Duelist Algorithm: An Algorithm Inspired by How Duelist Improve Their Capabilities in a Duel

Totok Ruki Biyanto[1(✉)], Henokh Yernias Fibrianto[1], Gunawan Nugroho[1], Agus Muhamad Hatta[1], Erny Listijorini[2], Titik Budiati[3], and Hairul Huda[4]

[1] Engineering Physics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
trb@ep.its.ac.id, joelhenokh@gmail.com
[2] Mechanical Engineering Department, Univesitas Sultan Ageng Tirtayasa, Cilegon, Indonesia
[3] Food Technology Department, State Polytechnic of Jember, Jember, Indonesia
[4] Chemical Engineering Department, Universitas Mulawarman, Samarinda, Indonesia

Abstract. This paper proposes an optimization algorithm based on human fight and learn from each duelist. The proposed algorithm starts with an initial set of duelists. The duel is to determine the winner and loser. The loser learns from the winner, while the winner try their new skill or technique that may improve their fighting capabilities. A few duelists with highest fighting capabilities are called as champion. The champion train a new duelist such as their capabilities. The new duelist will join the tournament as a representative of each champion. All duelist are re-evaluated, and the duelists with worst fighting capabilities is eliminated to maintain the amount of duelists. Several benchmark functions is used in this work. The results shows that Duelist Algorithm outperform other algorithms in several functions.

Keywords: Optimization · Algorithm · Duelist · Fighting

1 Introduction

Optimization is a process to achieve something better. Optimization usually consists of three parts which are optimization algorithm, model and objective function. For example, let there be a problem f(x) as a model then to find optimum value of x which is can be maximum, minimum or at specific value in between is the objective function using an algorithm. Different methods and algorithm have been proposed to solve the optimization problem. One of the most common methods used for optimization is genetic algorithm (GA) which is based on natural selection by evolving a population of candidate solution for defined objective function [1]. On the other hand, a different

Y. Tan et al. (Eds.): ICSI 2016, Part I, LNCS 9712, pp. 39–47, 2016.
DOI: 10.1007/978-3-319-41000-5_4

method for optimization called ant colony optimization is inspired by foraging behavior of real ants [2]. Another type of method is inspired by social behavior of animals which is called as particle swarm optimization (PSO) [3]. There's also a method for optimization which inspired by imperialistic competition called imperialist competitive algorithm (ICA) [4]. All of these mentioned methods are population based algorithm which is mean that there's a set of population and keep improving itself in each iterations [5]. There are other optimization algorithms also commonly used such as predatory search strategy [6], society and civilization optimization [7] and quantum evolutionary algorithm [8]. Nowadays, all this optimization methods are very useful for solving multiple problems such as energy management, scheduling, resource allocation, etc. [9–11].

In this paper, a new algorithm based on genetic algorithm is proposed which is inspired by human fighting and learning capabilities. As an overview, in genetic algorithm there are two ways to develop an individual into a new one. First is crossover where an individual mate with individual to produce a new offspring, this new offspring's genotype are based on their parents. The second one is mutation where an individual mutate into a new one. In duelist algorithm (DA), all the individual in population are called as duelist, all those duelists would fight one by one to determine the champions, winners and losers. The fighting itself just like real life fight where the strongest has possibility as a loser. There is a probability that the weak one would be lucky enough to win. In order to improve each duelist, there are also two ways to evolve. One of them is innovation. Innovation is only applicable to the winner. The other one is called as learning, losers would learn from winners. In GA, both mutation and crossover are seem to be blind in producing any solution to find the best solution. Blind means that each solution or produced individual in genetic algorithm may has not better solution. In fact, it may fall into the worst one. DA tries to minimize this blind effect by giving different treatment on duelists based on their classification. This paper described how duelist algorithm is designed and implemented.

2 Review of a Duel

Duel can be interpreted as a fighting between one or more person(s) with other person (s). Fighting require physical strength, skill and intellectual capability, for example in chess and bridge games. Common type of duel, which include physical strength is boxing, boxing is one of world's most popular sport where two persons need to knock down each of them under certain rules. In every duel, there are consist of the winner and the loser as well as the rules. In a match the probability become the winner depend on strength, skill and luck. After the match, knowing the capabilities of the winner and the loser are very useful. Loser can learn from how the winner, and winner can improve the capability and skill by training or trying something new from the loser. In the proposed algorithm, each duelist do the same to be unbeatable, by upgrading themself whether by learning from their opponent or developing a new technique or skill.

