21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Zombie Survival Optimization: A Swarm Intelligence Algorithm Inspired By Zombie Foraging

Hoang Thanh Nguyen and Bir Bhanu
Center for Research in Intelligent Systems, University of California, Riverside
nthoang@cs.ucr.edu *bhanu@cris.ucr.edu*

Abstract

Search optimization algorithms have the challenge of balancing between exploration of the search space (e.g., map locations, image pixels) and exploitation of learned information (e.g., prior knowledge, regions of high fitness). To address this challenge, we present a very basic framework which we call Zombie Survival Optimization (ZSO), a novel swarm intelligence approach modeled after the foraging behavior of zombies. Zombies (exploration agents) search in a space where the underlying fitness is modeled as a hypothetical airborne antidote which cures a zombie's aliments and turns them back into humans (who attempt to survive by exploiting the search space). Such an optimization algorithm is useful for search, such as searching an image for a pedestrian. Experiments on the CAVIAR dataset suggest improved efficiency over Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO). A C++ implementation is available.

1 Introduction

Swarm intelligence is a family of decentralized stochastic algorithms inspired by the behavior of swarms. Since standard particle filter implementations are outperformed by evolutionary computation methods such as Particle Swarm Optimization (PSO) [6], we set out to develop a framework which outperforms PSO and present a novel algorithm inspired by the foraging behavior of zombies.

The scenario is the following. A virus has broken out and a number of N zombies or agents occupy a search space. An airborne antidote has been dispersed into the search space in which the zombies randomly walk (exploration mode). The concentration of the antidote at any given location corresponds to the fitness function of that location (i.e., high fitness = high antidote concentration). Sufficient levels of antidote are able to "cure" any zombies who happen to breathe it in, turning the zombie agent into a human agent (exploitation mode). Other zombies sense this human and attempt to catch them (hunter mode). Since high concentrations of antidote in turn can cure the chasing zombies, the human attempts to survive by exploiting the local fitness space, searching for the position of highest fitness which best barricades them from the impending zombies. Zombies which are able to successfully reach the human agent acquire sustenance and the resulting bite turns the human agent back into a zombie.

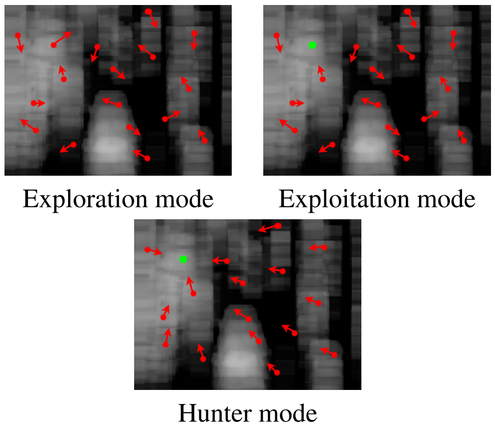

Figure 1: Agents automatically switch between 3 modes: random walking (search space exploration), survival mode (search space exploitation), and hunter mode (swarming).

2 Related work

Tracking is traditionally performed using particle filter. Swarm intelligence algorithms include Particle Swarm Optimization (PSO) [3], Bacterial Foraging Optimization (BFO) [4], and Ant Colony Optimization (ACO). The ultimate goal of any of these optimization

Symbol	Definition
N	Number of zombies/agents
S	Speed of zombies, e.g., step size in pixels
V	Variance of zombie walk direction
A	Fitness threshold which "cures" a zombie
G	Max generations for stop condition
T	Target fitness for stop condition

Table 1: Definitions of symbols used in the Zombie Survival Optimization algorithm.

algorithms is to find the global best fitness as efficiently as possible. Since fitness/objective function calls are often the most resource-intensive component of an optimization algorithm, efficiency is often defined as using the fewest number of fitness function calls as possible, i.e., fastest convergence to the global optimum.

Unique to ZSO is the automatic balance between exploration and exploitation, which is traditionally done via explicit parameters (e.g., PSO, BFO) or in a fixed manner (e.g., 90% exploration, 80% exploration).

