Proceedings of the 2013 International Conference on Machine Learning and Cybernetics, Tianjin, 14-17 July, 2013

AUTONOMOUS ROBOT PATH OPTIMIZATION USING FIREFLY ALGORITHM

MICHAEL BRAND, XIAO-HUA YU

Department of Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA 93407, USA
E-MAIL: xhyu@calpoly.edu

Abstract:

Path planning is an NP-complete problem with numerous practical applications, and is especially important for the navigation and control of autonomous robots. However, due to its computational complex nature, an optimal solution is often very difficult to be found using traditional methods.

In this research, a swarm intelligence approach inspired by the biological behavior of glowworms is studied and applied to the robot path optimization problem. Computer simulation results show this firefly algorithm can successfully find the optimal path in a dynamic environment, and outperforms the ant colony algorithm (ACO) for a larger grid workspace in terms of both path length and computational cost.

Keywords:

Glowworm swarm optimization; Robot path planning; Firefly algorithm

1. Introduction

Path planning has been a crucial and productive research area in the field of robotics. The basic motion planning problem is to find a collision-free path for a rigid moving object such as a robot to avoid obstacles ([6]). It is known that path planning is a NP (nondeterministic polynomial time) complete problem [6]. That is, the computational time required to solve such problem increases dramatically (usually in an exponential rate) when the size (or dimension) of the problem increases. Therefore, it is difficult to implement traditional optimization methods for robots to find the shortest path in real time.

Computational intelligence (CI) has become an important research area in recent years. Many bio-inspired models and algorithms have been developed, including artificial neural networks (ANN), genetic algorithms, particle swarm optimization (PSO), etc. ([5] [11] [12]) These approaches provide new and powerful computational tools for engineers and mathematicians to solve complex problems where closed-form solutions are often difficult to obtain.

Swarm intelligence (SI) is based on the collective behavior of decentralized, self-organized biological systems. A typical swarm intelligent system is made of a population of agents which only interact locally with each other and their environment. These agents follow very simple rules (with certain degree of randomness), yet such simple individual behavior can lead to the emergence of very complicated global behaviors of the overall system which may be unknown to each individual agent. Many swarm intelligent systems can be found in nature, such as ant colonies, bird flocking, bacterial growth, fish schooling, etc.

Ant colony optimization (ACO) is one of the most successful examples of swarm intelligent systems and has been applied to solve many different types of problems, including nonlinear function optimization and network routing in telecommunication networks. Inspired by the behavior of biological ants in real world, it is a meta-heuristic approach with multi-agents, in which each single agent is called an artificial ant [4].

Glowworm swarm optimization (GSO) was first introduced by Krishnanand and Ghose in 2006 ([1][2]). It was further developed by X. Yang in 2009 ([3]) and named as the firefly algorithm (FA). This approach shares a few common features with ACO but also has some significant differences.

In nature, biological glowworms or fireflies can emit light to attract their mates. Similarly, the agents in GSO algorithm carry a luminescence quantity called luciferin along with them, and "broadcast" it to their neighbors. The fitness function, or the objective function of optimization, is usually related with the light intensity of glowworms. Each glowworm can detect lights emitted from its neighbors within its sensory range, and move towards the one that has a luciferin value higher than its own in a probabilistic manner. Thus, by updating the movement and brightness of each agent, the algorithm can converge to a solution in a natural and metaheuristic way.

Though firefly algorithm is a relatively new approach, it has been successfully applied for multimodal function optimization [7] and power plant load and emissions dispatch (to minimize both fuel cost and emission of generating units)

978-1-4799-0260-6/13/$31.00 ©2013 IEEE

[8] [9]. In this research, the firefly algorithm (FA) is applied to path planning problem. Considering a two dimensional workspace with a pair of starting and ending points, the goal is to find an optimal solution that minimizes the path length. In the previous research such as [11], path planning with static (known) obstacles is considered using FA (off-line). In this paper, we extend the situation to a dynamic environment to further demonstrate the feasibility of this algorithm. That is, obstacles appear in the workspace after the robot leaves its initial starting point so that the robot must be navigated by the algorithm on-line to find the best path and avoid obstacles. Another contribution of this research is to compare the performance of FA with ant colony optimization algorithm (ACO) for various sizes of workspace, in terms of both path length and computational cost.