3 Duelist Algorithm

The flowchart of proposed algorithm is shown in Fig. 1. First, population of duelist is registered. Duelists have their properties, which is encoded into binary array. All of the duelists are evaluated to determine their fighting capabilities. The duel schedule is set to each duelist that contain a set of duel participants. In the duel, each duelist would fight one on one with other duelist. This one on one fighting is used rather than gladiator battle to avoid local optimum. Each duel would produce a winner and a loser based on their fighting capabilities and their 'luck'. After the match, the champion is also determined. These champions are the duelist that has the best fighting capabilities.

Then, each winner and loser have opportunity to upgrade their fighting capabilities, meanwhile each champion train the new duelist as such their capabilities. The new duelist will join in the next match. Each loser would learn from their opponents how to be a better duelist by replacing a specific part of their skillset with winner's skillset. On the other hand, winner would try to innovate a new skill by changing their skillset value.

Each duelist fighting capabilities is re-evaluated for the next match. All duelist then re-evaluated through post-qualification and sorted to determine who will be the champions. Since there are new duelists that was trained by champions, all the worst duelists are eliminated to maintain the amount of duelists in the tournament. This process will continue until the tournament is finished. The systematic explanation as follow:

3.1 Registration of Duelist Candidate

Each duelist in a duelist set is registered using binary array. Binary array in duelist algorithm is called as skillset. In a N_{var}-dimensional optimization problem, the duelist would be binary length times N_{var} length array.

3.2 Pre-Qualification

Pre-qualification is a test that given to each duelists to measure or evaluate their fighting capabilities based on their skillset.

3.3 Board of Champions Determination

Board of champions is determined to keep the best duelist in the game. Each champion should trains a new duelist to be as well as himself duel capabilities. This new duelists would replace the champion position in the game and join the next duel.

3.4 Duel Scheduling Between Duelists

The duel schedule between each duelist is set randomly. Each duelist will fight using their fighting capabilities and luck to determine the winner and the loser. The duel is

using a simple logic. If duelist A's fighting capabilities plus his luck are higher than duelist B's, then duelist A is the winner and vice versa. Duelist's luck is purely random. The pseudocode to determine the winner and the loser is shown in Algorithm 1.

Algorithm 1. Determination of the winner and the loser.

```
Require : Duelist A and B; Luck_Coefficient
Where : FC = Fighting Capabilities; LC = Luck Coefficient
A(Luck) = A(FC) * (LC + (random(0-1) * LC));
B(Luck) = B(FC) * (LC + (random(0-1) * LC));
If  ((A(FC) + A(Luck)) <= (B(FC) + B(Luck)))
  A(Winner) = 1;
  B(Winner) = 0;
Else
  A(Winner) = 0;
  B(Winner) = 1;

End
```

3.5 Duelist's Improvement

After the match, each duelist are categorized into champion, winner and loser. To improve each duelist fighting capabilities there are three kind of treatment for each categories. First treatment is for losers, each loser is trained by learning from winner. Learning means that loser may copy a part of winner's skillset. The second treatment is for winners, each winner would improve their own capabilities by trying something new. This treatment consist of winner's skillset random manipulation. Finally, each champion would trains a new duelist.

Algorithm 2. Winner and Loser Enhancement.

```
Require : Duelist A and B;
if  A(Winner) = 1;
  for i=1:(skillset_length)
    D = random(0...1);
    If D < Prob_Innovate
      A(skillset) = rand(0...9);
    end
  end
else
  for i=1:(skillset_length)
    E = random(0...1);
    If E < Prob_Learn
      A(skillset) = B(skillset);
    end
  end

  end
```

3.6 Elimination

Since there are some new duelists joining the game, there must be an elimination to keep duelists quantity still the same as defined before. Elimination is based on each duelist's dueling capabilities. The duelist with worst dueling capabilities are eliminated.

