混合粒子群優化及噪音擾動法求解最短路徑問題
Hybrid Particle Swarm Optimization and Noising Method for Solving the Shortest Paths Problem

邱錦清 許正憲 王龍發 劉憶瑩
德明財經科技大學資訊管理系
Chin-Ching Chiu, Cheng-Hsien Hsu, Lung-fa Wang, Yi-Ying Liu
Department of Management Information System, Takming University of Science and Technology
E-mail: chiu@takming.edu.tw

Abstract

A maze or labyrinth is a network of passages, usually intricate and confusing. Finding a path through a maze is a basic computer science problem that can take many forms. In this paper, we consider the case where a maze has a single goal location, and agents must find a path to that goal from starting point. This paper presents a particle swarm optimization (PSO) based algorithm to solve maze pathfinding problem. When generates a path accoding to particle's position, the priority-based decoding is used. Moreover, noising method of local search is to improve the solution. The proposed algorithm is to find a path that is shortest, or nearly shortest with respect to maze.

Keywords: Shortest path problem, particle swarm optimization, noising method

1. Introduction

Pathfinding usually takes the form of a state-space search applied to a two-, or three-dimensional map [6]. Each state describes a position on the map and application of a search algorithm will reveal a route between two points on the map. The computer/video games industry makes extensive use of pathfinding, with virtually every game currently available incorporating some form of agent-based AI and some spatial aspect. A* is often the algorithm of choice, its popularity coming from many sources including ease of implementation, its efficiency and the huge body of experience built up among the game programming community [1]. Pathfinding is the simple process of finding a route from one point to another comes with ease to humans as well as animals and is as essential to survival as is to convenience. On the other hand, it is a remarkably difficult task to replicate in the artificial world. Because it is essential to numerous technological applications, notably autonomous locomotion of mobile robots, movement of agents in games and mazes [7-8], route planning for electronic maps and so forth, intensive research has been performed into solving this task. A great deal of effort has been put into coming up with algorithms which generate the shortest/fastest possible paths [4-5], however, these have continuously incurred the penalty of either taking extraneously long periods of time to compute or requiring memory capacities that may not even be present in todays supercomputers. Regardless, it has been recognized that for the majority of practical applications, such optimal paths are almost never required and near-optimal paths will more than suffice.

There has been a recent interest in the field of the particle swarm Optinization (PSO) [2-3]. The basic idea is to imitate the focking of birds in order to solve combinatorial optimization problems within a reasonable amount of time.

2. The PSO with noising method methodology

The development of our algorithm is described as follows.

2.1 Problem formulation

The maze path planning problem is typically formulated as follows: given a maze, we need to plan a path between two specified locations, a start and end point. The path should be free of collision and satisfies certain optimization criteria (i.e. shortest path) [ ]. For a given static environment of maze with six columns and seven rows as figure 1, the black grids represent an obstacle, the white grids denotes channel, s is the source, and g is the destination.

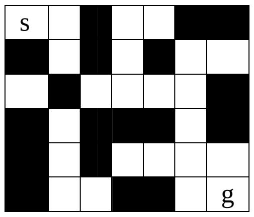

Figure 1. A maze M with 42 grids (m=6, n=7)

In figure 2, there are eight allowable direction each grid can be moved.

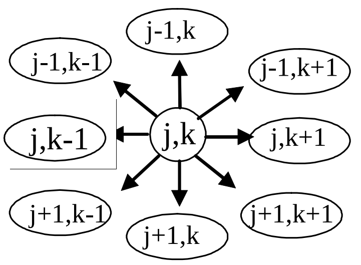

Figure 2. The possible moving direction of each grid for each particle *i*

We initialize the position and velocity of the maze as in figure 3, the negative infinite value ($-\infty$) denotes obstacle. The *x* and *v*, whose value between 0.0 and 1.0, represent the position and velocity of a particle, respectively.