The organization of this paper is as follows. Section 2 discusses the details of Glowworm swarm optimization and the firefly (FA) algorithm. Computer simulation results are presented in section 3. Section 4 concludes the paper and gives directions for future work.

2. Glowworm Swarm Optimization

Glowworm swarm optimization is one of the newly developed swarm intelligence approaches. It is inspired from the light emission behavior of glowworms in nature and was first introduced in 2005 ([1][2]). It was then further developed by X. Yang in 2009 ([3]) and named as the firefly algorithm (FA). It has been successfully employed to find the optimal values of various test functions ([7]).

The FA algorithm is based on the following three simple rules ([7]): 1) all fireflies are unisex so that one firefly is attracted to other fireflies regardless of their sex; 2) attractiveness is proportional to fireflies' brightness, thus for any two flashing fireflies, the less brighter one will move towards the brighter one. The attractiveness also decreases as their distance increases. If no one is brighter than a particular firefly, it moves randomly; 3) the brightness or light intensity of a firefly is affected or determined by the objective function to be optimized.

In each iteration of FA algorithm, a firefly moves towards a brighter firefly in its neighborhood based on a probability transition rule. Let $p_{i,j}$ be the probability of the i^{th} firefly moves towards the j^{th} firefly, then

$$p_{i,j} = \frac{\tau_j}{\sum_{k \in N_i} \tau_k} \tag{1}$$

where $\tau_j > \tau_i$; τ_i, τ_j and τ_k is the light intensity of the i^{th}, j^{th}, and k^{th} firefly, respectively; N_i is the set that satisfies $N_i = \{d(i,k) < r_i, \tau_k > \tau_i\}$. That is, N_i includes all the fireflies in the sensory area (neighborhood) of the i^{th} firefly that are brighter than it. *d(i,k)* is the (Euclidean) distance between the i^{th} and k^{th} firefly; and the k^{th} firefly is in the sensory area (neighborhood) of the i^{th} firefly if this distance is less than a pre-set value r_i. Note light intensity dims as the distance between two fireflies increases:

$$\tau(d) = \tau_0 e^{-\gamma d} \tag{2}$$

where τ_0 is the original light intensity and γ is the light absorption coefficient, d is the distance. For simplicity, an inverse distance law may be employed instead of Eq. (2):

$$\tau(d) = \frac{\tau_0}{1+\gamma d} \tag{3}$$

In each iteration, the i^{th} firefly moves to the j^{th} (brighter) firefly in its neighborhood based on the probability defined in Eq. (1) and its new position can be calculated by:

$$x_i(n+1) = x_i(n) + \eta \frac{x_j(n) - x_i(n)}{\left\| x_j(n) - x_i(n) \right\|} \tag{4}$$

where n is the index of iteration; η is the step size; and $\|\cdot\|$ is the norm of the distance vector. Note the update scheme in Eq. (4) may be different for different firefly algorithms.

The luciferin (light intensity) can be updated using

$$\tau_i(n+1) = (1-\rho)\tau_i(n) + \alpha J[x_i(n)] \tag{5}$$

where ρ and α are coefficients for modeling luciferin gradual drop (with time) and the effect of fitness on luciferin; and $J[x_i]$ is the value of fitness function related with the position of the i^{th} firefly.

3. Computer Simulation Results

In this section, computer simulation results using firefly algorithm for both static and dynamic environments are presented and compared with the performance of ACO with various workspace sizes.

Fig. 1 shows a typical workspace for simulation, where the starting location is indicated by a circle (bottom) and the ending location is indicated by another circle (top); and the black areas represents an obstacle.

