5-2007

A Multi-Agent Based Approach for Particle Swarm Optimization

Raheel Ahmad
Southern Illinois University Carbondale

Yung-Chuan Lee
Southern Illinois University Carbondale

Shahram Rahimi
Southern Illinois University Carbondale, rahimi@cs.siu.edu

Bidyut Gupta
Southern Illinois University Carbondale

A Multi-Agent Based Approach for Particle Swarm Optimization

Raheel Ahmad[1], Yung-Chuan Lee, Shahram Rahimi, Bidyut Gupta
Department of Computer Science,
Southern Illinois University,
Carbondale, IL, USA
{rahmad, ylee, rahimi, bidyut}@cs.siu.edu
[1] Phone no.: 618-453-6025
Fax no.: 618-453-6044

Abstract. *We propose a new approach towards Particle Swarm Optimization named Agent-based PSO. The swarm is elevated to the status of a multi-agent system by giving the particles more autonomy, an asynchronous execution, and superior learning capabilities. The problem space is modeled as an environment which forms clusters of points that are known to be non-optimal and this transforms the environment into a more dynamic and informative resource.*

1. Introduction

Particle Swarm Optimization (PSO) is a stochastic optimization technique, inspired by the idea of a flock of birds moving towards a final goal through cooperation as well as independent exploration. However, the particles in the swarm have limited autonomy and intelligence. The velocity of the particle in the problem space is governed by a central algorithm that is the same for every particle and the algorithm gives the particle limited flexibility and knowledge about the environment. Although, this makes every iteration computationally efficient, the overall performance can be improved by adding some sophistication to the algorithm. We argue that elevating the particle to the status of an agent will make the whole algorithm perform much more effectively, especially in a large problem space with a complex structure. In this paper we present a new approach for implementing a particle swarm for optimization by viewing the swarm as a multi-agent system.

2. Background & Motivation

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart [4], as a stochastic, population based optimization technique for efficiently finding an optimal or near-optimal solution to a numerical problem. It involves a "swarm" of "particles" that move through the problem space, where the location of a particle represents a possible solution. The particles adjust their velocity based on their own best location and that of the neighboring particles in search for the best location which minimizes a fitness function. It has proven to be a very efficient and quick technique for finding solutions to several optimization problems [2], [5]. However, the simplicity of the PSO does not provide any features for particles to effectively explore the problem space such as preventing them from analyzing points in the problem space that some particle may have already visited earlier and found it to be non-optimal.

A Multi-Agent System (MAS) is composed of a collection of autonomous software agents which are capable of completing desired goals cooperatively. The basic attributes of an agent that are considered typical are autonomy, learning and cooperation [7]. These properties imply that agents are capable of executing independently from any other control and possibly asynchronously, discover relevant knowledge from the the environment and other agents that may help in attaining the desired goals, and work cooperatively and competitively with other agents.

PSO and MAS are similar to each other in the first glance. They both are population-based approach and accomplish tasks cooperatively. However, agents are quite distinct from particles in a very specific manner and we discuss three reasons that are relevant to our approach. First of all, a particle is not usually considered *autonomous* as it can only move in the problem space according to the main algorithm, but agent are able to explore the environment with much more flexibility. Second, particles are purposely defined as less intelligent to gain computational performance by reducing their capability. On the other hand, learning, a form of intelligence, is one of the main component required by agents. Finally, PSO enforces synchronized execution in order to maintain simplicity in design. However, because of the autonomy, learning and cooperation components, agents in MAS are naturally asynchronously executed. Consequently, an approach that combines the simplicity of the PSO with the autonomy and learning of the MAS can benefit from both.

By introducing autonomy and learning to the PSO, particles become more intelligent and autonomous and can achieve more effective performance and use of resource. Since each particle is now able to take advantage of its surrounding environment, it can, for example, determine whether a costly fitness function evaluation is needed at all for each point it visits. If not, it can request information from its neighbors and update its location. In turn, this saves computational resources and can accelerate the performance.

Because of the introduced complexity of autonomy and learning in the particle, the performance of PSO might decrease to a certain degree. However, since each particle only performs the costly fitness function when needed, the modified PSO is in fact more effective and can easily outperform the original PSO in optimization scenarios where the problem space is non-trivial. In the rest of the paper, we use the terms particle and agent interchangeably.

Since PSO and MAS are normally classified as two different fields, optimization algorithm and artifact system respectively, researchers have traditionally focused on applying PSO as an optimizing algorithm in MAS and other methodologies. To our knowledge, there is only one related work which attempts at integrating PSO and MAS to form a MAS-based PSO approach, called MAPSO [8]. The only similarity between our modified PSO and MAPSO is that the particles have been enhanced as agents. The definitions of environment, the capability of particles and the approach mechanism are quite different, and we believe that our approach attempts at combining the two ideas at a much more fundamental level. Also, since a particle in the MAPSO approach utilizes another swarm at its own level to speed up the optimization performance, it requires large amount of resource to carry out optimization process and thus limits its practicality.

