The Kalman Swarm

A New Approach to Particle Motion in Swarm Optimization

Christopher K. Monson and Kevin D. Seppi

Brigham Young University, Provo UT 84602, USA
c@cs.byu.edu or kseppi@cs.byu.edu

Abstract. Particle Swarm Optimization is gaining momentum as a simple and effective optimization technique. We present a new approach to PSO that significantly reduces the number of iterations required to reach good solutions. In contrast with much recent research, the focus of this work is on fundamental particle motion, making use of the Kalman Filter to update particle positions. This enhances exploration without hurting the ability to converge rapidly to good solutions.

1 Introduction

Particle Swarm Optimization (PSO) is an optimization technique inspired by social behavior observable in nature, such as flocks of birds and schools of fish [1]. It is essentially a nonlinear programming technique suitable for optimizing functions with continuous domains (though some work has been done in discrete domains [2]), and has a number of desirable properties, including simplicity of implementation, scalability in dimension, and good empirical performance. It has been compared to evolutionary algorithms such as GAs (both in methodology and performance) and has performed favorably [3, 4].

As an algorithm, it is an attractive choice for nonlinear programming because of the characteristics mentioned above. Even so, it is not without problems. PSO suffers from premature convergence, tending to get stuck in local minima [4–7]. We have also found that it suffers from an ineffective exploration strategy, especially around local minima, and thus does not find good solutions as quickly as it could. Moreover, adjusting the tunable parameters of PSO to obtain good performance can be a difficult task [7, 8].

Research addressing the shortcomings of PSO is ongoing and includes such changes as dynamic or exotic sociometries [6, 9–12], spatially extended particles that bounce [13], increased particle diversity [4, 5], evolutionary selection mechanisms [14], and of course tunable parameters in the velocity update equations [7, 8, 15]. Some work has been done that alters basic particle motion with some success, but the possibility for improvement in this area is still open [16].

This paper presents an approach to particle motion that significantly speeds the search for optima while simultaneously improving on the premature convergence problems that often plague PSO. The algorithm presented here, KSwarm, bases its particle motion on Kalman filtering and prediction.

We compare the performance of KSwarm to that of the basic PSO model. In the next section, the basic PSO algorithm is reviewed, along with an instructive alternative formulation of PSO and a discussion of some of its shortcomings. Unless otherwise specified, "PSO" refers to the basic algorithm as presented in that section. Section 3 briefly describes Kalman Filters, and Section 4 describes KSwarm in detail. Experiments and their results are contained in Section 5. Finally, conclusions and future research are addressed in Section 6.

2 The Basic PSO Algorithm

PSO is an optimization strategy generally employed to find a global minimum. The basic PSO algorithm begins by scattering a number of "particles" in the function domain space. Each particle is essentially a data structure that keeps track of its current position $\boldsymbol{x}$ and its current velocity $\boldsymbol{v}$. Additionally, each particle remembers the "best" (lowest valued) position it has obtained in the past, denoted $\boldsymbol{p}$. The best of these values among all particles (the global best remembered position) is denoted $\boldsymbol{g}$.

At each time step, a particle updates its position and velocity by the following equations:

$$\boldsymbol{v}_{t+1} = \chi\big(\boldsymbol{v}_t + \phi_1(\boldsymbol{p} - \boldsymbol{x}) + \phi_2(\boldsymbol{g} - \boldsymbol{x})\big) \tag{1}$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}_{t+1} \ . \tag{2}$$

The *constriction coefficient* $\chi = 0.729844$ is due to Clerc and Kennedy [15] and serves to keep velocities from exploding. The stochastic scalars ϕ_1 and ϕ_2 are drawn from a uniform distribution over $[0, 2.05)$ at each time step. Though other coefficients have been proposed in an effort to improve the algorithm [7, 8], they will not be discussed here in detail.

2.1 An Alternative Motivation

Although the PSO update model initially evolved from simulated flocking and other natural social behaviors, it is instructive to consider an alternative motivation based on a randomized hill climbing search. A naive implementation may place a single particle in the function domain, then scatter a number of random sample points in the neighborhood, moving toward the best sample point at each new time step: $\boldsymbol{x}_{t+1} = \boldsymbol{g}_t$.