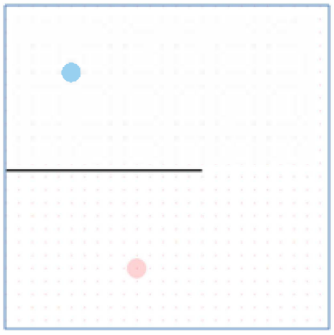

Fig. 1. A workspace with a starting point (bottom), an ending point (top), and an obstacle (black area in the middle)

The simulation starts by placing fireflies (with equal quantity of luciferin) uniformly in the workspace (they can also be placed randomly; but for the purpose of illustration, uniform placement is chosen. This initialization process should not affect the final results dramatically). Note one firefly must be manually placed at the starting location.

After initialization, fireflies move and eventually converge to the optimal solution, as illustrated in Fig. 2 where f_1, f_2, and f_3 represent three fireflies in the workspace; M_2 and M_3 represent the moving directions for f_2 and f_3, respectively. The radius of each circle around a firefly is proportional to its light intensity. Note the light of a specific firefly may be blocked by an obstacle and thus may not be seen by fireflies on the other side of the obstacle.

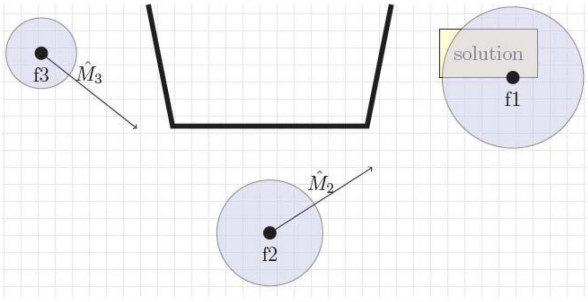

Fig. 2. Fireflies converging on a solution

In our simulation, to demonstrate the movement of fireflies, their trajectories are represented by lines. Fig. 3 shows the firefly trajectories in a workspace after the 10th iteration.

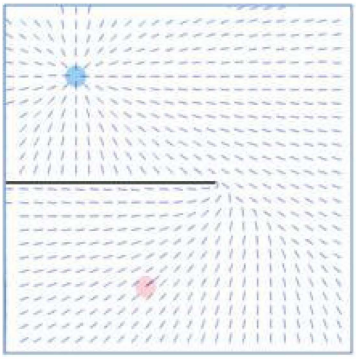

Fig. 3. The movement of fireflies after the 10-th iteration

The final simulation results (for a 500x500 grid workspace) are shown in Fig. 4 and Fig. 5, where Fig. 4 includes the trajectories of all the fireflies while Fig. 5 only shows the solution path found by the algorithm.

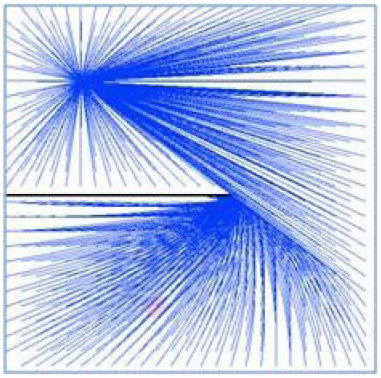

Fig. 4. Fireflies converge towards a solution

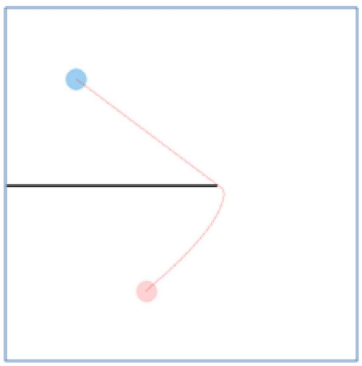

Fig. 5. The solution path found by the firefly algorithm

The effect of the total number of fireflies employed in FA algorithm is investigated in Fig. 6 and Table 1. Fig. 6 illustrates the path solution found by using 10, 26, 122 fireflies in the algorithm, respectively. Using fewer fireflies