3. APPROACH OUTLINES

By integrating autonomy and learning into the original PSO, we propose a modified PSO approach, named Agent-based PSO (APSO), where each particle is now termed as an agent. Also the environment is modeled itself as an agent and is responsible for providing extra information about the problem space to the agents. We further attach a boolean value with each point in the problem space to identify whether that point has been already visited by an agent. Obviously if the point has been visited, then it must not be a solution, otherwise it would have resulted in the termination of the algorithm. As in the original PSO, particles search for the optimal solution by traveling through the problem space in a multi-dimensional environment. Hence, if we can limited the number of unvisited points that a particle can possibly travel to in the future, this can increase the efficiency in a large problem space.

Based on this methodology, the environment is now transformed from a static one to a dynamic version where the points in the problem space are being continuously tagged by particles after they visit them for the first time. To further decrease the chance that particles visit the tagged points, the environment agent applies a density-based cluster algorithm to discover clusters of tagged points. Those clusters can then be utilized by an agent to identify areas in the problem space that it can possibly stay away from. The actual technique of clustering is described in the next section.

The simple particles in PSO are now transformed to agents in APSO with autonomy and learning capabilities. The autonomy property enables particle agents to be more proactive by allowing them to actively change the environment; for e.g., by changing the status of non-optimal points in the environment. Instead of the comprehensive learning concept in the traditional MAS in which agents adapt the changes in the environment to maximize its goal achieving ability, the learning attribute added to the particle agents in APSO is limited to only be more aware of the changes in the environment. An agent can request cluster information from the environment agent, and can use this to considerably limit the performance overhead and still retain the simplicity of the original PSO.

Since particle agents are now able to observe if a point is visited, there is no need to enforce all particle agents to perform fitness function evaluation as in PSO. Instead, if a particle agent arrives at a visited point which means this points has been evaluated before and not found optimal, the agent can skip the fitness evaluation and request its neighbors' positions and cluster information to update its own position. This optimizing mechanism facilitates asynchronous execution in APSO which can improve the overall optimization performance as well as extend APSO to be deployed in a heterogeneous environments. More information regarding this mechanism is discussed in the next.

Besides tagging points in problem space, each particle agent is also given other abilities as well. These include inquiring the current best solution from its neighbors to calculate the global best location, return its personal best solution to its neighbors, and to request information regarding the surrounding clusters from the environment agent. These extra *methods* also extend the particle's functionality beyond a simple agent.

4. DETAILED METHODOLOGY

In this section, we define the essential elements inthe proposed APSO and formulate the updating function to utilize the introduced cluster property. We then describe the functionalities of the environment agent and how it classifies clusters in the environment.

Definition 1: The environment in APSO of n dimension problem space, denoted by E, is defined as

$$E = \{P, C, n, FF, \epsilon, \epsilon_d\}$$

P is the set of points in the problem space where $p \in P, p = \{d_1 \times d_2 \times \cdots \times d_n, tag\}$ is a point that includes the n dimensional position and an extra boolean value called *tag*.

C represents the set of clusters in E and is defined by

$$C = \{(p_1, p_2, \cdots, p_{center}, \cdots, p_m), d \mid d \geq \epsilon d\}$$

FF is the fitness function and is defined within the environment because different criteria requires different fitness functions. ϵ is the minimum error criteria that is used to terminate the APSO execution. A predefined maximum iteration is employed to terminate the system when the swarm cannot find any solution to fulfill ϵ criteria. ϵ_d is the minimum density requirement for the environment agent to identify if a group of tagged, non-optimal points can be considered as a cluster in the problem space.

As we mentioned in the previous section, in APSO each point, p, contains one extra field, tag, to indicate if the point has been visited. The environment agent then use this information to discover clusters. Each cluster, C, contains a center point, p_{center}, and the current density of the cluster, d. Those two entries can be requested by a particle agent to compute the next position in its updating function.

Definition 2: The particle agent in APSO, denoted by PA, is defined by $PA : \{pB, gB, N\}$. N contains a set of particle agents which are predetermined as neighbors of PA. PA is defined with following functions:

- requesting each neighbor's current personal best location.
- returning its current personal best to neighbors.
- obtaining center location of each surrounding cluster.
- observing if the current position is visited.
- tagging current position if not optimal.

Each particle agent receives a set of its neighbors during the system initiation. Different topologies can be used to create different property of neighbors of a particle agent to obtain different performance. For simplicity, we assume that each particle agent receives a static set of neighbors.