$$\begin{bmatrix} x & x & -\infty & x & x & -\infty & -\infty \\ -\infty & x & -\infty & x & -\infty & x & x \\ x & -\infty & x & x & x & x & -\infty \\ -\infty & x & -\infty & -\infty & -\infty & x & -\infty \\ -\infty & x & -\infty & x & x & x & x \\ -\infty & x & x & -\infty & -\infty & x & x \end{bmatrix} \qquad \begin{bmatrix} v & v & -\infty & v & v & -\infty & -\infty \\ -\infty & v & -\infty & v & -\infty & v & v \\ v & -\infty & v & v & v & v & -\infty \\ -\infty & v & -\infty & -\infty & -\infty & v & -\infty \\ -\infty & v & -\infty & v & v & v & v \\ -\infty & v & v & -\infty & -\infty & v & v \end{bmatrix}$$

(a) position data (b) velocity data

Figure 3. Initialization of the position and velocity of the maze M for a particle *i*

A path from grid $g_{i,j,k}$ to grid $g_{i,j+x,k+y}$ is a sequence of grids ($g_{i,j,k}$, $g_{i,j+1,k+1}$,…, $g_{i,j+x,k+y}$) in which no grid is repeated.

2.2 PSO scheme

PSO is a versatile population-based optimization technique, in many respects similar to evolutionary algorithms. Basically, particles fly above the fitness landscape, while a particle's movement is influenced by its attraction to its neighborhood best, and its personal best. Rui Mendes [9] discusses the complete information about the particle swarm optimization. In the standard PSO algorithm, all particles have their position, velocity, and fitness values. The basic elements of PSO is summarized as follows:

- Particle: X_i^t is a candidate solution *i* in swarm at iteration *t*. The *i*th particle of the swarm is represented by a *mn*-dimension space and can be defined as $X_i^t = [x_{i,j,k}^t \mid_{j=1,2,\ldots,m,\, k=1,2,\ldots,n}]$, where *x*'s are the optimized parameters and $x_{i,j,k}^t$ is the position of the *i*th particle with respect to *jk*th dimension (say, *j*th row, *k*th column).

- Population: X^t is the set of *ps* particles in the swarm at iteration *t*, such that, $X^t = [X_1^t, X_2^t, \ldots, X_{ps}^t]$.

- Particle velocity: V_i^t is the velocity of particle *i* at iteration *t*. It can be described as $V_i^t = [v_{i,j,k}^t \mid_{j=1,2,\ldots,m,\, k=1,2,\ldots,n}]$ which a constant $V_{\max}$ (often set to 4) is used to limit the range of $v_{i,j,k}^t$, i.e., $v_{i,j,k}^t \in [-V_{\max}, V_{\max}]$ for avoiding the particle to converge to local optima.

- Particle best: PB_i^t is the best value of the particle *i* obtained untile iteration *t*. The best position position associated with the best fitness value of particle *i* obtained so far is called particle best $PB_i^t =$ $[pb_{i,j,k}^t \mid_{j=1,2,\ldots,m,\, k=1,2,\ldots,n}]$ and defined as with the fitness function $f(PB_i^t)$.

- Global best: GB^t is the best position among all paticles in the swarm, which is achieved so far and can be expressed as $GB^t =$ $[gb_{j,k}^t \mid_{j=1,2,\ldots,m,\, k=1,2,\ldots,n}]$ with the fitness function $f(GB_i^t)$.

The current velocity of the *jk*th dimension of the *i*th particle is updated as follows:

$$\begin{aligned} v_{i,j,k}^t &= wv_{i,j,k}^{t-1} + cognitive(t-1) + social(t-1) \\ cognitive(t-1) &= c_1 r_1 (pb_{i,j,k}^{t-1} - x_{i,j,k}^{t-1}) \\ social(t-1) &= c_2 r_2 (gb_{j,k}^{t-1} - x_{i,j,k}^{t-1}) \end{aligned} \tag{1}$$

where c_1 and c_2 are acceleration coefficients which were often set to be 2.0 according to past experience and r_1 and r_2 are uniform random numbers between [0,1]. The inertia weight *w* which is a parameter to control the impact of the previous velocities on the current velocity and can be dynamically varied by applying an annealing scheme [10].