If the particle takes this step by first calculating a velocity, the position is still given by (2) and the velocity update is given by

$$\boldsymbol{v}_{t+1} = \boldsymbol{g}_t - \boldsymbol{x}_t \ . \tag{3}$$

As this type of search rapidly becomes trapped in local minima, it is useful to randomly overshoot or undershoot the actual new location in order to do some directed exploration (after all, the value of the new location is already known). For similar reasons, it may be desirable to add momentum to the system, allowing

particles to "roll out" of local minima. Choosing a suitable random scalar ϕ, this yields

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \phi(\boldsymbol{g}_t - \boldsymbol{x}_t) \ . \tag{4}$$

The equation (4) is strikingly similar to (1). In fact, it is trivial to reformulate the PSO update equation to be of the same form as (1) [15, 12].

The fundamental difference between this approach and PSO is the way that $\boldsymbol{g}$ is calculated. In PSO, $\boldsymbol{g}$ is taken from other particles already in the system. In the approach described in this section, $\boldsymbol{g}$ is taken from disposable samples scattered in the neighborhood of a single particle.

This suggests that the basic PSO is a hill climber that uses existing information to reduce function evaluations. It is set apart more by its social aspects than by its motion characteristics, an insight supported by Kennedy but for different reasons [16].

2.2 Particle Motion Issues

Given that PSO is closely related to an approach as simple as randomized hill climbing, it is no surprise that attempts to improve the velocity update equation with various scaling terms have met with marginal success. Instead, more fundamental changes such as increased swarm diversity, selection, and collision avoiding particles have shown the greatest promise [4, 5, 14].

Unfortunately these methods are not without problems either, as they generally fail to reduce the iterations required to reach suitable minima. They focus primarily on eliminating stagnation, eventually finding better answers than the basic PSO without finding them any faster.

It has been pointed out that nonlinear programming is subject to a fundamental tradeoff between convergence speed and final fitness [4], suggesting that it is not generally possible to improve one without hurting the other. Fortunately, this tradeoff point has not yet been reached in the context of particle swarm optimization, as it is still possible to find good solutions more quickly without damaging final solution fitness.

For example, the development of a PSO visualization tool served to expose a particularly interesting inefficiency in the basic PSO algorithm. As the particles close in on $\boldsymbol{g}$ they tend to lose their lateral momentum very quickly, each settling into a simple periodic linear motion as they repeatedly overshoot (and undershoot) the target. This exploration strategy around local minima is very inefficient, suggesting that a change to particle motion may speed the search by improving exploration.

Such a change should ideally preserve the existing desirable characteristics of the algorithm. PSO is essentially a *social* algorithm, which gives it useful emergent behavior. Additionally, PSO motion is stochastic, allowing for randomized *exploration*. Particles also have *momentum*, adding direction to the random search. The constriction coefficient indicates a need for *stability*. Alterations to particle motion should presumably maintain these properties, making the Kalman Filter a suitable choice.

3 The Kalman Filter

Kalman filters involve taking noisy observations over time and using model information to estimate the true state of the environment [17]. Kalman filtering is generally applied to motion tracking problems. It may also be used for prediction by applying the system transition model to the filtered estimate.

The Kalman Filter is limited to normal noise distributions and linear transition and sensor functions and is therefore completely described by several constant matrices and vectors. Specifically, given an observation column vector $\mathbf{z}_{t+1}$, the Kalman Filter is used to generate a normal distribution over a belief about the true state. The parameters $\mathbf{m}_{t+1}$ and $\mathbf{V}_{t+1}$ of this multivariate distribution are determined by the following equations [18]:

$$\mathbf{m}_{t+1} = \mathbf{F}\mathbf{m}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mathbf{m}_t) \tag{5}$$

$$\mathbf{V}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\mathbf{V}_t\mathbf{F}^\top + \mathbf{V}_\mathbf{x}) \tag{6}$$

$$\mathbf{K}_{t+1} = (\mathbf{F}\mathbf{V}_t\mathbf{F}^\top + \mathbf{V}_\mathbf{x})\mathbf{H}^\top\big(\mathbf{H}(\mathbf{F}\mathbf{V}_t\mathbf{F}^\top + \mathbf{V}_\mathbf{x})\mathbf{H}^\top + \mathbf{V}_\mathbf{z}\big)^{-1} \ . \tag{7}$$

In these equations, $\mathbf{F}$ and $\mathbf{V}_\mathbf{x}$ describe the system transition model while $\mathbf{H}$ and $\mathbf{V}_\mathbf{z}$ describe the sensor model. The equations require a starting point for the filtered belief, represented by a normal distribution with parameters $\mathbf{m}_0$ and $\mathbf{V}_0$, which must be provided.