A particle agent maintains its current best solution, *pB*, which is the point in the problem space that it has found to be closest to an optimal solution, and the best solution among its neighbors, *gB*. In each iteration, a particle agent observes the condition of its current position to determine if it needs to perform fitness evaluation and it checks if the termination criteria is fulfilled. If not, it updates its *pB*, requests its neighbors' *pB* to compute the *gB*, and tags the current position as *visited*. A detailed description of this mechanism is described later in this section.

Definition 3: The original updating function in PSO is enhanced in APSO to take advantage of the new cluster concept and is defined by the followings:

$$\begin{aligned} V_{d+1}^m = V_d^m &+ R_{pBest}\left(pBest^m - X_d^m\right) \\ &+ R_{gBest}\left(gBest^m - X_d^m\right) \\ &- R_c \Sigma_{i=1}^{N_c}\left(C_{i_{center}}^m - X_d^m\right) \\ X_{d+1}^m = X_d^m &+ V_{d+1}^m \end{aligned}$$

Here V_{d+1}^m is the velocity of particle agent m in the iteration $d+1$ and is used to compute particle agent's next position. The idea is to move toward the better solution area in the problem space and move away from the clusters at the same time. Furthermore, R_{pBest}, R_{gBest} and R_c are random numbers to retain the stochastic feature of the original PSO algorithm.

To illustrate the importance of constructing clusters of visited points and the improvement this will provide, let us consider an example as shown in Figure 1. The cloud-like areas indicate dense visited points in (a) for the original PSO and discovered clusters in (b) for the the APSO. We further assume $V_d = 0$ and $R_{pBest} = R_{gBest} = R_c = 1$ to simplify the example but still maintain the correctness. In Figure 1(a), a particle updates its next position solely depending on *pB* and *gB*, the particle will move to a position in the problem space that is surrounded by a large number of non-optimal points, and therefore it is highly unlikely that the new point itself will provide a final solution. Since the particle cannot observe the condition of the new position x_{d+1}, it computes the fitness function to determine if x_{d+1} is optimal. From the figure, we know that x_{d+1} is either visited by other particles or most likely not optimal as lots of its surrounding points are not optimal.

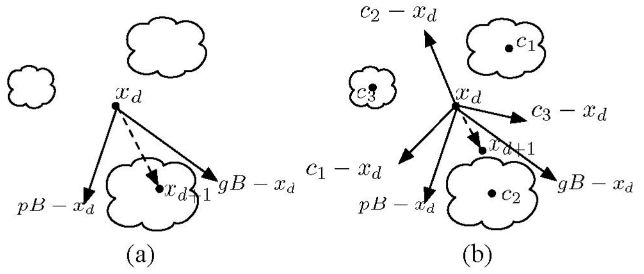

Figure 1 (a) original PSO (b) APSO

On the other hand, considering (b) in Figure 1, each particle agent is aware of the condition of its position and the information of surrounding clusters. Now, by taking the the clusters into account, the new position x_{d+1} has been calculated by the new velocity update function as away from the cluster c_2. The particle agent then determines if x_{d+1} has been visited before evaluating fitness function. This technique not only reduces the chance that a particle agent visits non-optimal points in the problem space but also accelerates the optimization process by preventing unnecessary fitness function evaluation.

Optimization Procedure

After the APSO system is initialized, each particle agent is assigned a set of particles as its neighbors and a starting position which is normally distributed over the problem space. The system then activates particle agents to search for an optimal solution until either maximum iteration is reached or the minimum error criteria is fulfilled.

Each particle agent first determines whether its current position is visited by checking the tag of the position. If the position is tagged, the agent skips the fitness function evaluation and gathers necessary information to update its position. It requests the *pB* values from its neighbors to calculate its current *gB* and request the location of neighboring clusters from the environment agent. Each particle agent then compute its new velocity value V_{d+1} in order to move to the next position by using the velocity updating function defined in definition 3 in this section.

On the other hand, if the current position is not tagged, the particle agent follows the same procedure as particles in the original PSO. It first evaluates the fitness value of the current position by using the fitness function to check if the minimum error criteria ϵ is fulfilled. If ϵ is fulfilled, the particle agent notifies the system about the solution, and the system then broadcasts a terminating message to all other particle agents and returns the solution to the user. If ϵ is not fulfilled, the particle agent then check if the maximum allowed iteration is reached. If reached, it notifies the system and terminates itself. Else, it tags the current point as visited and update its position as mentioned earlier. After all the particle agents have terminated and none has reported an optimal solution, the system returns a *no solution found* message to the user. The overall optimization procedure can is shown as a low-chart in Figure 2.

