example 1.

As in the previous example, one simulation has been performed for each encounter, corresponding to the following initial conditions:

$$\begin{aligned}
\mathbf{r^{P_1}} &= (0,0) & \mathbf{v^{P_1}} &= (0,2) \\
\mathbf{r^{P_2}} &= (15,0) & \mathbf{v^{P_2}} &= (0,2) \\
\mathbf{r^{E}} &= (10,0) & \mathbf{v^{E}} &= (0,1)
\end{aligned}$$

The results of one execution of the tool is shown in Tab. 2. As in the previous example, some simulations

Table 2: Parameters Obtained for Example 2

$K^P_{r_{11}}$	−0.68	$K^E_{r_{11}}$	−2.00
$K^P_{r_{12}}$	−0.06	$K^E_{r_{12}}$	−1.77
$K^P_{r_{21}}$	0.31	$K^E_{r_{21}}$	1.88
$K^P_{r_{22}}$	−0.73	$K^E_{r_{22}}$	1.50
$K^P_{v_{11}}$	−1.49	$K^E_{v_{11}}$	−0.44
$K^P_{v_{12}}$	0.34	$K^E_{v_{12}}$	−0.17
$K^P_{v_{21}}$	0.07	$K^E_{v_{21}}$	1.40
$K^P_{v_{22}}$	−2.00	$K^E_{v_{22}}$	0.00

have been performed to test the validity of the solution. The first one, shown in Fig. 5 corresponds to the best control laws and the same initial conditions of the simulation which defines the fitness of the individuals. Figure 6 shows the evolution of J_P during this simulation. Figure 7 shows a simulation with different initial conditions. The behavior of the players in both simulations is still quite coherent. The simulation of Fig. 8 corresponds to the same conditions of Fig. 5 but with $v^E_{max} = 2$. The simulation of Fig. 9 confronts the best evader with a pursuer control law obtained when the evolution is aborted prematurely. It can be seen that the behavior of the evader is, now, much worse than the previous one. Finally, the obtained control law for the evader is confronted with the best pursuer control law of the previous example (with only one pursuer) in Fig. 10. It can be seen that the new control law for the pursuer is more suitable for the two-pursuer case.

6 CONCLUSIONS

A method based on co-evolutionary algorithms has been proposed to solve differential games. This method has been implemented in a computer tool which allows to define the problem with a modeling program. In this way the formulation of complex problems is allowed. The co-evolutionary algorithm is able to deal with these kind of problems as it has been verified by means of two examples.

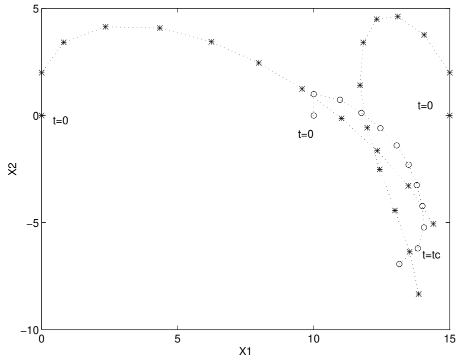

Figure 5: Example 2: Best Control Laws with Training Initial Conditions

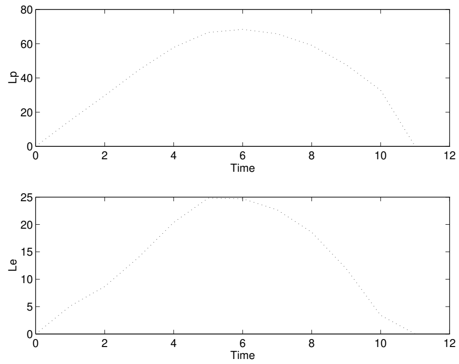

Figure 6: Evolution of L_P and L_E for the Case of Fig. 5

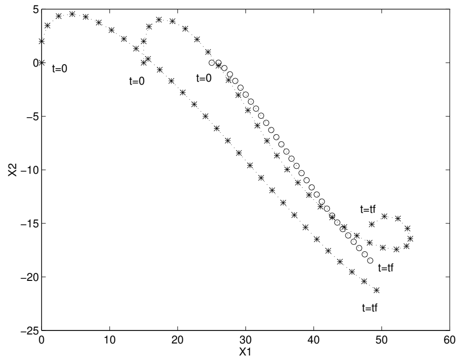

Figure 7: Example 2: Best Control Laws with Test Initial Conditions

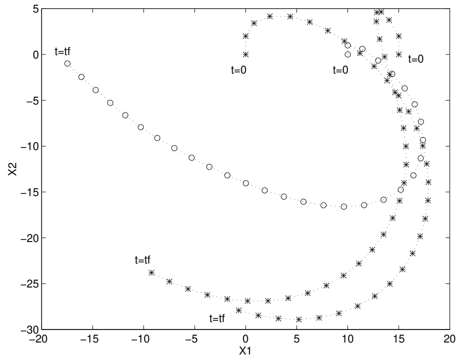

Figure 8: Example 2: Best Control Laws with Training Initial Conditions and $v^E_{max} = 2$

