Combination of Particle Swarm Optimization with Stigmergy and Evolution

Y. Luna Lin, Angeline Rao, Jazz Zhao *

November 28, 2018

1 Introduction

Particle swarm optimization (PSO) was first proposed in 1995 [1] and it has been shown that some of its basic variants can be as effective as, if not more than, standard optimization methods and those based on evolutionary algorithms (EA). In this project we aim to further explore modifications in the PSO framework, by incorporating stigmergy and subjecting the swarm to artificial evolution.

PSO deploys a swarm of simple particles over the search space. In the original formulation, a particle has its current position, velocity, memory of its personal best solution, and knowledge of the swarm's best solution. Each particle updates its velocity based on its personal best and the swarm's collective best solution. This is analogous to having attractive potentials centered at the personal best and the global best, both of which exert forces on the particles and change their trajectory. Inspired by the module on social insects and stigmergy, we suspect that leaving traces in the environment can improve PSO performance, by enriching the particles' collective memory. Since the search space is continuous and possibly has very high dimensions, we plan to discretize the search space into a coarser grid and allow particles to deposit pheromone in their current grid cell. As in standard ant colony optimization methods, the amount of pheromone deposited will represent how good that solution is. A particle flying over the search space can either use the value of its current cell, or interpolate values from cells in a small neighborhood around it.

The other aspect we will explore is the application of EA, and if our stigmergic PSO works, we will subject our algorithm to evolutionary pressure. The ultimate goal of combining PSO and EA is to create an automated optimization framework that will adapt to find the best solution, regardless of the type of problem. In our project, we likely will not be building a general purpose optimization software, but we aim to explore some aspects of PSO-EA combination.

There are many examples of applying EA to PSO, or simply combining the two approaches to complement each other. For instance, in [3], a fraction of the particles is randomly chosen to reproduce and only recombination is used. A natural extension is to regulate reproduction

*Listed in alphabetical order.

using the particles' fitness, and incorporate mutations. With the addition of stigmergy, we have a few more degrees of freedom to apply EA on, for example the parameters in the functions that represent how good a particle perceives its current solution to be, and how it responds to the pheromone in the environment.

Another existing example is using evolutionary strategies to evolve hyperparameters that control the randomness in the system. It has been shown ([2],[5]) that the distribution of the random numbers used in the velocity update plays a role in PSO performance. A Lévy distribution with fatter tails than a Gaussian can hasten the convergence. A possible extension is to explore if this performance boost still holds true for our stigmergic PSO approach, as well as the effects of other types of distributions.

For simplicity, we will constrain the problems that we are solving to 2D problems. To judge the performance of our modified PSO algorithm, we will evaluate how it does against other benchmarks on test problems. We will start out with the ellipsoidal function as a toy example. If that goes well, we will try the much more challenging Rosenbrock function. If our PSO is capable of optimizing the "banana" function, we will tackle some more challenging multimodal problems. For each test problem, we will evaluate the performance by assessing if a solution is found, its accuracy and uncertainty, the number of convergent particles, and the time it takes, measured in function evaluations, computational time, and generations if evolution is involved. If we have time, we will consider metrics of success other than convergence, time, accuracy and uncertainty, e.g. adaptability to dynamic problems and robustness in the face of adversary.

If everything goes smoothly in our project, a potential extension that we could pursue is studying how our algorithm behaves when we change the problem space. As we saw in class, there are cases in stigmergic algorithms where a new best solution becomes available and we want to see whether the particles can find this new best solution given the pheromone that has already been deposited in the various grid spaces. We could experiment with changing our fitness function or adding additional probabilistic elements to the algorithm so that it is more open to finding a new best solution.

2 Background and Related Work

In the two decades since its conception, PSO has undergone many iterations and modifications. The basic algorithm uses a collection of simple agents, each with three D-dimensional vectors, where D is the spatial dimension. These vectors represent the current position in the solution space, $\vec{x}$, the agent's best solution in the past, $\vec{p}_{best}$, and the agent's current velocities, $\vec{v}$.

Each particle is subject to an acceleration, which is a function of particles internal variables and information it knows about the swarm,

$$\vec{v}(t+1) = \vec{v}(t) + \vec{a}(t) \tag{1}$$
$$\vec{a}(t) = F(\vec{x}, \vec{v}, \vec{p}_{best}, \vec{g}_{best}) \tag{2}$$

The velocities are used in updating the particle's position,

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1) \tag{3}$$

The basic algorithm uses the following acceleration rule

$$\vec{a}(t) = \vec{U}(0, \phi_1) \otimes \vec{(}p_{best} - \vec{x}) + \vec{U}(0, \phi_2) \otimes (\vec{g}_{best} - \vec{x}) \tag{4}$$

where $\vec{U}(0, \phi_\alpha)$ is a vector with components drawn from a uniform distribution from 0 to ϕ_α; ϕ_1 and ϕ_2 tune the strength of attraction of $\vec{p}_{best}$ and $\vec{g}_{best}$. This update rule can result in pathological cases where the velocities diverge. Methods such as inertial weights, constriction coefficient, and maximum velocity constraints have been proposed to address this issue. For example, inertial weights modify Eq. 1 to be [7]

$$\begin{aligned} \vec{v}(t+1) = \omega\vec{v}(t) + a(t) \implies \\ \Delta\vec{v} = a(t) - (1-\omega)\vec{v}(t) \end{aligned} \tag{5}$$

The physical intuition of the factor $1-\omega$ is the friction coefficient of the medium the swarm occupies.