3 Technical approach

Agents are initialized in the search space with a uniformly random location and direction. There are three states or modes in which agents can be in:

1. **Exploration mode:** random search
2. **Hunter mode:** actively hunting a human
3. **Human mode:** local intelligent search

Agents are initialized in the first mode which attempts to explore the search space. Table 1 defines all symbols used in the algorithm.

3.1 Zombie exploration mode

While the exploration search function is generic, the default exploration function is defined as follows: at each time step, each zombie moves forward in a step of size S in the current direction with a variance of V. This variance adds noise to the system which aids in climbing out of local optima. Zombies continue in this manner until they either 1) sense a human in which case they start chasing them, 2) stumble upon a position of high fitness and turn into a human, or 3) reach the boundary of the search space in which case they simply change direction.

3.2 Zombie hunter mode

Zombies are attracted to humans and chase after them. At each time step, if there are any humans, zombies will change their direction to face the nearest human and continue to stumble toward them. Stumbling consists of the same variation when walking as in exploration mode (i.e., zombies can't walk in perfect straight lines) with the exception that the general direction is continually reset to face the nearest human.

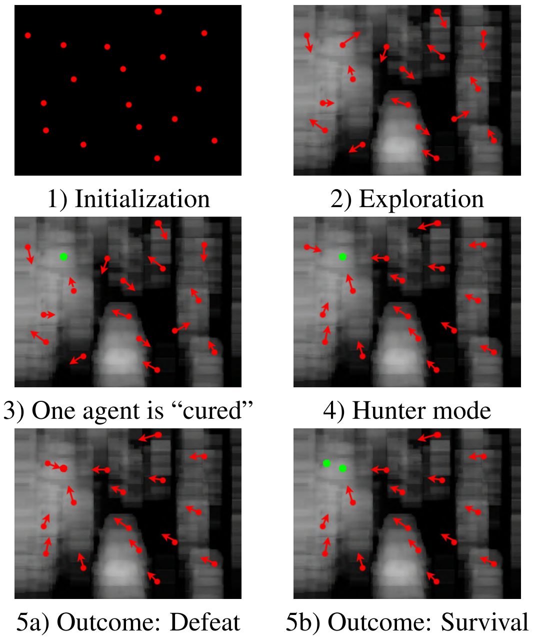

Figure 2: 1-2) Initialize and explore. 3) One or more agents turn into humans (green) due to high antidote concentrations. 4) Zombies pursue humans. 5a-5b) Humans are either defeated or survive.

3.3 Human exploitation mode

Zombies can become humans if the fitness of their current position exceeds threshold A, which can be dynamically defined by the mean of the range R of fitness values so far:

$$A = \frac{max(R) + min(R)}{2}$$

Humans realize that they will be chased and attempt to find a local optimum near their current position. Knowing that this improves the odds of pursuant zombies turning into humans themselves, this offers them the best long-term strategy for survival. Like the exploration function, the exploitation search function is also generic but is defined as a local mean shift search [1].

The algorithm ends when either a target fitness value T is reached or a maximum number of generations G

Algorithm 1 Zombie Survival Optimization (ZSO)

```
1:  $I \leftarrow$ image to search
2:  procedure ZOMBIESURVIVALOPTIMIZATION
3:                                  ▷ Initialize N zombies in search space
4:      for all zombies z do
5:          $z.location \leftarrow$ random point,
6:              e.g., $([0, I_{width}], [0, I_{height}])$
7:          $z.direction \leftarrow$ random direction,
8:              e.g., $[0, 360)$ degrees
9:      end for
10:                                 ▷ Zombies hunt for humans
11:     for $G$ generations do
12:         for all zombie $z$ (asynchronous) do
13:             $z.location \leftarrow z.location +$
14:                 $(z.direction * V * S)$
15:             $f \leftarrow$ evaluate fitness at $z.location$
16:                                 ▷ Search exploitation mode (human)
17:             if $f >$ threshold $A$ then
18:                 $z.is\_human \leftarrow true$
19:                 Gradient ascent search
20:                     of local neighborhood
21:                 if any zombie $z'$ within $S$ reach then
22:                                 ▷ Bitten by zombie
23:                     $z.is\_human \leftarrow false$
24:                 end if
25:             else                ▷ Exploration mode (zombie)
26:                 if any humans exist then
27:                     Find closest human $h$
28:                     $z.direction \leftarrow$ toward $h$
29:                 end if
30:             end if
31:         end for                 ▷ Zombie loop
32:     end for                     ▷ Iteration loop
33:     return Location of best fitness
34: end procedure
```