In the above discussion, PSO is restricted in real number space. The resulted changes in position are defined as formula 3, such that $x_{i,j,k}^{t+1} = x_{i,j,k}^t + v_{i,j,k}^t$, for $j = 1,2,\ldots,m,\ k = 1,2,\ldots,n$.

$$X_i^{t+1} = X_i^t + V_i^t \tag{2}$$

The SPP is to find a path between two given grids having minimum total distance. In PSO, the quality of a particle (solution) is measured by a fitness function. For the SPP, the fitness function is obvious as the goal is to find the minimal distance path. Thus, the fitness of the i^{th} particle is defined as

$$f(X_i) = Dist(X_i) + \nabla(X_i) \tag{3}$$

$$Dist(X_i) = \sum_{seq=1}^{N_i-1} dist(g_{i,w,x}, g_{i,y,z}) \tag{4}$$

Where $g_{i,w,x}=PP_i(seq)$, $g_{i,y,z}=PP_i(seq+1)$, PP_i is the set of sequential grid IDs for the i^{th} particle, $N_i=|PP_i|$=number of grids that constitute the path represented by the i^{th} particle, and $dist(g_{i,w,x}, g_{i,y,z})$ is the distance of between the grid $g_{i,w,x}$ and grid $g_{i,y,z}$, such that $dist(g_{i,w,x}, g_{i,y,z}) = \sqrt{(w-y)^2+(x-z)^2}$. Thus, the fitness function takes minimum value when the shortest path is obtained. If the path represented by a particle happens to be an invalid path, its fitness is assigned a penalty value so that the particle's attributes will not be considered by others for future search. The penalty function of a particle is defined as follows.

$$\nabla(X_i) = \begin{cases} 0 & \text{if valid path} \\ \infty & \text{if invalid path} \end{cases} \tag{5}$$

The particle best position of each particle is updated using the following equation.

$$PB_i^t = \begin{cases} PB_i^{t-1} & if \quad f(X_i^t) \geq f(PB_i^{t-1}) \\ X_i^t & if \quad f(X_i^t) < f(PB_i^{t-1}) \end{cases} \tag{6}$$

Finally, the global best position found so far in the swarm population is obtained for $1 \leq i \leq ps$ as

$$GB^t = \begin{cases} PB_i^t & \arg\min f(PB_i^t) \\ & if \quad \min f(PB_i^t) < f(GB^{t-1}) \\ GB^{t-1} & otherwise \end{cases} \tag{7}$$

2.3 Path encoding and decoding

The main issue in applying PSO to the SPP is the encoding of a maze path into a particle in PSO. This encoding in turn affects the effectiveness of a solution/search process. The guiding information are the priorities of every grids in the maze. During PSO initialization, these priorities are assigned randomly. The path is generated by sequential grid appending procedure beginning with the source grid and terminating at the destination node, the procedure is referred as to path growth strategy. At each step of path construction from a particle, there are usually several grids available for consideration and the one with the highest priority is added into path and the process is repeated until the destination node is reached. For effective decoding, a dynamic grid adjacency matrix is maintained in the computer implementation and is updated after every node selection so that a selected grid is not a candidate for future selection [13].

```
//$S(g_{i,j,k})$ is the allowable moving adjacent grids of $g_{i,j,k}$
Particle_decoding($X_i$)
$g_{i,j,k}$ =source grid
$x_{i,j,k}$ =$-\infty$
$cnt=0$
PATH($cnt$)={ $g_{i,j,k}$ }
while({$g_{i,x,y} \in S(g_{i,j,k})$, and $x_{i,x,y} \neq -\infty$ } $\neq$ {})
        $cnt=cnt+1$
        $g_{i,x,y}$=argmax{$x_{i,x,y}$| $g_{i,x,y} \in S(g_{i,j,k})$,$x_{i,x,y} \neq -\infty$ }
        $g_{i,j,k} = g_{i,x,y}$
        PATH($cnt$)=PATH($cnt$) $\cup$ { $g_{i,j,k}$ }
        $x_{i,j,k}=-\infty$
        if ($g_{i,j,k}$ =goal grid)
           return the path PATH($cnt$)
        else
              consider the neighborhood of grid $g_{i,j,k}$, fill
              dead end for dynamic reducing the maze.
end while
return invalid_Path
```