PSO has also been subjected to genetic programming (GA). A selected examples are summarized below.

In [3], particles in the swarm are randomly chosen to reproduce and combine their states probabilistically. The authors also investigated the effect of separating the swarm into sub-populations.

In [6], the authors notice that GA and PSO improves at different times of the optimization process, and therefore combine the two method by first applying one, and once the performance levels off, applying the other.

The most general approach of combining PSO with GA is presented in [4], where GA is used to evolve the function form and parameters of F in Eq. 2.

3 Plan of Execution

A timeline of milestones and corresponding labor division is provided.

1. Week 1, 11/12 - 11/19:
    - Research current PSO frameworks and implementations and create a shortlist of suitable packages that we can expand by adding a stigmergy component and evolutionary control, if applicable. (Angie)
    - Explore and evaluate different methods of adding an evolutionary control component to existing swarm optimization algorithms with focus on theoretical guarantees. Come up with a shortlist (e.g. genetic programming) of proposed modifications of the PSO controller. (Jazz)
    - Implement basic PSO algorithm in python, in case the open source codes are not amenable to customization, this will serve as the basis that we will add stigmergy and evolution on. (Luna)
2. Week 2, 11/19 - 11/26:

- Modify either our own or the most promising PSO implementation on the shortlist to include the stigmergy component. (Angie, Luna)
- Carry out performance test of basic PSO on ellipsoidal and Rosenbrock functions. We will determine at this point whether we are at a good position to try multimodal problems or stick to unimodal tests. (Luna, Angie)
- Parameter sweep test in the Hill equation that tunes stigmergic responses, test stigmergic PSO performances. (Luna)
- Execute a parameter search with focus on how the resolution of our search space, i.e. grid size, affects our various evaluation metrics. (Angie)
- Determine the exact types of evolutionary algorithm to be implemented, as chosen from the shortlist, as well as the implementation strategy. Ideally start the implementation. (Jazz)
- Develop appropriate fitness function to include rate of convergence, global vs. local optimum, (Jazz, Luna)
- If we have enough time, consider choosing and implementing benchmark algorithms for comparison, e.g. stochastic gradient descent, simulated annealing, etc. (Angie)

3. Week 3, 11/26 - 12/3:

   - Present a working implementation of evolutionary (with or without stigmergy) PSO, whether that is our own in-house implementation or leveraging an existing PSO framework or package (All)
   - Explore different fitness functions and objectives for evolutionary algorithms with respect to their effects on proposed optimal solutions and their convergence. (Jazz)
   - Finalize the evaluation metrics for our algorithm, as well as the problems we will evaluate on and algorithms to compare to. (All)
   - Collect and analyze preliminary results for the in-class project presentation. Prepare the presentation. (All)

4. Week 4, 12/3 - 12/10: Wrap-Up[1] (All)

   - Ensure our algorithm is working as expected. Debug our implementation if necessary.
   - Collect final results of our algorithm on chosen evaluation problems, and analyze them under all metrics of evaluation.
   - Compare our results to those from benchmark algorithms, derive an overall evaluation.

[1]Future plans: If the results of our algorithm are competitive and look promising, we would like to continue refining our methodology with the goal of a publication. Otherwise, we plan to stop working on this project with the submission of the final writeup.

References

[1] Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural netowrks IV*, pp. 1942-1948.

[2] Kennedy, J. (2004). Probability and dynamics in the particle swarm. In *IEEE congress on evolutionary computation (CEC04)*, pp. 340-347.

[3] Løvbjerg, M., Rasmussen, T. K., and Krink, T. (2001). Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the third genetic and evolutionary computation conference (GECCO)*, pp. 469-476.

[4] Poli, R., Langdon, W. B., and Holland, O. (2005). Extending particle swarm optimization via genetic pro- gramming. In M. Keijzer et al. (Eds.), *Lecture notes in computer science: Vol. 3447. Proceedings of the 8th European conference on genetic programming*, pp. 291300.

[5] Richer, T. and Blackwell, T. M. (2006). The Lévy particle swarm. In *Proceedings of IEEE congress on evolutionary computation*, pp. 3150-3157.

[6] Robinson, J., Sinton, S., and Rahmat-Samii, Y. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Proceedings IEEE international symposium on antennas and propagation*, pp. 314317.

[7] Shi, Y. and Eberhart, R. C. (1998). A modified paricle swarm optimizer. In *Proceedings of the IEEE international conference on evolutionary computation*, pp. 69-73.