(e.g., 10 fireflies) yields a more "rigid" solution path while using more fireflies (e.g., 122 fireflies) yields a "smoother" solution path. Though optimal solution path is found in both cases, the computational cost is significantly different. Table 1 summarizes the path length and the execution time of algorithm of each case. For example, when using 26 fireflies, the path length is 430 while the program execution time is only 14.3 seconds. When using 626 fireflies, the path length is 443 while the program execution time is dramatically increased to 1088.3 seconds. In other words, using more fireflies results in a higher computational cost but the quality of solution may not be improved. However, a "smoother" path may be preferred in some applications such as the operation of a robot manipulator.

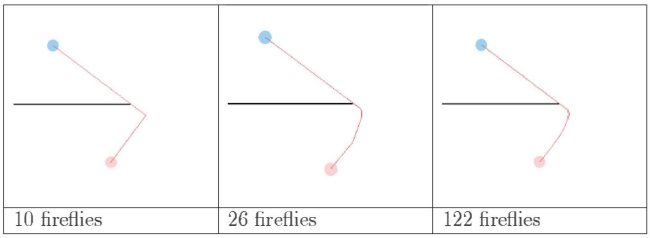

Fig. 6. Solution (path) comparison with different numbers of fireflies in the algorithm

Table 1. Comparison on path length with different numbers of fireflies in the algorithm

Firefly Count	Path Length	Execution time (s)
10	449	14.2
26	430	14.3
65	435	18.0
122	447	49.8
626	443	1088.3

In Fig. 7, an obstacle is added during computer simulation to simulate a "dynamic" environment. There is no obstacle at the beginning of simulation and Fig. 7(a) illustrates that the optimal path in this case is just a straight line. After the 70th iteration, an obstacle (black area in the middle of Fig. 7(b)) is added. The FA algorithm is able to adjust the solution path to avoid the obstacle, as demonstrated in Fig. 7(b) after 173 iterations.

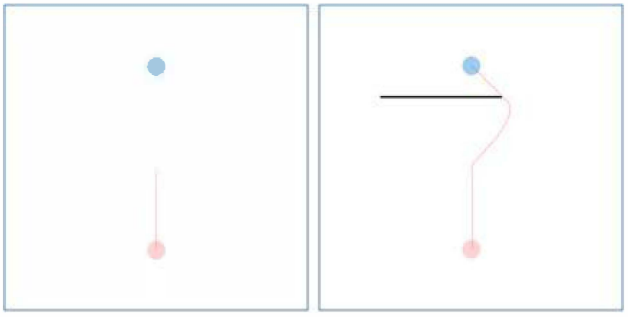

Fig. 7. Path planning in a dynamic environment(a) No obstacle when simulation starts(b) Obstacle added after the 70th iteration

The performance of FA algorithm is compared with the ant colony algorithm (ACO) and the results are shown in Fig. 8 and Fig. 9. Fig. 8 compares the path length found by the FA and ACO algorithm. The lengths of solution paths are almost identical for the two algorithms when the workspace size is small; however, the FA algorithm yields a slightly better results when the size of workspace increases.

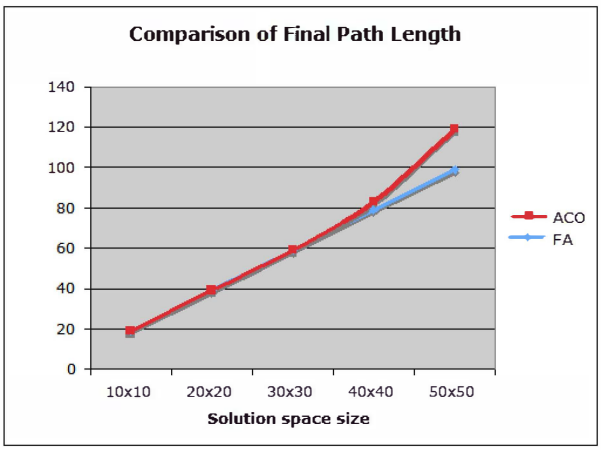

Fig. 8. Comparison of the path length vs. grid size(firefly algorithm and ant colony algorithm)

