Realization of Gymnastic Movements on the Bar by Humanoid Robot Using a New Selfish Gene Algorithm

Lyes Tighzert and Boubekeur Mendil

Abstract This paper proposes a new selfish gene algorithm called the Replaces and Never Penalizes Selfish Gene Algorithm (RNPSGA). This new variant of selfish gene algorithm replaces the alleles of the less fit individual by the alleles of the fittest rather than penalizing them. The intensification of the search is then increased. The proposed algorithm is tested under some famous benchmark functions and compared to the standard selfish gene algorithm. We analyzes also the quality of convergence, the accuracy, the stability and the processing time of the proposed algorithm. We design by Solidwork a new virtual model of the humanoid robot hanging on the bar. The model is controlled using Simscape/Matlab. The proposed algorithm is then applied to the designed humanoid robot. The objective is to realize the gymnastic movements on the bar. An intelligent LQR controller is proposed to stabilize the swing-up of the robot.

Keywords Selfish gene · Computational intelligence · Humanoid · Robots · Gymnastic · Solidworks/Simmechanic · LQR · Optimal control

1 Introduction

The evolutionary algorithms are a stochastic search and optimization algorithms inspired from the Darwinian theory of evolution. This kind of computing started in the middle of twentieth century by means of the influence of several works [1, 2]. In the last decades this field have motivated many researchers and scientists who had contributed to its amazing progress. The evolutionary algorithms can be subdivided

L. Tighzert (✉) · B. Mendil
Laboratoire de Technologie Industrielle et de l'Information (LTII),
Faculté de Technologie, Université de Bejaia, 06000 Bejaia, Algeria
e-mail: ltighzert@gmail.com

B. Mendil
e-mail: bmendil@yahoo.fr

S. Chikhi et al. (eds.), *Modelling and Implementation of Complex Systems*, Lecture Notes in Networks and Systems 1,
DOI 10.1007/978-3-319-33410-3_4

into four subfields: evolution strategies [3], evolutionary programming [4], genetic programming [5] and genetic algorithms [6]. As genetic algorithms imitate the process of natural selection of the survival of the fittest, they use a population of individuals that generates offspring by means of crossover and mutation operators. In each generation the algorithm selects the fittest individuals to survive for the next generation. Several variants have been developed and proposed in the literature [7–9]. The basic differences between them is in their representation of chromosomes, their crossover operators, their mutation operators and their selection strategy. In evolutionary algorithms, we are principally interested by the intensification and the diversification of the search process [10]. The crossover operator allows the transfer of genetic information between individuals, hence it helps for the intensification of the search. The mutation adds new information into genotypes and helps the algorithm to explore different regions of search space and to avoid local optimums. The mutations diversify the search process and are indispensable to guaranty the convergence of the algorithm [11]. On 1989, the British biologist Richard Dawkins reformulates the theory of evolution in terms of genes rather than individuals [12]. We call his theory: the selfish gene theory. Dawkins suggests that evolution of spaces is based on genes. The body that have the biological spaces is created by genes to assure their duplication (their own survival).

En 1998 Corno et al. developed a new genetic algorithm based on the theory of Richard Dawkins. The algorithm is called the Selfish Gene Algorithm (SGA) [13]. This algorithm is focalized on genes and uses a pool of genes called the virtual population rather than a population of individuals. As this algorithm is central for our study we give a formal precise definition. Its pseudo code is shown in Fig. 1.

In SGA, the alleles compete for loci. It has each one a probability of selection that measures the chances that has an allele to be selected to form one of the individuals used in the tournament selection. The algorithm uses tournament selection to compare two individuals between them. The alleles of the winner of the tournament are rewarded by incrementing their selection probabilities. And the alleles of the failed individual are penalized by decreasing their selection probabilities. If one of the individual selected is fitter than the best ever found, it replaces it.

In this paper we propose a novel selfish gene algorithm called the Replaces and Never Penalizes Selfish Gene Algorithm (RNP-SGA). This algorithm does not penalize the alleles. The idea behind this variant is to replace the alleles of the fit less individual by the alleles of the fittest (the winner of the tournament). The rest of this paper is organized as fallow. In the next section we introduce the proposed algorithm. In section three we analyze and compare the RNP-SGA and the SGA in terms of quality of convergence, accuracy, stability and the time consumption. The fourth section is devoted to studying, modeling and controlling of a gymnastic humanoid robot on the bar. We use the proposed algorithm to optimize an LQR controller designed to assure the movement of swing-up on the bar. Section five conclude this paper.

