Particle swarm optimization in dynamic environments and its application in MIDDLE

Yang Wang

May 8, 2015

1 Introduction

Particle swarm optimization (PSO) is a population-based stochastic algorithm for optimization which is based on social-psychological principles (Kennedy, 2010). The idea came from social behavior simulation and was first applied to optimization problems by Kennedy and Eberhart in the 1990s (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995). Kennedy is a social psychologist and Eberhart is an electrical engineer. It is surprising at first glance that PSO serves both areas equally well. Quote from their original paper (Kennedy & Eberhart, 1995): "Why is social behavior so ubiquitous in the animal kingdom? Because it optimizes. What is a good way to solve engineering optimization problems? Modeling social behavior." Shi and Eberhart (Shi & Eberhart, 1998) later modified the method by introducing inertia weight as new parameters.

PSO makes almost no assumption on the problems at hand and can deal with various types of optimization problems which are hard to tackle using classic optimization methods. For instance, most of its implementations don't use the gradient of the objective function, therefore no differentiablity is required. However, rather than an "algorithm", PSO is more of a metaheuristic and no convergence to optimal solution is guaranteed.

In PSO, the system is initialized with a population of random solutions (particles) in the parameters hyperspace. At each iteration, particles are assigned a velocity and move around to test new parameter values. Moreover, each particle is assigned a neighborhood topology in which it is linked bidirectionally to the particles nearby.

To be more specific, the most common implementation of PSO defines the particle's behavior in the D-dimensional parameter hyperspace as the following two formulas:

$$\begin{aligned} v_{id}^{t+1} &\leftarrow wv_{id}^{t} + U(0,c_1)(p_{id} - x_{id}^{t}) + U(0,c_2)(p_{gd} - x_{id}^{t}) \\ x_{id}^{t+1} &\leftarrow x_{id}^{t} + v_{id}^{t+1} \end{aligned}$$

where i is the target particle's index, d is the dimension, $\vec{x}_i$ is the particles position, $\vec{v}_i$ is the velocity, $\vec{p}_i$ is the best position found so far by particle i, $\vec{p}_g$ is the best position found so far by the swarm, w, c_1 and c_2 are inertia weight, cognitive learning rate and the social learning rate respectively.U is a uniform random number generator. The standard version of PSO implementation (R package *PSO* and *hydroPSO*) uses $w = 0.72$ and $c_1 = c_2 = 1.19$, following an analysis published by Clerc and Kennedy (Clerc & Kennedy, 2002).

Alternatively, the inertia weight can be replaced by constriction factor to constrain the velocity and accelerate the covergence. The update formula is as below and many research have reported better performance using constriction factor (Eberhart & Shi, 2000).

$$\begin{aligned} v_{id}^{t+1} &\leftarrow \chi(v_{id}^{t} + U(0,c_1)(p_{id} - x_{id}^{t}) + U(0,c_2)(p_{gd} - x_{id}^{t})) \\ x_{id}^{t+1} &\leftarrow x_{id}^{t} + v_{id}^{t+1} \end{aligned}$$

where

$$\chi = \frac{2}{|2 - \rho - \sqrt{\rho^2 - 4\rho}|}, \quad \rho = c_1 + c_2$$

It represents the interactions of a number of individuals, none knowing what the goal is, each knowing its immediate state and its best performance in the past, each presenting its neighbors with its best success-so-far at solving a problem, each functioning as both source and target of influence in the dynamically evolving system. As individuals emulate the successes of their neighbors, the population begins to cluster in optimal regions of a search space, reliably discovering good solutions to difficult problems featuring, for instance, non-linearity, high dimension, deceptive gradients, local optima, etc. (Kennedy, 2010)

Numerous extensions of the basic PSO heuristic are possible and new and sophisticated PSO are continually being introduced in an attempt to improve optimization performance. Also the unique characteristic of PSO compared to other classic optimization algorithms gives it an edge in approaching certain tough questions. In this report, we are going to review some of the state-of-the-art PSO algorithms, especially their applications in dynamic and noisy environments. More specifically, our ultimate goal is to build an efficient optimizer for the revolutionary framework (Maintained Individual Data Distributed Likelihood Estimation, or MIDDLE) proposed by Boker (S. M. Boker et al., 2013) for assisting health science human-subject research with networked devices. The main idea of the MIDDLE paradigm is that data can be privately maintained by participants on their personal devices and never revealed to researchers while statistical models are fit and scientific hypotheses are tested. Participants first download the scientific research "app" to their mobile device, which facilitate the data collection and model testing. Statistical models are optimized by sending a candidate vector of model parameters to each individual data site (cell phone, sensor, etc), where an likelihood function is calculated locally. Only the likelihood function value is returned to the researcher's central optimizer. The optimizer aggregates likelihood function values from responding remote data sites and chooses a new set of parameters. This process is repeated until models sufficiently converge. No personal data is ever seen by the researcher in the whole process.