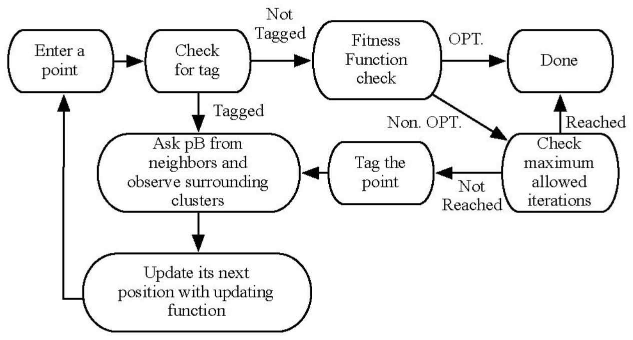
Enter a point
Check for tag
Not Tagged
Fitness Function check
OPT.
Done
Tagged
Non. OPT.
Reached
Ask pB from neighbors and observe surrounding clusters
Tag the point
Not Reached
Check maximum allowed iterations
Update its next position with updating function

Figure 2 - Particle Agent Optimization Procedure

Environment Agent

The main responsibilities of the environment agent are the discovery of clusters of visited, non-optimal points with a pre-defined density constraint and providing the characteristics of discovered clusters such as their density and the center point to particle agents. Although this clustering process is managed in parallel of the other particle agents by the environment agent, the clustering algorithm needs to be efficient as it will effect the overall performance of the swarm. There are several studies regarding density-based clustering algorithms and each of them performs differently and utilize different input parameters to discover the clusters. We briefly review two common density-based algorithms, CLARANS and DBSCAN, and justify the reasons why we choose DBSCAN.

CLARANS (Clustering Large Applications based on RANdomized Search) proposed by Ng and Han is based on randomized search technique [6]. It utilizes two input parameters, the maximum number of neighbors examined and the number of local minima obtained, to control the quality of discovered clusters which also affects the performance. On the other hand, Ester, Kriegel, Sander, and Xu propose DBSCAN (Density Based Spatial Clustering of Applications with Noise) to classify clusters by checking the number of points in a given radius [3]. Therefore, even though authors indicate that DBSCAN only requires one input parameter, users can also supply two parameters: maximum radius of the neighborhood and minimum points in such neighborhood.

We have chosen DBSCAN mainly because DBSCAN outperforms CLARANS at least 100 times as well as discovers more clusters [3]. This is extremely important as our improvement introduces the extra overhead in recognizing clusters. In order to make and improvement over the original PSO, the overhead has to be minimized. Furthermore, because of monotonous property of our environment e.g. the size of the a cluster can only increases, DBSCAN only needs to check the boundary points in each cluster after it has created the first set of clusters.

5. Conclusion & Future Work

In this paper we have proposed a unique approach towards developing a PSO algorithm. The new approach combines the optimization technique with the feature set of a multi-agent system. The result is a new population-based optimization algorithm where the simple particle is now elevated to the status os an autonomous and intelligent agent, and the cluster itself is now viewed as a multi-agent system. The future research initiative will include a comparison of the optimization performance of APSO with the traditional PSO techniques and analysis of the improvements made by varying the clustering algorithm. We have recently introduced another improvement to PSO by providing a novel approach to making the PSO run in a parallel computing environment [1]. We intend to compose the two proposals into a composite PSO algorithm to provide a superior approach than any other previously implemented.

Reference

[1] Ahmad, R., Lee, Y., Rahimi, S., "A New Approach Towards a Parallel Particle Swarm Optimization

Algorithm," in review, Scalable Computing: Practice and Experiences Journal, special issue of Parallel Evolutionary Algorithms, Dec. 2006.
[2] Correll, N. and Martinoli, A. (2004). Modeling and optimization of a swarm-intelligent inspection system. In Proceedings of the 7th Symposium on Distributed Autonomous Robotic System (DARS), Springer-Verlag, London.

[3] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceeding of 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, pp. 226-231.

[4] Kennedy, J. and Eberhart R. (1995). Particle Swarm Optimization. In Proceeding of the IEEE International Conference on Neural Networks, Perth, Australia, Vol. 4, pp. 1942-1948.

[5] Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. IEEE International Conference on Evolutionary Computation, pp. 303-308.

[6] Ng, R. T. and Han, J. (1994). Efficient and Effective Clustering Methods for Spatial Data Mining, In Proceeding of 20th International Conference on Very Large Data Bases, Santiago, Chile, pp. 144-155.

[7] Nwana, H. S. and Ndumu, D. T. 1997. An Introduction to Agent Technology. In Software Agents and Soft Computing: Towards Enhancing Machine intelligence, Concepts and Applications H. S. Nwana and N. Azarmi, Eds. Lecture Notes In Computer Science, Springer-Verlag, London, Vol. 1198, pp. 3-26.

[8] Zhao, B., Guo, C. X., and Cao, Y. J. (2005). A Multiagent-Based Particle Swarm Optimization Approach for Optimal Reactive Power Dispatch. IEEE transactions on power system. Vol. 20, No. 2, pp.1070-1078