is reached, whichever comes first. Algorithm 1 details the full pseudo code for the ZSO algorithm. A C++ implementation can be downloaded at `http://www.cs.ucr.edu/~nthoang/zso/`.

4 Experimental results

4.1 Dataset

Experiments are performed on the CAVIAR [2] dataset (see Figure 3). 1,843 search spaces were generated from objects in the first 300 frames of the six recommended [5] corridor videos *EnterExitCrossingPaths1cor*, *OneStopMoveNoEnter1cor*, *ShopAssistant2cor*, *ThreePastShop1cor*, *TwoEnterShop3cor*, and *WalkByShop1cor*. These fitness spaces were generated

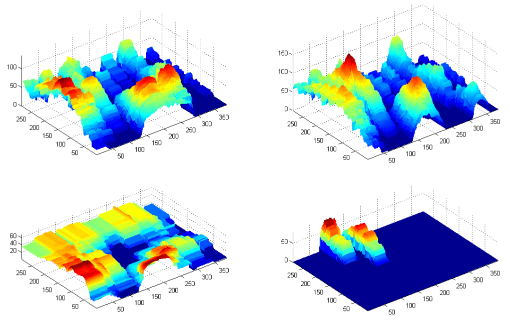

Figure 3: Top: sample frames from the CAVIAR dataset [2]. Bottom: four of the 1,843 fitness space images generated from the dataset using HSV signatures initialized on Viola-Jones pedestrian detections. Resolution is 384$\times$288 with higher (red) locations representing high fitness and lower locations (blue) representing low fitness.

by automatically detecting pedestrians using a Viola-Jones head and shoulders detector and tracking the initialized signature on all the frames using the exhaustive search. These fitness spaces range in difficulty from simple to complex (see Figure 3).

4.2 Metrics and effect of parameters

Number of fitness evaluations is the number of function calls made to the fitness function (the lower, the better). Given a number of evaluations, the **best average fitness** is defined as the average of the best fitness of all 1,843 search spaces (the higher, the better).

Figures 4 and 5 demonstrate the effects of the parameters on algorithm performance.

4.3 Results

Figure 6 shows the performance of ZSO (100 agents, 100 generations, 25px step, 25 deg variance, 0.50 threshold) in comparison with standard PSO (50 agents, 200 generations) and BFO (10 agents, 500 reproductions, 10 swims, 5px step, 90% disperse) algorithms as