One of the challenges in building the optimizer is how to deal with the ever-changing sample in the experiment. In every iteration, participants are allowed to opt into and opt out of the experiment at any time. Therefore new information keeps coming in even in the middle of optimization process. It poses difficulty in deriving convergence properties analytically. Therefore heuristic methods like PSO may present better and more robust performance. The general idea is to send different parameter vectors to different agents (participants' wearable devices). In this case, the workload of calculating the function values at each particle in the system can be distributed to each agent. However, as new participants coming in and old participants opt out, the objective function is also dynamically changing and the best position recorded by the swarm may become outdated. Therefore, how to adapt PSO in dynamic and noisy environments becomes a major issue in our MIDDLE framework.

The report is structured as follows: in section 2, we are going to review some PSO algorithms which presents satisfactory performance in dynamic environments; in section 3, a hybrid PSO algorithm which integrates gradient descent method is discussed and tested on benchmark functions ; in section 4, the implementation of PSO combined with steepest descent algorithm will be applied on test problems and simulation results will be shown; section 5 summarizes the report and indicates future direction.

2 PSO in dynamic environments

Many real world optimization problems are dynamic in which global optimum and local optima change over time. Particle swarm optimization has performed well to find and track optima in dynamic environments. If the change to the objective function is relatively small, particles in the swarm can quickly detect the change and converge to the new peak without further adjustment. However, if the movement of the objective function is significant, for example the new peak is far away from the region where most of the particles have converged to, then the particles in the swarm may lose track of the peak and keep circulating in the wrong region. The two key issues to be addressed in dynamic PSO problems are diversity loss and outdated information. Diversity loss is the situation where all particles have converged to a certain region and therefore lost the ability to search the other region for other emerging optimal positions. Outdated information means the information shared by all the particles (personal best position and global best position) suddenly becomes outdated due to the environment change. To tackle these two issues, researchers have proposed several novel approaches which will be briefly reviewed below.

In Carlisle and Dozier's work (Carlisle & Dozier, 2000, 2001), they proposed a method for adapting the particle swarm optimizer for dynamic environments by resetting each particle's personal best position to their current position as the environment constantly changes. Two methods for initiating the reset were examined: periodic resetting, based on the iteration count, and triggered resetting, based on the detected magnitude of the change in the environment. This resetting process differs from a restart, in that the particles, in retaining their current location, have retained the profits from their previous experiences, but are forced to redefine their relationship to the goal at that point.

Their preliminary results suggest that these modifications allow PSO to locate and track the moving optimum in dynamic environments. But the designed experiments have many obvious weaknesses, too. The objective functions is simple and the method cannot handle environments with localized fluctuations. Moreover a good strategy is needed that can account for chaotic rather than linear change. Also whether the resetting procedure is optimal remains skeptical and therefore indicate areas than could be explored in the future.

As an alternative adaptation, Blackwell and Bentley (T. M. Blackwell, Bentley, et al., 2002; T. M. Blackwell, 2003) introduced charged swarms with the aim to maintain diversity throughout the run. Shortly afterwards, a new algorithmic variant, which broadens the implicit atomic analogy of charged swarms to a quantum model was introduced, together with the idea of constructing interacting multi-swarms (T. Blackwell & Branke, 2004).

In charged PSO (CPSO), mutually repelling particles orbit a nucleus of neutral particles. This nucleus is, in fact, a conventional PSO swarm. The idea is that the charged sub-swarm maintains population diversity, at least within the spatial extent of the charged orbits, so that function change can be quickly detected, and the swarm can adapt. Meanwhile the neutral swarm can continue to exploit the neighborhood of the optimum in increasing detail. The update rule for the neutral particles are the same as generic PSO, whereas the rule for charged particles is as follows.

$$\begin{aligned} v_{id}^{t+1} &= wv_{id}^{t} + U(0,c_1)(p_{id} - x_{id}^{t}) + U(0,c_2)(p_{gd} - x_{id}^{t}) + a_{id} \\ \vec{a}_i &= \sum_{j \neq i} \frac{Q_i Q_j}{r_{ij}^3} \vec{r}_{ij}, \qquad p_{core} < r_{ij} < p \end{aligned}$$

where $\vec{r}_{ij} = \vec{x}_i - \vec{x}_j$ and each particle has a charge of magnitude Q_i. However, CPSO suffers from $O(N^2)$ complexity, arising from the Coulomb repulsion. Therefore a quantum model is proposed to avoid the computational complexity while maintaining diversity. The orbiting electrons are replaced by a "quantum cloud", which is actually a probability distribution governing where the electron will be found upon measurement. The quantum particles are randomized within a ball of radius r_{cloud} centered on the swarm attractor. The velocity of the quantum particle becomes an irrelevant concept and they are not repelled from other quantum particles or attracted to any attractor. Their role is to provide the information of the good position they found to the whole sub-swarm.