Algorithm1: Selfish Gene Algorithm

```
1. VP = uniform initialization of the virtual population
2. P = initialize all probabilities p_ij to l/n_i
3. B = select (VP)  / * assume the best */
4. Repeat
5. / * tournament  */
6. G1 = select-individual ( VP,P) ;
7. G2 = select-individual ( VP,P) ;
8. if (fitness(G1) > fitness(G2)) then
9.     reward-alleles (G1) ;
10.    penalize-alleles (G2) ;
11. / * update best */
12.   if (fitness(G1) > fitness(B))  then
a.      B = G1 ;
13.   End if
14. Else
15.    reward-alleles (G2) ;
16.    penalize-alleles (G1) ;
17. / * update best */
18.    if (fitness(G2 > fitness(B)) then
19.      B = G2 ;
20.    End if
21. End if
22. until termination condition
23. return B
```

Fig. 1 The pseudo code of SGA [13]

2 The Replaces and Never Penalize Selfish Gene Algorithm

In this section we introduce the algorithm called the Replace and Never Penalize Selfish Gene Algorithm (RNP-SGA). This algorithm is a variant of selfish gene algorithm. The RNPSGA uses also the tournament selection but it does not penalize alleles. It rewards the alleles of the winner of the tournament and replaces the alleles of the failed individual by the alleles of the winner. This action of replacing allows a duplication of the genes of the winner in the virtual population. Hence, the intensification of the search process is increased. Actually, the proposed algorithm by rewarding the "good" alleles through increasing their selection probabilities will automatically penalize the "bad" ones; their selection probabilities will be relatively less significant. To assure a good exploration of the search space we increment the mutation probability. The mutation in RNP-SGA occurs randomly and can affect

Algorithm 2: Replaces and never penalizes Selfish Gene Algorithm

```
1.  VP = uniform initialization of the virtual population
2.  P = initialize all probabilities p_ij to l/n_i
3.  B = select (VP)  / * assume the best  */
4.  Repeat
5.  / * tournament  */
6.  G1 = select-individual ( VP, P) ;
7.  G2 = select-individual ( VP, P) ;
8.    if (fitness(G1) > fitness(G2)) then
9.        reward-alleles (G1) ;
10. / * replace alleles of G2 by those of G1 in VP  */
11.     VP(G2)= VP(G1) ;
12. / * update best */
13.     if (fitness(G1) > fitness(B))  then
14.       B = G1 ;
15.     End if
16. Else
17.     reward-alleles (G2) ;
18. / * replace alleles of G1 by those of G2 in VP  */
19.     VP(G1)= VP(G2) ;
20. / * update best */
21.     if (fitness(G2 > fitness(B)) then
22.        B= G2 ;
23.     End if
24. End if
25. / * mutation */
26. VP=mutate(VP)
27. until termination condition
28. return B
```

Fig. 2 The pseudo code of the proposed RNP-SGA

any gene in the virtual population. The pseudo code of the proposed algorithm is shown in Fig. 2.

The proposed algorithm uses a virtual population of genes. Each of them has a probability of selection that measures the chances of the given allele to be selected into the chromosomes of the individuals selected to compete in the tournament. The alleles of the winner (the fitter) are duplicated in the virtual population by over-writing (replacing) those of the failed. This action of overwriting ameliorates the intensification of the search. To avoid duplication of the same genes in the entire virtual population long before finding the optimal solution we increment the probability of mutation.

Generally, in evolutionary computing we are aimed to give a dynamic change to the parameters governing the algorithms [11]. So we propose the decreasing of the probability and step size of mutations to assure good accuracy and quality of convergence. Now we propose benchmarking of the proposed algorithm. The results of the simulation tests are shown in next section.

3 Benchmarking of the RNPSGA

3.1 Mathematical Functions

We propose in this section the benchmarking of the proposed algorithm under some famous benchmark functions. We choose from literature [8, 9] a set of seven functions. The Table 1 gives the mathematical form of the used benchmark functions. The value of the optimal of all those functions is zero and its position is on the origin of the D dimensions space.

Table 1 Mathematical form of the test functions

Name and type	Mathematical form	Interval
Uni modal functions		
F1: Sphere function	$\sum_{i=1}^{D} x_i^2$	[−100,100]
F2: Schwefel's problem	$\sum_{i=1}^{D} \lvert x_i \rvert^2 + \prod_{i=1}^{D} \lvert x_i \rvert^2$	[−100, 100]
F3: Generalized Rosenbrock function	$\sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−5.12, 5.12]
Multimodal functions		
F4: Generalized Rastrigin function	$\sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[−5.12, 5.12]
F5: Generalized Griewank function	$\frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[−600, 600]
F6: Ackley function	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + e + 1$	[−32, 32]
Composite functions		
F7: Composite function	Composite function CF2 in [14]	[−5, 5]

