A HYBRID DIFFERENTIAL EVOLUTION METHOD AND ITS APPLICATION TO THE PHYSICAL TRAVELLING SALESMAN PROBLEM

Bang Le Hai, Takashi Ashida, Ruck Thawonmas, and Frank Rinaldo
Intelligent Computer Entertainment Laboratory
Ritsumeikan University

ABSTRACT

Differential Evolution (DE) is a simple and efficient evolutionary algorithm for optimization problems over continuous space. A variant of DE is the Down-hill Simplex method based on Differential Evolution (DSM DE) which has the advantage of converging faster than DE. However, the problem with DSM DE is that it doesn't guarantee to converge to a global optimum. In this paper, we present a way to improve DE by combining DE with DSM DE and the application of the new method to the problem of finding the optimum path in the physical travelling salesman problem.

I. INTRODUCTION

The physical travelling salesman problem

The Physical Travelling Salesman Problem (PTSP) is a modification of the well-known combinatorial optimization problem, the Travelling Salesman Problem (TSP). PTSP is a single player, real-time game, where the objective is to direct the agent to visit all waypoints scattered around a map as quickly as possible. PTSP can be seen as an abstract representation of video games characterized by two game elements: order selection, and steering. Examples of such games include CrystalQuest, XQuest and Crazy Taxi.

In the current PTSP competition[1], each map contains 10 waypoints and multiple obstacles. Figure 1 below shows an example of a map with obstacles and waypoints

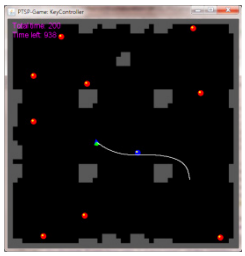


Figure 1 An example of a map

Differential Evolution (DE)

The DE algorithm [1] is a population based algorithm like genetic algorithms using the similar operators: crossover, mutation and selection. The main difference between DE and genetic algorithms is that genetic algorithms rely on crossover while DE relies on a mutation operation. DE has been proven as a simple, fast and efficient way to optimize functions in continuous space. Below are the main steps of DE, where *D* is the number of dimensions.

1. *Population initialization*: N_P vectors are chosen randomly to form the population.
2. *Mutation*: A target vector x is chosen randomly from the population, and a mutant vector v is generated according to

$$v = x_{r1} + F \times (x_{r2} - x_{r3}) \qquad (1)$$

where x_{r1}, x_{r2}, x_{r3} are three vectors randomly chosen from the population (x, x_{r1}, x_{r2} and x_{r3} are mutually different); F is a real constant factor.
3. *Crossover*: a trial vector u is formed by mixing the target vector x with the mutant vector v:

$$u_i = \begin{cases} v_i \; if \; (rand_i \leq CR) \; or \; i = rn_i \\ x_i \; if \; (rand_i > CR) \; and \; i \neq rn_i \end{cases}$$

where i=1, 2,…, D; $rand_i \in [0,1]$ is a random number; $CR \in [0,1]$ is the crossover constant; $rn_i \in 1,2,\dots,D$ is a randomly chosen index.
4. *Selection*: the trial vector is compared to the target vector using a greedy criterion. If vector u has a better cost function value than x then x is replaced by u, otherwise x is retained.
5. Repeat execution from step 2 to step 4 until a stopping criterion is met, usually a maximum number of iterations.

Down-hill Simplex method based Differential Evolution (DSM DE)

DSM DE [2] is a variant of DE using Down-hill simplex method [3]. The difference between DSM DE and classical DE is at the mutation step, in DSM DE the mutant vector is generated by

$$v = x_{ri} + F \times (x_{rbest} + x_{rworst}) \qquad (2)$$

where x_{rbest}, x_{rworst} is respectively the vector which has the best cost function value and the vector which has the worst cost function value among 3 vector x_{r1}, x_{r2}, x_{r3} and x_{ri} is the remaining one.

Combination of DSM DE and classical DE

Our experiments with DSM DE on many function optimization problems showed that DSM DE converges fast at first but is easily trapped in a local optimum. DE by contrast, has slower convergence speed but guarantees to converge to a global optimum with high confidence. Our idea is first using DSM DE to boost the speed of convergence and then use classical DE to find the true global optimum. The details are below:

At mutation step, we choose three random vectors x_{r1}, x_{r2}, x_{r3} and generate two mutant vectors v_a, v_b, where v_a is generated according to (1) andv_b is generated according to (2)

The final mutant vector v is created by mixing v_a and v_b. This process is done by first determining the number of

[1] http://www.ptsp-game.net/

978-1-4673-1501-2/12/$31.00 ©2012 IEEE

positions to mix (N_x) by (3) below, and then replacing the values at N_x random positions on v_a with the values of corresponding positions on v_b.

$$N_x = N_{max} \times \frac{F_x}{F_{start}} \qquad (3)$$

where N_{max} is a fixed number, F_x is the cost function value of target vector x, F_{start} is the cost function value of the population after initialization. (Note that we assume the task of optimization here is function miminization).