To model the relations and interactions between sub-swarms, they proposed two forms of swarm interaction : exclusion and anti-convergence (T. Blackwell & Branke, 2006). If two or more sub-swarms cluster around a single peak, then only the sub-swarm with best function value at its swarm attractor will remain, and the other sub-swarms will be excluded and reinitialized in the search space. Two swarms are considered to be competing for the same peak when the distance between their swarm attractor are within an exclusion radius r_{excl}. Also, when the number of peaks exceeds that of sub-swarms, anti-convergence mechanism will expels the worst sub-swarms and reinitialize it in the search space whenever all swarms have converged. As a result, there is at least one swarm watching out for new peaks.

3 Hybrid PSO with gradient information

Compared to traditional optimization algorithms, for example, gradient descent method, particle swarm optimization wastes some computational effort by doing a random search, which reduces the convergence rate. Specifically, in MIDDLE, one of our concerns is to accelerate the convergence (namely the number of iterations needed) and to minimize the communication effort between central optimizer and participant's wearable devices. On the other hand, gradient descent method converges rapidly but may get stuck in local minimum and lose diversity soon. Thus, an approach that combines the strengths of both methods is of interest. In Noel's paper (Noel, 2012), he proposed the new gradient-based PSO algorithm (GPSO) in which the PSO algorithm is used for global exploration and a gradient based scheme is used for accurate local exploration. The GPSO algorithm avoids the use of inertial weights and constriction coefficients which can cause the PSO algorithm to converge to a local minimum prematurely if improperly chosen. To be more specific, in the new GPSO algorithm, the PSO algorithm is first used to approximately locate a good local minimum. Then after a fixed number of PSO iterations a gradient based local search is done with the best solution found by the PSO algorithm as its starting point. The results of local search are then compared with the global best position found by PSO and appropriate update will be completed and the same procedures will be repeated until convergence. In the paper, the quasi Newton-Raphson (QNR) algorithm was used to perform the local search. In Lakhbab and Bernoussi's work (Lakhbab & El Bernoussi, 2012), nonmonotone spectral gradient method was used for the local search. In Qteish and Hamdan's paper (Qteish & Hamdan, 2010), conjugate gradient (CG) optimization algorithm is used and adaptive weight factor for each particle and iteration number and periodic restart are two key features introduced from CG into the generic PSO algorithm. All of these hybrid PSO have shown superiority compared with the classical PSO algorithm, in terms of convergence speed and quality of obtained solutions.

4 Implementation of gradient PSO using steepest descent method

In this report, we implemented the gradient PSO using steepest descent method (SD). Specifically, the PSO iterations and SD iterations will be run alternately. SD will use the global best positions found by the last round of PSO iterations as its starting point. In each SD iteration, the gradient of the objective function at the starting point will be calculated and the negative of the gradient will be used as the steepest descent direction. The line search procedure, finding the optimal step size γ along this direction can be performed by trying multiple different step sizes and the one giving the best function improvement will be picked. In our experiment, 40 step sizes will be used to make sure the line search covers a wide range on the direction. Note that in MIDDLE, one SD iteration necessitates two round of communication between central optimizer and experiment participants, once for the calculation of gradient, and once for the line search. The bottleneck of MIDDLE is the communication effort, not the size of information exchanged (returned function values won't take too much space anyway). Therefore, we can always reduce the number of iterations by requesting more function evaluations in each communication.

Simulation results show that our implementation of GPSO presents faster convergence rate compared to existing R package *hydroPSO* (Zambrano-Bigiarini & Rojas, 2014). The test functions we are using are Rastrigin function and a Full Information Maximum Likelihood univariate model in the R package *OpenMx* (S. Boker et al., 2011).

The Rastrigin function is a non-convex function with a large number of local minimum and finding the minimum of this function is a fairly difficult problem, especially in high-dimensional case, since algorithms can easily stuck in local minimum. It is defined by:

$$f(x) = 10n + \sum_{i}^{n}[x_i^2 - 10cos(2\pi x_i)]$$

The optimal value of the function is 0 attained at origin. The plot of the 2-D function is shown as below.