3.2 Simulation Settings

In order to compare the SGA and the proposed algorithm we give the parameters used in our experiments:

- The maximum number of iteration N = 40000
- Dimension: three case D = 10, D = 30 and D = 50
- Dimension of VP is set to 50 × D (VP is a matrix)
- Normal distribution mutation: $x = x + \sigma N(0, 1)$
- Probability of mutation μ = 0.01 for SGA and μ = 0.1 for RNPSGA
- $segma = 0.5(Max - Min)$
- $segma\_dump\_initial = 1;$
- $segma\_dump\_final = 0.992;$
- $$segma_{dump} = \left(\frac{N-K}{N-1}\right)^2 \times (segma\_dump\_initial - segma\_dump\_final) + segma\_dump\_final;$$
- Dynamic of the standard deviation: $segma = segma \times segma\_dump$
- Rewording of genes: additive function +0.01
- Penalizing of the genes: subtractive function −0.01

The algorithms are implemented in Matlab 2014a code source. The computer used for our simulations is Intel® Core™ i5-2350M CPU 2.30 GHz and 8 GB of RAM. The results and the discussion are on the next subsection.

3.3 Results and Discussion

We experiment the two described algorithms with the functions listed above. The entire mathematical benchmarks are tested with dimensions 10, 30 and 50. Each function had been tested for 50 independent runs for the different dimensions shown above.

1. Convergence study: the algorithms used for comparison are stochastic. This means that their results differ from one run to another. Hence we need to repeat the same experiment for several times and then we compare the averaged values found. The detailed results are shown in Table 2. The results given in bold style mean that the corresponding algorithm has found a better solution, the averaged value of 50 independent runs, compared to the other. The table give the averaged optimal values found, the standard deviation and the processing time. It indicates also how many times the algorithm is executed for each instance (N). The results given in bold style mean that the corresponding algorithm performs the problem more than the other. We conclude from Table 2 that RNPSGA is batter that SGA. It gives 90.4762 % of good results. The proposed algorithm can be compared to more recent and efficient algorithms to measure objectively its quality (see the results shown in [15–17]).

Table 2 Average, standard deviation, number of success for each instance (N) and time processing of the best fitness values found by RNP-SGA and SGA

Algorithms		Function D								
		F1			F2			F3		
		10	30	50	10	30	50	10	30	50
RNP-SGA	Avr.	**0**	**0**	**0.001**	**0**	**2.E−6**	**0.94**	**10.46**	**65.41**	**166.2**
	St. d	0	0	0.006	0	7.E−7	1.51	11.81	36.59	64.49
	N	50	50	500	50.0	50.0	50.0	48.0	50.0	50.0
	T	3.54	6.58	9.55	3.642	6.726	9.723	3.847	6.912	9.926
SGA	Avr.	89.7	4350	14355	1.224	24.04	60.38	98.65	4532.	19768
	St. d	91.1	2004.	3301.	0.764	5.726	10.71	61.39	2130.	7089.
	N	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
	T	3.57	5.91	8.22	3.605	6.071	8.408	3.807	6.232	8.6
Algorithms		Function D								
		F4			F5			F6		
		10	30	50	10	30	50	10	30	50
RNP-SGA	Avr.	**6.72**	**34.16**	**76.77**	**0.012**	**6.E−4**	**0.023**	**0.399**	19.11	19.50
	St. d	2.49	9.850	14.90	0.015	0.002	0.017	2.794	3.902	2.754
	N	37.0	50.0	50.0	50.0	50.0	50.0	49.0	39.0	50.0
	T	3.63	6.691	9.699	3.954	6.926	9.914	3.865	6.857	9.823
SGA	Avr.	9.24	86.35	212.3	1.700	39.86	137.0	4.825	**19.05**	**17.02**
	St. d	3.79	16.69	28.35	0.960	13.42	28.85	1.592	1.607	0.046
	N	13.0	0.0	0.0	0.0	0.0	0.0	1.0	11.0	0.0
	T	3.60	6.091	8.431	3.960	6.278	8.673	3.830	6.181	8.540

Algorithms		Function D		
		F7		
		10	30	50
RNP-SGA	Avr.	**108.9**	**0.040**	**0.175**
	St. d	28.88	0.279	0.328
	N	44.0	50.0	50.0
	T	33.81	39.08	44.92
SGA	Avr.	154.6	169.7	226.4
	St. d	32.57	14.84	14.47
	N	6.0	0.0	0.0
	T	34.03	38.66	43.77

2. Algorithms' stability: the standard deviation is the most used indicator of stability. It measures the distribution of the results found around the averaged value of 50 independent runs. We conclude through this measure the accuracy of the proposed algorithm.
3. Computational time: the time that makes an evolutionary algorithm to generate the solution is an important factor. In real word applications we search for fast algorithms. The evolutionary algorithms are not recommended for online

problems in many applications. Add to this that real applications need high time for an algorithm to generate their solutions. If we compare the time processing of the proposed algorithm and the SGA, we can conclude that there are no difference between them. As SGA had been used in real-world applications, the RNP-SGA can be also used.