II. APPLICATION OF THE HYBRID DE IN TPSP

Lehmer code

To use DE in this game, we used Lehmer code to represent a path as a chromosome. The definition of Lehmer code is as follows:

if $P < P_0, P_1, P_2, \dots, P_{n-1} >$ is a permutation of the set of n element $0, 1, 2 \dots, n-1$ then the Lehmer code of the permutation P is a sequence of the numbers

$$\mathrm{L}(P) = < L(P)_0, L(P)_1, \dots, L(P)_{n-1}$$

where $L(P)_i = \#\{\mathrm{j} > \mathrm{i} : P_j < P_i\}$. In other words the term $L(P)_i$ counts the number of terms in $(P_0, P_1, P_2, \dots, P_{n-1})$ to the right of P_i that are smaller than it, a number between 0 and $n - i - 1$.

In this game, we numbered the waypoints from 0 to 9, so a path (solution) can be seen as a permutation of 10 elements $(0,1,\dots,9)$.

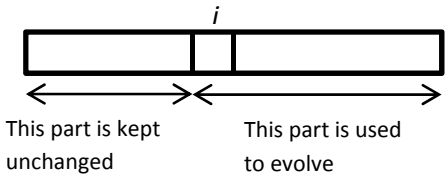
With the use of Lehmer code, we can define some operations on permutation as below:

If P, A are two permutations of n elements $< 0, 1, \dots, n >$

- $P + A = L^{-1}[(L(P)_0 + L(A)_0) \bmod \mathrm{n}, \dots]$
- $P - A = L^{-1}[(L(P)_0 - L(A)_0) \bmod \mathrm{n}, \dots]$
- $P \times k = L^{-1}[floor(L(P)_0 \times k) \bmod \mathrm{n}, \dots]$ (k is a real number).

The use of the hybrid DE in the game

From this point, for convenience, when we say "evolve the population from index i", that means only the part starting from index i of the Lehmer code of individuals in the population is used for mutation and crossover. The part from index 0 to $i - 1$ is the same for all the individuals in the population and kept unchanged during evolution.



Below is the process in detail:

- At the beginning of the game (time for constructing the controller)
  1. Set the current nearest waypoint as the first waypoint to visit. So if we define the solution path as an array s, we set $s[0]$ by the number attached to the nearest waypoint.
  2. Set $i = 1$, initialize the population and evolve the population from index i.
  3. At the end of this stage, set $i = i + 1$ and choose the current best individual b_P in the population and set $s[1] = b_P[1]$
- When the game has started, at each step:
  + If the ship is reaching the waypoint $s[i]$, evolve the population from index $i + 2$
  + If the ship has reached the waypoint $s[i]$:
    1. Choose $s[i+1]$ as the current target waypoint
    2. Choose the current best individual b_P in the population and set $s[i+2] = b_P[i+2]$, $s[i+2]$ will be waypoint visited next after $s[i+1]$
    3. Set $i=i+1$

  When there are only 3 waypoints left, the total number of candidates for the solution path is reduced to 6. At this point, we decide to stop using DE and instead use simulation to get the exact time to visit 3 left points in each case and choose the best one.

III. EXPERIMENTS AND RESULTS

Below are the results of the application of the new method to the PTSP compared to the nearest neighbor heuristic and the classical DE. Table 1 shows the average time spent of each controller. The controller which implements the hybrid DE or DE runs 50 times on each map. The constants are set to $CR = 0.9$; $F = 0.5$; $N_{max} = D$, $N_P = 25$.

Table 1 Comparisons of different controllers

	NN	Hybrid DE	DE
Map 1	1946	1734.52	1808.82
Map 2	1469	1464.02	1390.84
Map 3	1406	1330.18	1332.86
Map 4	2071	1984.86	1949.62
Map 5	2119	2050.76	2069.64
Map 6	2001	1921.02	1929.24
Map 7	2190	2041.8	2037.3
Map 8	1838	1538.48	1542.714

IV. CONCLUSION

In this paper we presented a new hybrid DE method by combining the classical DE with the DSM DE and how to apply it to find a good path in the PTSP. The results showed that the hybrid DE can be applied quite well to the PTSP and it generally has better results than the nearest neighbor heuristic. Compared with the classical DE algorithm, the new method is a little bit better although in some cases, the classical DE gives better results. The reason is that the spent time by the ship to reach all the waypoints does not only depend on the length but also the complexity of the path. In our current implementation, we used a simple way to evaluate the length and complexity of the path and in the future we'd like to find a way to evaluate more accurately the time spent for a given path in order to achieve better results.

REFERENCES

[1] R. Storn, "Differential Evolution, A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11, pp. 341-359, 1997.

[2] D. Kamiyama, K. Tamura, and K. Yasuda, "Down-hill Simplex Method Based Differential Evolution", The transactions of the Institute of Electrical Engineers of Japan. C, 130(7), pp. 1271-1272, 2010. (in Japanese)

[3] J.A. Nelder and R. Mead, "A Simplex Method for Function Minimization", Computer Journal, Vol. 7, pp. 308-313, 1965.