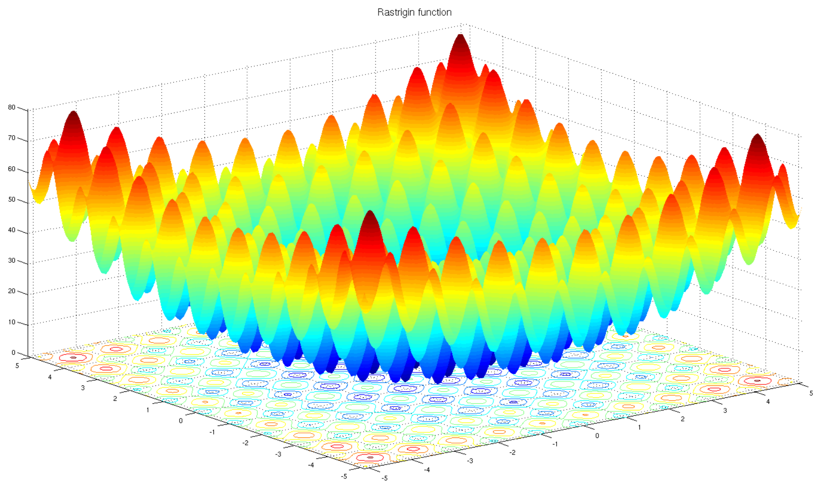

Figure 1: The 2-D Rastrigin function

The number of particles used in the test is 40. For GPSO, we run 5 iterations of PSO followed by 5 SD iterations and so on. We run the simulation for 10 times in each case. The results of using GPSO and hydroPSO on Rastrigin function are shown below. We can see that the GPSO convergence is much faster and the best value it can achieve during the simulation is comparable to that of standard PSO. However, the mean performance of GPSO in terms of accuracy is not as good as standard PSO. One of the possible reason is that the random exploration of the PSO iteration in GPSO fails to enter a better region and further guide the exploitation in the SD iterations. One possible solution might be to direct the exploration to more promising regions instead of searching around randomly. Therefore, more sophisticated modification to the algorithm is of importance.

Method	Dimension	Iteration	Best value	Average value
hydroPSO	2	93	0	0
GPSO	2	52	0	0
hydroPSO	3	131	0	0.4264
GPSO	3	71	0	1.1054
hydroPSO	4	183	0.9949	1.4213
GPSO	4	91	0.9949	2.8853

Table 1: The Rastrigin function experiment

We also run experiments on a Full Information Maximum Likelihood univariate model. This model consists of 5 parameters and the optimal value of the uni-modal likelihood function is 3151.4916549.

References

Blackwell, T., & Branke, J. (2004). Multi-swarm optimization in dynamic environments. In *Applications of evolutionary computing* (pp. 489–500). Springer.

Blackwell, T., & Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *Evolutionary Computation, IEEE Transactions on*, *10*(4), 459–472.

Blackwell, T. M. (2003). Swarms in dynamic environments. In *Genetic and evolutionary computationgecco 2003* (pp. 1–12).

Blackwell, T. M., Bentley, P. J., et al. (2002). Dynamic search with charged swarms. In *Gecco* (Vol. 2, pp. 19–26).

Boker, S., Neale, M., Maes, H., Wilde, M., Spiegel, M., Brick, T., ... others (2011). Openmx: an open source extended structural equation modeling framework. *Psychometrika*, *76*(2), 306–317.

Boker, S. M., Brick, T. R., Timo von Oertzen, C., Pritiken, J., Hunter, M., Maes, H., & Neale, M. C. (2013). Maintained individual data distributed likelihood estimation.

Carlisle, A., & Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments. In *Proceedings of the international conference on artificial intelligence* (Vol. 1, pp. 429–434).

Carlisle, A., & Dozier, G. (2001). Tracking changing extrema with particle swarm optimizer. *Auburn Univ., Auburn, AL, Tech. Rep. CSSE01-08*.

Clerc, M., & Kennedy, J. (2002). The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (Vol. 1, pp. 39–43).

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary computation, 2000. proceedings of the 2000 congress on* (Vol. 1, pp. 84–88).

Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of machine learning* (pp. 760–766). Springer.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of ieee international conference on neural networks* (Vol. 4, p. 1942-1948).

Lakhbab, H., & El Bernoussi, S. (2012). A hybrid method based on particle swarm optimization and nonmonotone spectral gradient method for unconstrained optimization problem. *International Journal of Math. Analysis*, *6.60*, 2963-2976.

Noel, M. M. (2012). A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. *Applied Soft Computing*, *12*(1), 353–359.

Qteish, A., & Hamdan, M. (2010). Hybrid particle swarm and conjugate gradient optimization algorithm. In *Advances in swarm intelligence* (pp. 582–588). Springer.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary computation proceedings, 1998. ieee world congress on computational intelligence., the 1998 ieee international conference on* (pp. 69–73).

Zambrano-Bigiarini, M., & Rojas, R. (2014). hydropso: Particle swarm optimisation, with focus on environmental models [Computer software manual]. Retrieved from `http://www.rforge.net/hydroPSO`, `http://cran.r-project.org/web/packages/hydroPSO` (R package version 0.3-4)