The filtered or "true" state is then represented by a distribution:

$$\mathbf{x}_t \sim \text{Normal}(\mathbf{m}_t, \mathbf{V}_t) \ . \tag{8}$$

This distribution may be used in more than one way. In some applications, the mean $\mathbf{m}_t$ is assumed to be the true value. In others, the distribution is sampled once to obtain the value. In this work, the latter is done.

The above describes how to do Kalman *filtering*, yielding $\mathbf{m}_t$ from an observation $\mathbf{z}_t$. A simple form of *prediction* involves applying the transition model to obtain a belief about the next state $\mathbf{m}'_{t+1}$:

$$\mathbf{m}'_{t+1} = \mathbf{F}\mathbf{m}_t \ . \tag{9}$$

There are other forms of prediction, but this simple approach is sufficient for the introduction of the algorithm in the next section, and for its use in particle swarms.

4 The Kalman Swarm (KSwarm)

KSwarm defines particle motion entirely from Kalman prediction. Each particle keeps track of its own $\mathbf{m}_t$, $\mathbf{V}_t$, and $\mathbf{K}_t$. The particle then generates an observation for the Kalman filter with the following formulae:

$$\boldsymbol{z}_v = \phi(\boldsymbol{g} - \boldsymbol{x}) \tag{10}$$

$$\boldsymbol{z}_p = \boldsymbol{x} + \boldsymbol{z}_v \ . \tag{11}$$

Similar to PSO, ϕ is drawn uniformly from $[0, 2)$, and the results are row vectors. The full observation vector is given by making a column vector out of the concatenated position and velocity row vectors: $\mathbf{z} = (\boldsymbol{z}_p, \boldsymbol{z}_v)^\top$. This observation is then used to generate $\mathbf{m}_{t+1}$ and $\mathbf{V}_{t+1}$ using (5), (6), and (7)

Once the filtered value is obtained, a prediction $\mathbf{m}'_{t+2}$ is generated using (9). Together, $\mathbf{m}'_{t+2}$ and $\mathbf{V}_{t+1}$ parameterize a normal distribution. We say, then, that

$$\mathbf{x}_{t+1} \sim \text{Normal}(\mathbf{m}'_{t+2}, \mathbf{V}_{t+1}) \ . \tag{12}$$

The new state of the particle is obtained by sampling once from this distribution. The position of the particle may be obtained from the first half of $\mathbf{x}_{t+1}^\top$, and the velocity (found in the remaining half) is unused.

This method for generating new particle positions has at least one immediately obvious advantage over the original approach: there is no need for a constriction coefficient. Particle momentum comes from the state maintained by the Kalman Filter rather than from the transition model. In our experiments, this eliminated the need for any explicit consideration of velocity explosion.

5 Experiments

KSwarm was compared to PSO in five common test functions: Sphere, DejongF4, Rosenbrock, Griewank, and Rastrigin. The first three are unimodal while the last two are multimodal. In all experiments, the dimensionality $d = 30$. The definitions of the five functions are given here:

$$\text{Sphere}(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 \tag{13}$$

$$\text{DeJongF4}(\boldsymbol{x}) = \sum_{i=1}^{d} i x_i^4 \tag{14}$$

$$\text{Rosenbrock}(\boldsymbol{x}) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{15}$$

$$\text{Rastrigin}(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 + 10 - 10\cos(2\pi x_i) \tag{16}$$

$$\text{Griewank}(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \ . \tag{17}$$

The domains of these functions are given in Table 1.

5.1 Experimental Parameters

In all experiments, a swarm size of 20 was used. Though various sociometries are available, the *star* (or *gbest* [1]) sociometry was used almost exclusively in

Table 1. Domains of Test Functions

Function	Domain
Sphere	$(-50, 50)^d$
DeJongF4	$(-20, 20)^d$
Rosenbrock	$(-100, 100)^d$
Griewank	$(-600, 600)^d$
Rastrigin	$(-5.12, 5.12)^d$

the experiments because it allows for maximum information flow [13]. Each experiment was run 50 times for 1000 iterations, and the results were averaged to account for stochastic differences. The parameters to (5), (6), and (7) are given below, and are dependent on the domain size of the function. The vector containing the size of the domain in each dimension is denoted $\boldsymbol{w}$. The column vector $\mathbf{w} = (\boldsymbol{w}, \boldsymbol{w})^\top$ is formed from two concatenated copies of $\boldsymbol{w}$. In the following equations, $\mathbf{I}_n$ is an identity matrix with n rows.