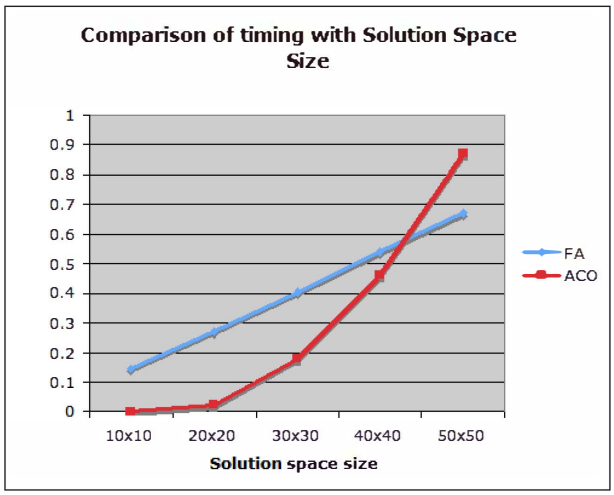

Fig. 9. Comparison of computational time vs. grid size(firefly algorithm and ant colony algorithm)

The comparison of program execution time of the two algorithms is shown in Fig. 9. The FA algorithm maintains a near-linear relationship as workspace size increases, which is very beneficial for solving more complicated path planning problem in a larger grid. On the other hand, the ACO algorithm is obviously more suitable for path planning in a smaller workspace with low computational cost.

4. Conclusions

The firefly algorithm (FA) is one of the newly developed swarm intelligence algorithms. In this research, this approach is applied to find the shortest and collision free path in a two dimensional workspace for robot path planning, in both static and dynamic environment. Computer simulation results demonstrate that this method outperforms the ant colony algorithm (ACO) for a larger workspace, in terms of both path length and computational cost. Future works may include further investigation on the values of the various parameters of this algorithm, such as the minimum number of fireflies required for real-time implementation.

References

[1] K. Krishnanand, P. Amruth, M. Guruprasad, S. Bidargaddi, D. Ghose, "Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources," Proceedings IEEE International Conference on Robotics and Automation, pp.958-963, 2006

[2] K. Krishnanand, D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," IEEE Swarm Intelligence Symposium, pp. 84–91, 2005

[3] X. Yang, "Firefly algorithms for multimodal optimization," in: Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Sciences, Vol. 5792, pp. 169-178, 2009

[4] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004

[5] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999

[6] J. Latombe, "Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts," Int. J. of Robotics Research, Vol. 18 (11), pp. 1119-1128, November 1999

[7] X. Yang, "Firefly Algorithm, Stochastic Test Functions and Design Optimisation," Int. J. Bio-Inspired Computation, Vol. 2 (2), pp.78–84, 2010

[8] T. Apostolopoulos, A. Vlachos, "Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem", International Journal of Combinatorics, doi:10.1155/2011/523806, 2011

[9] K. Swarnkar, "Economic Load Dispatch Problem with Reduce Power Losses using Firefly Algorithm", Journal of Advanced Computer Science and Technology, Vol. 1 (2), pp. 42-56, 2012

[10] C. Liu, Z. Gao, W. Zhao, "A New Path Planning Method Based on Firefly Algorithm", International Joint Conference on Computational Sciences and Optimization, 2012

[11] Sanjay Sarma O V, Vishwanath Lohit T, Deepak Jayaraj, "Path Planning in Swarm Robots using Particle Swarm Optimisation on Potential Fields", International Journal of Computer Applications, Vol. 60 (13), pp. 13-20, 2012

[12] J. Zhou, G. Dai, D. He, et al., "Swarm Intelligence: Ant-based Robot Path Planning", International Conference on Information Assurance and Security, pp. 459 – 463, 2009