Figure 4 The priority-based decoding procedure

2.4 Noising method

The local search essentially diverdifiers the search scheme. Recently, one such efficient metaheristics called noising method, was proposed by Chron and Hurdy [11,12]. For computation of the optimum of a combinatorial optimization problem, instead of taking te genuine data into account directly, they are perturbed by some progressively decreasing "noise" while applying local search. The noising method used here is based on noising the variations in the optimizing function *f*, that is, perturbing the variations of *f*. When a neighbor solution X' of the solution X is computed by applying an elementary transformation [11, 12] to X, the genuine variation $\Delta f(X,X') = f(X') - f(X)$ is not considered, but a noised variation $\Delta f_{noised}(X,X') = \Delta f(X',X) + \zeta^k$ is used, where ζ^k denotes the noise at each trial *k* and depends the noise rate (*NR*). Similar to iterative descent method in a function minimization problem, if $\Delta f_{noised}(X,X')$ < 0, X' becomes the new current solution, otherwise *X* is kept as the current solution and another neighbor of *X* is tried. The generic elementary transformation used for local neighborhood search is the swappong of grid priority values at two randomly selected of a particle priority (position) vector and two such swapping transformations are successively applied in each trial for generation a trial solution in the local search [13].

2.5 The proposed algorithm

The particle swarm optimization algorithm to minimize SPP in a given maze presents as follows.

Algorithm PSOSPP

```
Step 1. initialize $t$=0, $f(GB^t)=\infty$, $GB$={},
          $PATH_{global}$={}, $PATH_i$={} for $i$=1,…,$ps$.
Step 2. fill dead end for reducing the maze on the
          efficiency consideration, except grid $s$ and $g$.
Step 3. for each particle $i$,
          initialize each allowable moving grid's position
            $x_{i,j,k}$ in $X_i^t$ randomly from [0,1],
          initialize each allowable moving grid's velocity
            $v_{i,j,k}$ in $V_i^t$ randomly from [0,1].
Step 4. for each particle $i$,
          (a) Generate path $PATH_i$= Particle_Decoding($X_i^t$).
          (b) Evaluate fitness value $f(X_i^t)$ using Eq. (3)
          (c) Local search using noising method described in section 2.4.
Step 5. Update particle best position and global best
          position according to Eq.(6) and Eq. (7),
          respectively.
Step 6. Update $PATH_{global}$ and $PATH_i$ for $i$=1,…,$ps$.
Step 7. Update velocity : update the $i^{th}$ particle velocity
          using Eq. (1) restricted by maximum and
          minimum threshold $V_{\max}$ and $-V_{\max}$.
Step 8. Update position: update the $i^{th}$ particle position
          using Eq. (2).
Step 9. t=t+1
Step 10. Repeat step 3 to 9 until a given maximum
          number of iterations $iter_{max}$ is achieved.
Step 11. Output the best path $PATH_{global}$ and its
          distance.
```

3. Computer simulation and conclusion

The accuracy and efficiency of the proposed algorithm were verified by implementing simulation programs in Java language executed in Intel Pentium D CPU 3.4GHz with 512MB-DRAM on MS-Windows XP. We use some simulation case of different size of maze between 20×20 and 100×100. The velocity of each particle is updated according to different method as follows: (a) an inertia weight w=1, (b) an inertia weight w=0.5, (c) a dynamic inertia weight w=0.9~0.4 varing according to loop iteration, (d) a constriction factor K=0.729843788, and c_1=c_2=2.05 [14], (e) a constriction factor $K=2/(|2-c-\sqrt{c^2-4c}|)$ where c= c_1+c_2 and c>4 [14].