$$\mathbf{m}_0 = 0 \qquad \mathbf{V}_0 = \theta \operatorname{diag}(\mathbf{w}) \tag{18}$$

$$\mathbf{H} = \mathbf{I}_{2d} \qquad \mathbf{V}_\mathbf{z} = \theta \operatorname{diag}(\mathbf{w}) \tag{19}$$

$$\mathbf{F} = \begin{pmatrix} \mathbf{I}_d & \mathbf{I}_d \\ \mathbf{0} & \mathbf{I}_d \end{pmatrix} \qquad \mathbf{V}_\mathbf{x} = \theta \operatorname{diag}(\mathbf{w}) \ . \tag{20}$$

The initial mean $\mathbf{m}_0$ is a column vector of $2d$ zeros. The scalar θ indicates how large the variance should be in each dimension, and was set to 0.0001 for all experiments, as this produced a variance that seemed reasonable. The transition function simply increments position by velocity while leaving the velocity untouched.

All of the vectors used in the Kalman equations are of length $2d$ and all matrices are square and of size $2d$. This is the case because the model makes use of velocity as well as position, so extra dimensions are needed to maintain and calculate the velocity as part of the state. This implies that the sample obtained from (12) is *also* a vector of length $2d$, the first half of which contains position information. That position information is used to set the new position of the particle and the velocity information is unused except for the next iteration of the Kalman update equations.

5.2 Results

Table 2 shows the final values reached by each algorithm after 1000 iterations were performed. It is clear from the table that the KSwarm obtains values that are often several orders of magnitude better than the original PSO algorithm.

In addition to obtaining better values, the KSwarm tends to find good solutions in fewer iterations than the PSO, as evidenced by Figs. 1, 2, 3, 4, and 5. Note that each figure has a different scale.

Because the results obtained using the star sociometry were so striking, this experiment was also run using a sociometry where each particle had 5 neighbors. The corresponding results were so similar as to not warrant inclusion in this work.

Table 2. PSO vs. KSwarm Final Values

Function	PSO	KSwarm
Sphere	370.041	**4.723**
DejongF4	4346.714	**4.609**
Rosenbrock	2.61e7	**3.28e3**
Griewank	13.865	**0.996**
Rastrigin	106.550	**53.293**

These results represent a clear and substantial improvement over the basic PSO, not only in the final solutions, but in the speed with which they are found. It should be noted that much research has been done to improve PSO in other ways and that KSwarm performance in comparison to these methods has not been fully explored. The purpose of this work is to demonstrate a novel approach to particle motion that substantially improves the basic algorithm. The comparison and potential combination of KSwarm with other PSO improvements is part of ongoing research and will be a subject of future work.

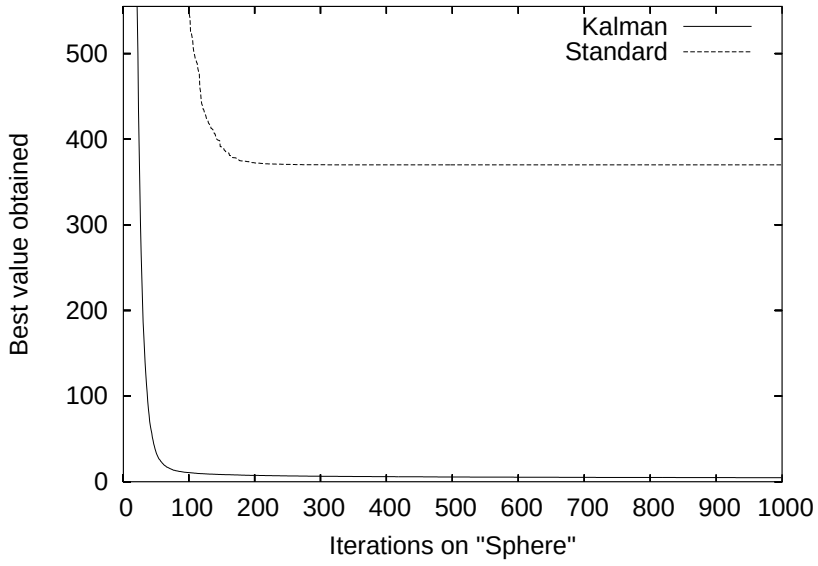

Fig. 1. Sphere

5.3 Notes on Complexity

It is worth noting that the Kalman motion update equations require more computational resources than the original particle motion equations. In fact, because