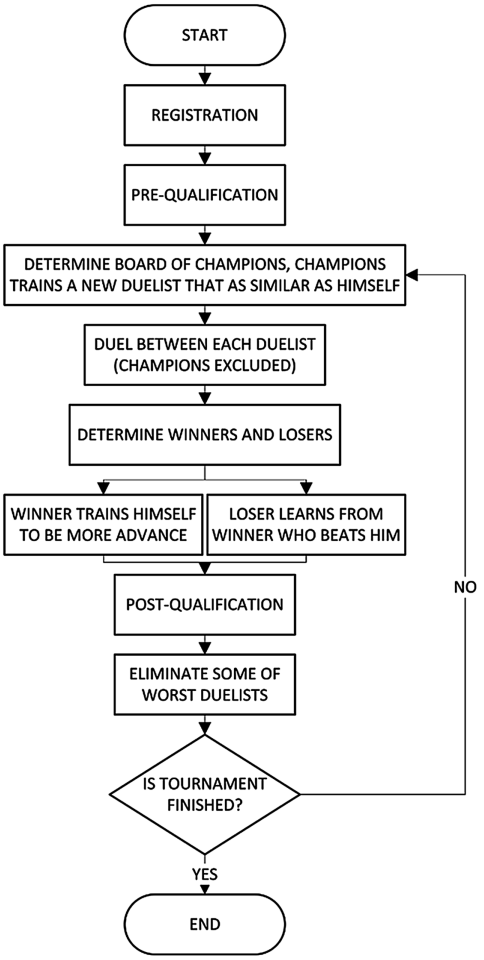

Fig. 1. Duelist algorithm flowchart

4 Experimental Studies

This section discuss about Duelist Algorithm performance using a benchmark for computational speed comparison and 10 other benchmarks for algorithm's robustness comparison. The detail of these functions are shown as follow:

$$f = -(\sqrt{x^2 + y^2} * \cos(x - y) * e^{\cos((x*(y+5)/7))}) \quad (1)$$

While 10 other benchmark functions are benchmark function with noise based on real-parameter black box optimization benchmarking [12]. The 10 mathematical optimization problems are f_{110}, f_{111}, f_{113}, f_{115}, f_{116}, f_{119}, f_{120}, f_{121}, f_{122}, f_{123}. The proposed algorithm is compared with a group of commonly used algorithms including Genetic Algorithm [1], Particle Swarm Optimization [3] and Imperialist Competitive Algorithm [4].

4.1 Benchmark Function

Duelist algorithm and several other algorithms are applied on total of 11 benchmark function. The first functions is used to test the algorithm's computational speed and other ten functions are used to test the algorithm's ability in finding the optimum value and robustness.

4.2 Parameter Setting

The proposed Duelist Algorithm has been tested for optimization problems to show the advantages of proposed algorithm. The first benchmark was repeated 10 times and the other ten were repeated 100 times. For the last ten function evaluation, total population and iteration that used in GA and DA is 100 and 500 respectively. Mutation and Crossover probability is set at 0.05 and 0.8. For DA, innovate and learning probability are set as 0.1 and 0.8 respectively with luck coefficient of 0.01. In PSO algorithm, the velocity constants are set 0.4 and 0.6 and the number of swarms at 100. The number colonies in ICA is set at 100, with initial number of imperialists of 8, revolution rate of 0.4 and number of decades of 500. Some change in algorithm's parameter are changed for the first benchmark. The changes in GA are max generation of 200 with mutation probability and crossover probability of 0.5 and 0.8 respectively. In ICA, the decades are shorten into 200 decades. In DA, luck coefficient of 0 and max generations of 200.

4.3 Result of First Evaluation

In this section, each algorithm is tested using first benchmark function. To provide a fair comparison, all algorithm used same initial position or population. The average result of the test are shown as follow:

The experiment shows that DA is able to reach global optimum under lesser number of iterations that GA and PSO (Fig. 2).

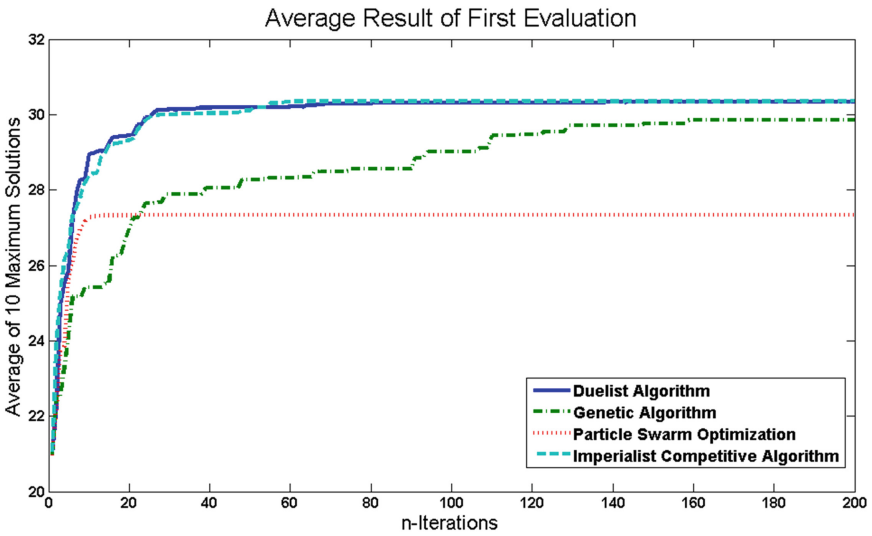

Fig. 2. Comparison result between algorithms (Color figure online)