4 Realisation of Gymnastic Movement on the Bar by Humanoid Robot

In this section we propose the utilization of the RNPSGA to realize the swing-up movement on the bar by a humanoid robot. Firstly we need the dynamic model of the humanoid robot on the bar. Secondly, we derivate the linear model around the swing-up position. Then the RNPSGA can be applied to compute the matrix Q and R for the designing of the optimal LQR controller.

4.1 Virtual Model

We present here the virtual model designed using Solidworks 2014. The complete model of the humanoid robot hanging on a high bar is represented in Fig. 3. To control the humanoid robot, we combine between Solidwork and Simscape/Matlab. The model of the designed robot on Simmechanics/Matlab is shown in Fig. 4. Therefore, the exact mechanical properties of the humanoid are copied directly from Solidworks. The details are shown in Table 3.

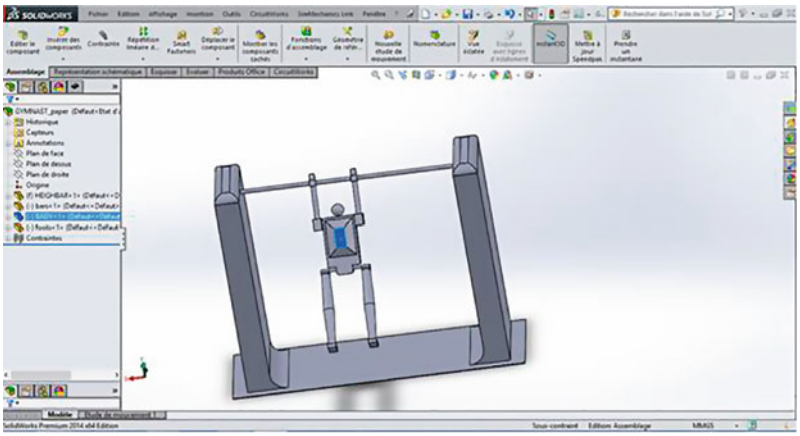

Fig. 3 Virtual model of the humanoid designed with Solidworks

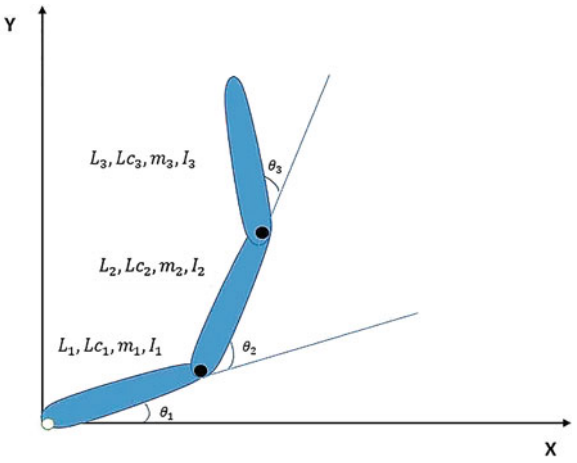

Fig. 4 The humanoid as a three link underactuated pendulum

Table 3 Parameters of the humanoid robot

The link	M	I	l_c	l
Link 1	8.0076	0.505	0.39874	0.50542
Link 2	31.8146	1.52895	0.23665	0.57
Link 3	21.2453	3.10134	0.48835	0.9767

4.2 The Nonlinear Dynamic Model of the Gymnastic Humanoid Robot

The Gymnastic humanoid robot can be seen as being compound of three main links [15, 18]. The first link represents the arms, the second represents the torso and a third represents the legs. The joints of the robot are therefore the hands, the shoulders and the hips. Both shoulders and hips are actuated by two servos in each side. The hands are not actuated indeed, and therefore the system can be approximated by a three link underactuated pendulum. The dynamic behavior of this multi-body robotic system can be derived from the classical Euler-Lagrange equations. The model is represented in Fig. 4.

We precise here the used notations:

- θ_i is the angle of joint i in respect to the previous link.
- m_i is the mass of link i.
- I_i is the inertia of link i.
- τ_i is the torque actuated on the active joint i.
- L_i is the length between joint i and joint i + 1.
- Lc_i is the length between joint i and the center of gravity of the mass i.