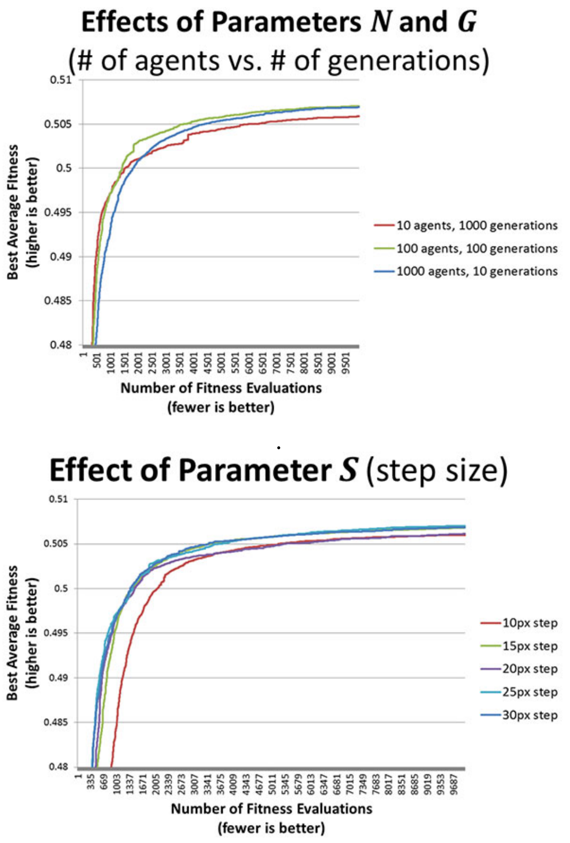

Figure 4: Effects of params N, G, and S.

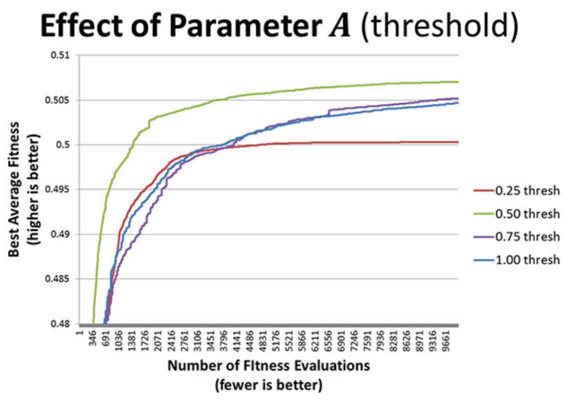

Figure 5: Effects of parameter A.

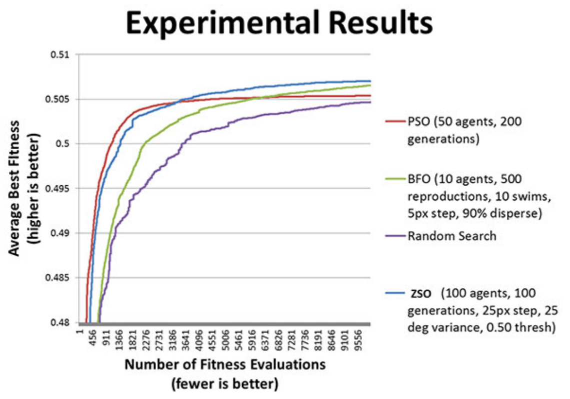

Figure 6: Experimental results comparing ZSO with PSO and BFO.

well as with random search. Parameters were individually optimized for each algorithm using a random subset and fixed for all remaining tests. As the number of fitness evaluations increases, ZSO achieves higher performance over PSO and BFO, suggesting that ZSO may be better suited for finding the global optimum.

5 Conclusions

A new swarm intelligence algorithm modeled after the behavior of zombies vs. humans is proposed to address the challenge of balancing between exploration and exploitation for search optimization. Experiments on real-world multi-person tracking data show that it can be more efficient than BFO as well as PSO which in turn outperforms traditional particle filter [6].

Acknowledgements

This work was supported in part by NSF grants 0727129 and 0905671 and ONR grant N00014-09-C-0388.

References

[1] M. Deilamani and R. Asli. Moving object tracking based on mean shift algorithm and features fusion. In *IEEE AISP*, pages 48 –53, June 2011.

[2] R. B. Fisher. The PETS04 surveillance ground-truth data sets. In *IEEE PETS*, pages 1–5, 2004.

[3] J. Kennedy and R. Eberhart. Particle swarm optimization. *ICNN*, 4:1942–1948 vol.4, 1995.

[4] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE CSM*, Vol. 22, No. 3:52–67, 2002.

[5] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *ECCV*, pages 605–619, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. Sequential particle swarm optimization for visual tracking. In *IEEE CVPR*, pages 1 –8, June 2008.