4.4 Result of Second Evaluation

The statistical results of 100 experiments of ten noisy benchmark functions with twenty dimensions functions are presented in Table 1. The bold font shows the optimum value that achieved for each benchmark function. Based on the table, it can be observed that the proposed algorithm is able to overcome other algorithms. The mean values of proposed algorithms indicate that DA has a good robustness in finding optimum value.

Table 1. Comparison between algorithms for noisy benchmark function

f		GA	PSO	ICA	DA
110	min	614.99719	−21.51084	429.91794	**−125.28074**
	mean	4885.52966	847.83925	2648.06026	**280.93054**
111	min	−134.93092	−134.81238	−132.96055	**−135.01945**
	mean	−91.24741	−87.13642	−69.58788	**−107.01454**
113	min	−79.07112	−82.01971	−80.65960	**−83.57565**
	mean	−59.25298	−73.02693	−69.48691	**−79.75573**
115	min	−52.78967	−52.70470	−27.06354	**−69.60157**
	mean	378.56370	78.94026	219.80703	**45.72266**
116	min	295.49628	13.41322	21.22042	**−70.08705**
	mean	2240.40197	547.91493	685.23940	**−29.25039**
119	min	−57.12290	−57.75113	−57.64251	**−57.89313**
	mean	−56.13720	−57.33738	−56.95080	**−57.86155**

(*Continued*)

Table 1. (*Continued*)

f		GA	PSO	ICA	DA
120	min	−57.89985	−57.89981	−57.89982	**−57.89986**
	mean	−57.88350	−57.87494	−57.87659	**−57.88931**
121	min	−56.01884	−52.66815	−40.04706	**−57.82896**
	mean	−22.50195	64.25804	14.71232	**−52.39053**
122	min	−38.63358	−38.66433	−38.60398	**−38.67088**
	mean	−38.31492	−38.48721	−38.41675	**−38.51987**
123	min	**−38.71998**	−38.71978	−38.71992	−38.71989
	mean	−38.71506	−38.71104	−38.71265	**−38.71602**

5 Conclusion

In this paper, an optimization algorithm based on how duelist improve himself to win a fight is proposed. Each individual in the population is called duelist. Each duelist fight with other duelist to determine who is the winner and the loser. Winner and loser have their own way of improving themself. The winners are improved by learning theirself. In the other hand, loser improve himself by learning from the winner. After several improvements and duels, some duelists will become the best solution for given problem. The algorithm is tested by using 11 different optimization problems. The result shows that the proposed algorithm is able to surpass the other algorithm and present robust results.

References

1. Melanie, M.: An introduction to genetic algorithms, Cambridge, Massachusetts London, England, Fifth printing, vol. 3 (1999)
2. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theoret. Comput. Sci. **344**, 243–278 (2005)
3. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**, 33–57 (2007)
4. Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE Congress on Evolutionary computation, CEC 2007, pp. 4661–4667 (2007)
5. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. Eur. J. Oper. Res. **94**, 392–404 (1996)
6. Linhares, A.: Synthesizing a predatory search strategy for VLSI layouts. IEEE Trans. Evol. Comput. **3**, 147–152 (1999)
7. Ray, T., Liew, K.M.: Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans. Evol. Comput. **7**, 386–396 (2003)
8. Han, K.-H., Kim, J.-H.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans. Evol. Comput. **6**, 580–593 (2002)

9. Hartmann, S.: A competitive genetic algorithm for resource-constrained project scheduling. Naval Res. Logistics (NRL) **45**, 733–750 (1998)
10. Dandy, G.C., Simpson, A.R., Murphy, L.J.: An improved genetic algorithm for pipe network optimization. Water Resour. Res. **32**, 449–458 (1996)
11. Balci, H.H., Valenzuela, J.F.: Scheduling electric power generators using particle swarm optimization combined with the Lagrangian relaxation method. Int. J. Appl. Math. Comput. Sci. **14**, 411–422 (2004)
12. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: experimental setup (2010)