The direct cinematic model of the center of mass of each link is:

$$x_1 = Lc_1\, cos(\theta_1) \tag{1}$$

$$y1 = Lc_1 \sin(\theta_1) \tag{2}$$

$$x_2 = L_1\, cos(\theta 1) + lc_2\, cos(\theta_1 + \theta_2) \tag{3}$$

$$y_2 = L_1\, sin(\theta_1) + Lc_2\, sin(\theta_1 + \theta_2) \tag{4}$$

$$x_3 = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) + Lc_3 \cos(\theta_1 + \theta_2 + \theta_3) \tag{5}$$

$$y_3 = L_1 \sin(\theta_1) + \mathrm{L}_2 \sin(\theta_1 + \theta_2) + Lc_3 \sin(\theta_1 + \theta_2 + \theta_3) \tag{6}$$

The Lagrangian of the system is given by:

$$L = \sum_{i=1}^{3} (T_i - U_i) \tag{7}$$

where the kinetic energy T and the potential energy U of links are given by:

$$T_1 = \frac{1}{2}\left[m_1\left(\dot{x}_1{}^2 + \dot{y}_1{}^2\right) + I_1\dot{\theta}_1{}^2\right] \tag{8}$$

$$T_2 = \frac{1}{2}\left[m_2\left(\dot{x}_2{}^2 + \dot{y}_2{}^2\right) + I_1\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2\right] \tag{9}$$

$$T_3 = \frac{1}{2}\left[m_2\left(\dot{x}_3{}^2 + \dot{y}_3{}^2\right) + I_1\left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)^2\right] \tag{10}$$

$$U_1 = m_1 g y_1 \tag{11}$$

$$U_2 = m_2 g y_2 \tag{12}$$

$$U_3 = m_3 g y_3 \tag{13}$$

The nonlinear model is then derived from Euler-Lagrange equations. We found the model given by Eq. 14

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} \mathrm{G}_1 \\ \mathrm{G}_2 \\ \mathrm{G}_3 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_1 \\ \tau_2 \end{bmatrix} \tag{14}$$

where the inertial terms are:

$$d_{11} = \mathrm{A1} + 2m_2L_1Lc_2\,cos(\theta_2) + 2m_3L_1L_2\cos(\theta_2) + 2m_3L_2Lc_3\cos(\theta_3) + m_3L_1Lc_3\,cos(\theta_2+\theta_3)$$

$$d_{12} = \mathrm{A2} + m_2L_1Lc_2\,cos(\theta_2) + m_3L_1L_2\,cos(\theta_2) + 2m_3L_2Lc_3\,cos(\theta_3) + m_3L_1Lc_3cos(\theta_2+\theta_3)$$

$$d_{13} = A_3 + m_3L_2Lc_3\,cos(\theta_3) + m_3L_1Lc_3\,cos(\theta_2+\theta_3)$$

$$d_{21} = d_{12}$$

$$d_{22} = A_2 + 2m_3L_2Lc_3\,cos(\theta_3)$$

$$d_{23} = A_3 + m_3L_2Lc_3\,cos(\theta_3)$$

$$d_{31} = d_{13}$$

$$d_{32} = d_{23}$$

$$d_{33} = A_3 + m_3L_2Lc_3\cos(\theta_3)$$

$$A_1 = m_1Lc_1^2 + m_2L_1^2 + m_2Lc_2^2 + m_3L_1^2 + m_3L_2^2 + m_3Lc_3^2 + I_1 + I_2 + I_3$$

$$A_2 = m_2Lc_2^2 + m_3L_2^2 + m_3Lc_3^2 + I_2 + I_3$$

$$A_3 = m_3Lc_3^2 + I_3$$

The gravitational terms are:

$$G_1 = \mathrm{B}_1cos(\theta_1) + \mathrm{B}_2cos(\theta_1+\theta_2) + \mathrm{B}_4\,cos(\theta_1+\theta_2+\theta_3)$$

$$G_2 = \mathrm{B}_3cos(\theta_1+\theta_2) + B_4\,cos(\theta_1+\theta_2+\theta_3)$$

$$G_3 = B_4\,cos(\theta_1+\theta_2+\theta_3)$$

$$\mathrm{B}_1 = (m_1lc_1 + m_2l_1 + m_3l_1)g$$

$$\mathrm{B}_2 = (m_2lc_2 + m_3l_2)g$$

$$\mathrm{B}_3 = (m_2lc_2 + m_3l_2)g$$

$$B_4 = m_3lc_3g$$

And the Corriolis:

$$\begin{aligned} C_1 = & \mathrm{F}_4(\dot{\theta}_2+\dot{\theta}_3)(2\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)sin(\theta_2+\theta_3) \\ & +\mathrm{F}_1\dot{\theta}_2(2\dot{\theta}_1+\dot{\theta}_2)sin(\theta_2)+\mathrm{F}_2\dot{q}_2(2\theta_1+\theta_2)sin(\theta_2) \\ & +\mathrm{F}_3\dot{\theta}_3(2\dot{\theta}_1+2\dot{\theta}_2+\dot{\theta}_3)sin(\theta_3) \end{aligned}$$

$$\begin{aligned} C_2 = & -F_1\dot{\theta}_1^2\,sin(\theta_2)-\mathrm{F}_2\dot{\theta}_1^2\,sin(q2)+\mathrm{F}_3\dot{\theta}_3(2\dot{\theta}_1+2\dot{\theta}_2+\dot{\theta}_3)\,sin(\theta_3) \\ & -\mathrm{F}_4\dot{q}_1^4\,sin(\theta_2+\theta_3) \end{aligned}$$

$$C_3 = -\,\mathrm{F}_3(\dot{\theta}_1+\dot{\theta}_2)^2 sin(\theta_3)-\mathrm{F}_4\dot{\theta}_1^2\,sin(\theta_2+\theta_3)$$

$$F_1 = -m_2L_1Lc_2,\ \mathrm{F}_2 = -\,m_3L_1L_2,\ F_3 = -\,m_3L_2Lc_3,\ \mathrm{F}_4 = -\,m_3L_1Lc_3,$$

4.3 Linearization of the Model

We are interested by the realization of the gymnastic movement. In the literature we talk about the swing-up control problem [14]. The objective is to realize and to stabilize the humanoid at the vertical unstable equilibrium position. As the model derived from Euler-Lagrange equations is highly nonlinear, we derive here the linearized model around the vertical position.

$$\theta = \begin{bmatrix} \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T \text{ and } \dot{\theta} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

The linearized model of the humanoid for the vertical unstable equilibrium is given by the fooling equations:

$$D\ddot{q} + Gq = T\tau \tag{15}$$

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}1 \\ \ddot{\theta}2 \\ \ddot{\theta}3 \end{bmatrix} + \begin{bmatrix} g_{11} & \mathrm{g}_{12} & \mathrm{g}_{13} \\ \mathrm{g}_{21} & d_{22} & \mathrm{g}_{23} \\ \mathrm{g}_{31} & \mathrm{g}_{32} & \mathrm{g}_{33} \end{bmatrix} \begin{bmatrix} \theta_1 - \frac{\pi}{2} \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{16}$$

where:

$$d_{11} = \mathrm{A}_1 + 2m_2L_1Lc_2 + 2m_3L_1L_2 + 2m_3L_2Lc_3 + 2m_3L_1Lc_3$$

$$d_{12} = \mathrm{A}_2 + m_2L_1Lc_2 + m_3L_1L_2 + 2m_3L_2Lc_3 + m_3L_1Lc_3$$

$$d_{13} = A_3 + m_3L_2Lc_3 + m_3L_1Lc_3$$

$$d_{21} = d_{12}$$

$$d_{22} = A_2 + 2m_3 L_2 Lc_3$$

$$d_{23} = A_3 + m_3 L_2 Lc_3$$

$$d_{31} = d_{13}$$

$$d_{32} = d_{23}$$

$$d_{33} = A_3 + m_3 L_2 Lc_3$$

$$g_{11} = B_1 + B_2 + B_4$$

$$g_{12} = B_3 + B_4$$

$$g_{12} = \mathrm{B}_4$$

$$g_{22} = g_{21} = g_{12}$$

$$g_{33} = g_{31} = g_{32} = g_{23} = g_{13}$$

4.4 The State Space Model of the Humanoid

The state vector is constituted by the position vector $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, hence we have:

$$X = [\, q_1 - \tfrac{\pi}{2} \quad q_2 \quad q_3 \quad \dot{q}_1 \quad \dot{q}_2 \quad \dot{q}_3 \,]^T \tag{17}$$

The state space model is given by:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{18}$$

$$y(t) = Cx(t) + Du(t) \tag{19}$$

The matrix A, B, C and D are given by:

$$A = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ D^{-1}G & 0_{n\times n} \end{bmatrix}, B = \begin{bmatrix} 0_{n\times m} \\ D^{-1}T \end{bmatrix}, C = I_{n\times n} \text{ and } D = 0$$

The numerical values of the state space matrix A and B are:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 18.8797 & -21.4186 & -5.1879 & 0 & 0 & 0 \\ -18.3221 & 46.7787 & 8.3809 & 0 & 0 & 0 \\ -0.2202 & -10.0165 & 6.1066 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.2169 & 0.0326 \\ 0.4124 & -0.1147 \\ 0.1147 & -0.1410 \end{bmatrix}$$