The figure 5 show a path in the maze M obtained by PSOSPP, the shortest tour (path) form source grid s to destination grid g is represented as ($g_{opt,1,1}$, $g_{opt,2,2}$, $g_{opt,3,3}$, $g_{opt,3,4}$, $g_{opt,3,5}$, $g_{opt,4,6}$, $g_{opt,5,7}$, $g_{opt,6,7}$) which the shortest path is obtained by paticle *opt*. Assume size of each grid is 1, the distance of this path is 8.65685.

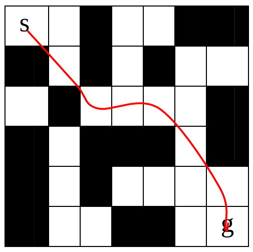

Figure 5 A possible tour of the maze M

In our simulation, the proposed algorithm may obtain the shortest path in most cases, and the computation time seems to be significantly shorter than that needed for the exhaustive method. When the proposed method fails to give an exact solution, the deviation from the exact solution is very small. The technique presented in this paper would be helpful for readers to understand the correlation between pathfinding, and maze environment.

References

[1] Cain T., Practical Optimisations for A* Path Generation. AI Game Programming Wisdom, Charles River Media 2002.

[2] Kennedy J. and Eberhart R. C., Particle swarm optimization, Proc. IEEE International Conference on Neural Networks, IV, Piscataway, NJ, pp. 1942~1948, 1995.

[3] Shi Y. H. and Ebehart R. C., A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998.

[4] Hart P. E., Nilsson N. J., and Raphael B., A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, SSC-4(2): 100-107, 1968.

[5] Hart P. E., Nilsson N. J., and Raphael B., Correction to "A formal basis for the heuristic determination of minimum cost paths. SIGART Newsletter, 37: 28-29, 1972.

[6] Reese B., and Stout B., Finding a Pathfinder. AAAI Spring Symposium on Artificial Intelligence and Computer Games, pp. 69-72, 1999.

[7] Scott G. V, and Zach Matley, Evolving Sparse Direction Maps for Maze pathfinding. IEEE Transactions on Evolutionary Computation, 1(1):835-838, 2004.

[8] Stentz A., Optimal and Efficient Path Planning for Partially-Known Environments. Proceedings of

the IEEE International Conference on Robotics and Automation (ICRA ’94), vol. 4, pp. 3310–3317, 1994.

[9] Rui Mendes, James Kennedy and Jose Neves, The Fully Informed Particle Swarm Simpler, Maybe Better. IEEE Transactions of Evolutionary Computation, vol. 1, no. 1, 2005.

[10] S. N. Sivanandam, P. Visalakshi and A. Bhuvaneswari, Multiprocessor scheduling using hybrid particle swram optimization with dynamically varying inertia. International Journal of Computer Science Applications vol. 4, no. 3, 95-106, 2007.

[11] I. Charon and O. Hurdy, The noising method : a new method for combinatorial optimization, Operations Research Letters, vol. 14, no. 3, pp. 133-137, 1993.

[12] I. Charon and O. Hurdy, The noising method : a generalization of some metaheruistics, European Journal of Operational Research, vol. 135, no. 1, pp. 86-101, 2001.

[13] Ammar W. Mohemmed and Nirod Chandra Sahoo, Effient Computation of Shortest Paths in Networks using Particle Swarm Optimization and Noising Metaheuristics, Hindawi Publishing Corporation Discrete Dynamic in Nature and Society, vol. 2007, Article ID 27383, pp. 1-25, 2007.

[14] M. Clerc and J. Kennedy, The particle swarm explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.