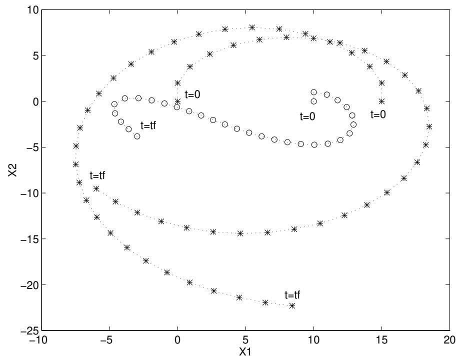

Figure 9: Example 2: Best Evader Against Non-optimal Pursuers

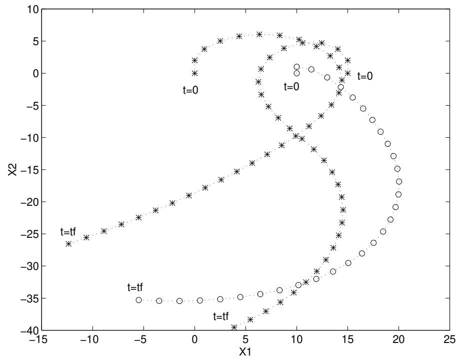

Figure 10: Example 2: Best Evader Against the Best, One-pursuer Pursuers

Acknowledgments

The authors want to thank Federico Cuesta who gave the idea of implementation of co-evolutionary algorithms.

This work has been supported by the Spanish Ministry of Education and Culture under grants CICYT TAP 97–0553 and CICYT TAP 98–0541 and by the "Departamento de Métodos Cuantitativos" of Telefónica, S.A.

References

[Angeline and Pollack, 1993] Angeline, P. and Pollack, J. (1993). Competitive environment evolve better solutions for complex tasks. In *Fifth International Conference on Genetic Algorithms*, pages 264–270.

[Aracil and Gordillo, 1997] Aracil, J. and Gordillo, F. (1997). *Dinámica de Sistemas*. Alianza Editorial. In Spanish.

[de Moura Olivera and Jones, 1998] de Moura Olivera, P. B. and Jones, A. H. (1998). Co-operative co-evolutionary multi-variable system identification using structured genetic algorithms. In *Application of Multi-variable System Techniques*. Professional Engineering Publishing.

[Gordillo et al.,] Gordillo, F., Vicente, S. M., Gala, M., Aracil, J., and Alcalá, I. Optimal behavior in competition between telecomunication companies: a co-evolutive, differential-game approach. Submitted to Journal of Computational Economics.

[Hillis, 1992] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In C. G. Langton, C. Taylor, J. D. F. and Rasmussen, S., editors, *Artificial Life II*, pages 313–323. Addison-Wesley.

[Husband and Mill, 1991] Husband, P. and Mill, F. (1991). Simulated co-evolution as the mechanism for emergent planning and scheduling. In Belew, R. and Booker, L., editors, *Proceedings of the Fourth International Conference on Genetic algorithms*, pages 264–270. Morgan Kaufmann.

[Imado and Ishihara, 1993] Imado, F. and Ishihara, T. (1993). Pursuit-evasion geometry analysis between two missiles and an aircraft. *Computers and Mathematics with Applications*, 26:3:125–139.

[Isaacs, 1965] Isaacs, R. (1965). *Differential Games*. Wiley.

[Jones and de Moura Oliveira, 1997] Jones, A. H. and de Moura Oliveira, P. (1997). Genetic design of robust PID controllers. In Smith, G. D., Steele, N. C., and Albrecht, R. F., editors, *Artificial Neural Nets and Genetic Algorithms*. Springer-Verlag.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press.

[Moriarty, 1984] Moriarty, G. (1984). Differential game theory applied to a model of the arms race. *IEEE Technology and Society Magazine*, pages 10–17.

[Paredis, 1997] Paredis, J. (1997). Coevolutionay process control. In Smith, G. D., Steele, N. C., and Albrecht, R. F., editors, *Artificial Neural Nets and Genetic Algorithms*. Springer-Verlag.

[Roberts et al., 1983] Roberts, N., Andersen, D. F., Deal, R., Grant, M. S., and Shaffer, W. (1983). *Introduction to Computer Simulation: A System Dynamics Modeling Approach*. Addison-Wesley.

[Sheppard and Salzberg, 1995] Sheppard, J. W. and Salzberg, S. L. (1995). Combining genetic algorithms with memory based reasoning in pursuit games. In *Proceedings of the 1995 International Conference on Genetic Algorithms*.

[Ventana Systems, Inc, 1997] Ventana Systems, Inc (1997). Vensim 3.0. Reference Manual.

[Vincent and Grantham, 1997] Vincent, T. and Grantham, W. (1997). *Nonlinear and Optimal Control Systems*. Wiley.