4.5 Realization of the Gymnastic Movement by the Humanoid Robot

We propose in this subsection the designing of an intelligent optimal controller to stabilize the humanoid robot on the vertical unstable position (swing-up control). Therefore, we design firstly the LQR controller and then we optimize its parameters (the Q and R) by the RNP-SGA algorithm proposed in this paper and we compare the results given by RNPSGA and the results found by SGA. The optimization criterion is derived from the error between the trajectory of the variables X and the position of the swing up movement. The mathematical formula of this criterion is:

$$J = \sqrt{\int_{t_0}^{t_f} \left(X(t) - \hat{X}\right)^2} \tag{20}$$

where $X(t)$ is the trajectory of the state is space and $\hat{X}$ is the desired position. We note that $\widehat{X} = \left[\frac{\pi}{2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\right]^T$. The matrix Q and R are assumed diagonal. We propose here to compute them using the SGA and the proposed RNPSGA. The Simscape model is shown in Fig. 5. The result are shown in the next subsection.

4.6 Results and Discussion

We present here the results given by the SGA and the RNPSGA. The algorithms search for the diagonal elements of the matrix Q and R that minimize the criterion J. Once Q and R are known, we calculate the feedback K that stabilizes the robot on the vertical position. In LQR control, we search for K that minimizes the cost function given by:

$$C = \int \left\{ x' Q x + u' R u + 2 * x' N u \right\} dt \tag{21}$$

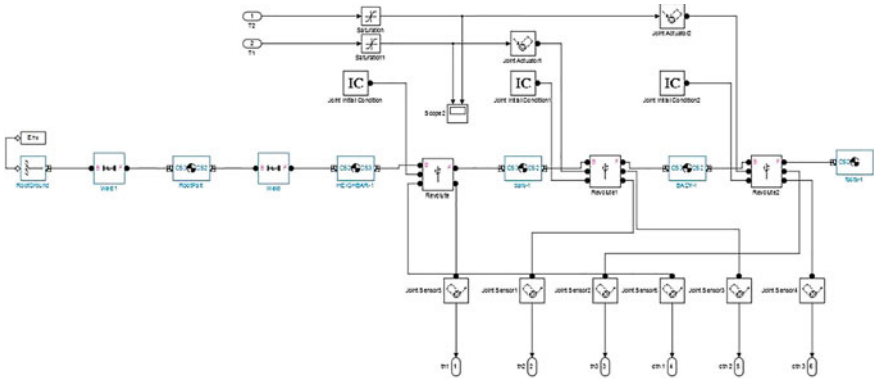

Fig. 5 Simscape model of the gymnastic humanoid robot

The feedback K is calculated using the algebraic Riccati equation. The low for the LQR controller is known to be: $u(t) = -k(t)x$

The results found by SGA and RNP-SGA are summarized in Table 4. The results found by RNPSGA are better than those found by SGA. The Fig. 6 gives the allure of the obtained gymnastic movement in virtual 3D space. The allures of the evolution of optimization process is shown in Fig. 7.

Table 4 The results found by SGA and RNP-SGA

Algorithm	Average of J	Q	R
RNPSGA	0.205	Diag[595.85 0.02 2.00E − 4 0.013 0.05 972.9751]	Diag[1.0E−5 1.0E−5]
SGA	0.431	Diag[494 1.0E − 5 1.0E − 5 1.0E − 5 39.175 968.3]	Diag[1.0E−5 1.0E−5]

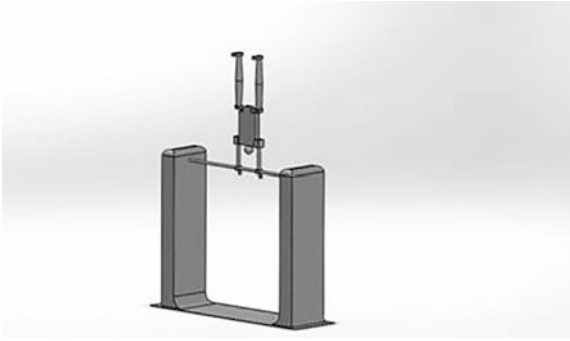

Fig. 6 The swing-up movement obtained

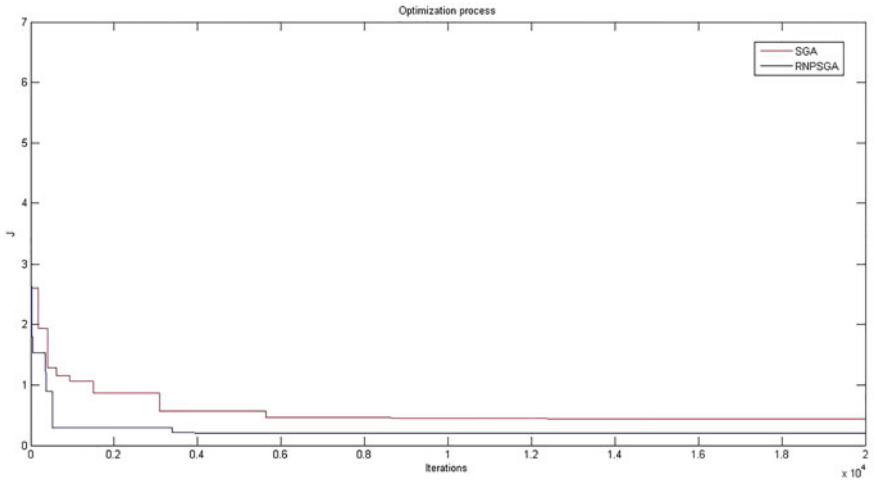

Fig. 7 Optimization process

5 Conclusion

This paper purposes a new variant of selfish gene algorithm called the Replaces and Never Penalizes Selfish Gene Algorithm (RNP-SGA). The algorithm uses tournament selection and overwrites the genes of failed individual by the alleles of the winner of the tournament. As the algorithm duplicates the genes of the winners of the tournament in the virtual population, then it allows more intensification. The proposed algorithm is tested firstly for unconstrained optimization problem using unimodal and multimodal benchmark functions. A statistical study shows its high performance compared to standard SGA in terms of quality of convergence, accuracy, stability and processing time. Secondly we test the algorithm on a humanoid robot. The objective is to stabilize the movement of the swing-up control. The model of the humanoid robot is designed using Solidwork specifically to do the movements of the gymnastic on horizontal bar. The results given by RNP-SGA are compared to those found by SGA.

References

1. Bremermann, H.J.: Optimization through evolution and recombination. In: Yovits, M.C. et al. (eds.) Self-Organizing Systems. Spartan, Washington, DC (1962)
2. Box, G.E.P.: Evolutionary operation: a method for increasing industrial productivity. Appl. Stat. **VI**(2), 81–101 (1957)
3. Rechenberg, I.: Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library translation No. 1122, Farnborough
4. Fogel, L.J.: Autonomous automata. Ind. Res. **4**, 14–19 (1962)

5. Burgin, G.H.: On playing two-person zero-sum games against nonminimax players. IEEE Trans. Syst. Sci. Cybern. **SSC-5**(4), 369–370 (1969)
6. Holland, J.H.: Outline for a logical theory of adaptive systems. J. Assoc. Comput. Mach. **3**, 297–314 (1962)
7. Neema, M.N., Maniruzzaman, K.M., Ohgai, A.: New genetic algorithms based approaches to continuous p-median problem. Springer Science (2008)
8. Chow, C.K., Yuen, S.Y.: An evolutionary algorithm that makes decision based on the entire previous search history. IEEE Trans. Evol. Comput. **15**(6), 741–768 (2011)
9. Yuen, S.Y., Devr, C.K.C.: A genetic algorithm that adaptively mutates and never revisits. IEEE Trans. Evol. Comput. **13**(2), 454–472 (2009)
10. Ghosh, D.: A Diversification Operator for Genetic Algorithms. In: OPSEARCH. Springer (2012)
11. Marsili Libelli, S., Alba, P.: Adaptive mutations in genetic algorithms. Soft Comput. (2000)
12. Dawkins, R.: The Selfish Gene, new edn. Oxford University Press, London (1989)
13. Corno, F., Reorda, M.S., Squillero, G.: The selfish gene algorithm: a new evolutionary optimization strategy. In: ACM Symposium on Applied Computing SAC'98, Atlanta (1998)
14. Xin, Xin, Kaneda, Masahiro: Swing-up control for a 3-DOF gymnastic robot with passive first joint: design and analysis. IEEE Trans. Robot. **23**(6), 1277–1285 (2007)
15. Liang, J.J., Suganthan, P.N., Deb, K.: Novel composition test functions for numerical global optimization. In: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005, pp. 68–75. IEEE (2005)
16. Chow, C.K., Yuen, S.Y.: An evolutionary algorithm that makes decision based on the entire previous search history. IEEE Trans. Evol. Comput. **15**(6), 741–769 (2011)
17. Yuen, S.Y., Chow, C.K.: A non-revisiting simulated annealing algorithm. Proceedings of IEEE Congress on Evolutionary Computation, pp. 1886–1892 (2008)
18. Teodoro, P.D.D.: Humanoid robot: development of a simulation environment of an entertainment humanoid robot. Master Thesis, Instituto Superior Técnico (2007)