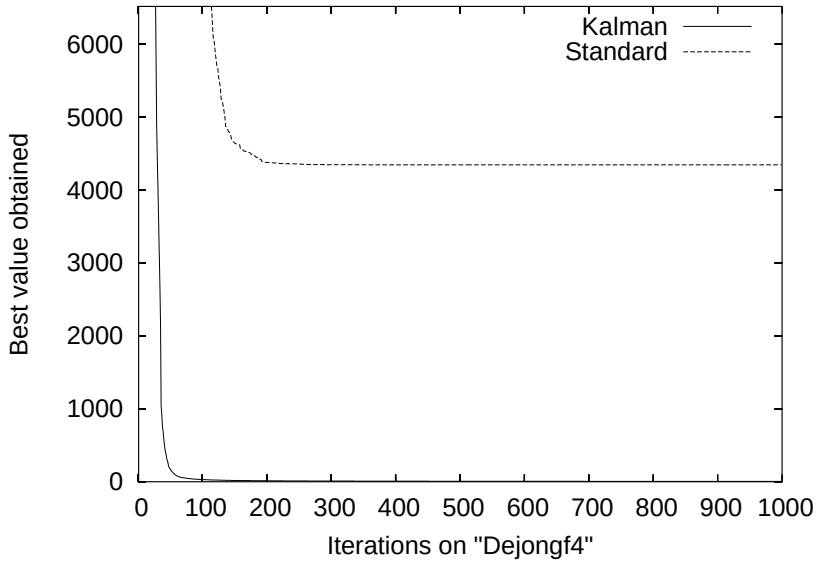

Fig. 2. DeJongF4

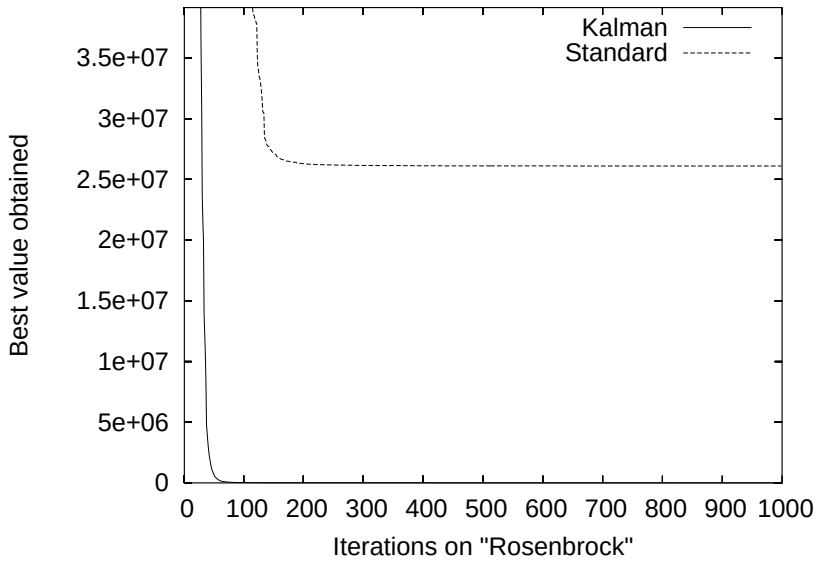

Fig. 3. Rosenbrock

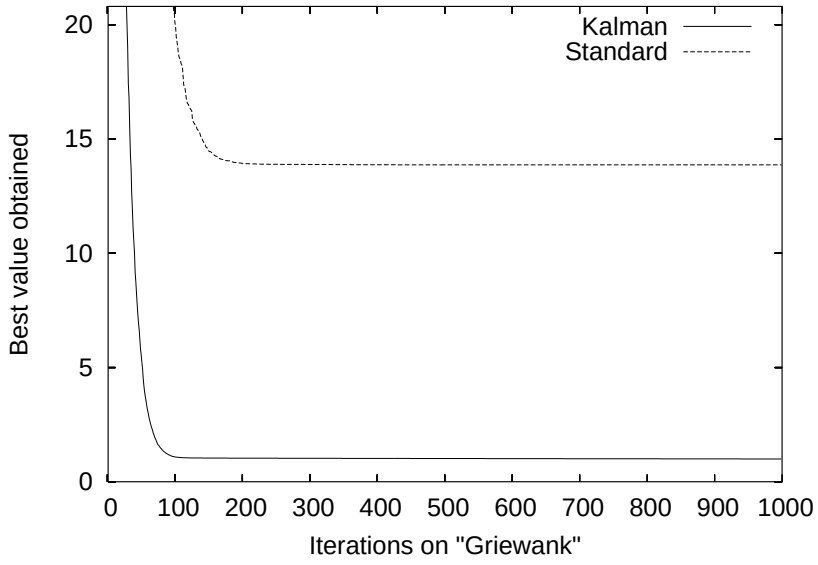

Fig. 4. Griewank

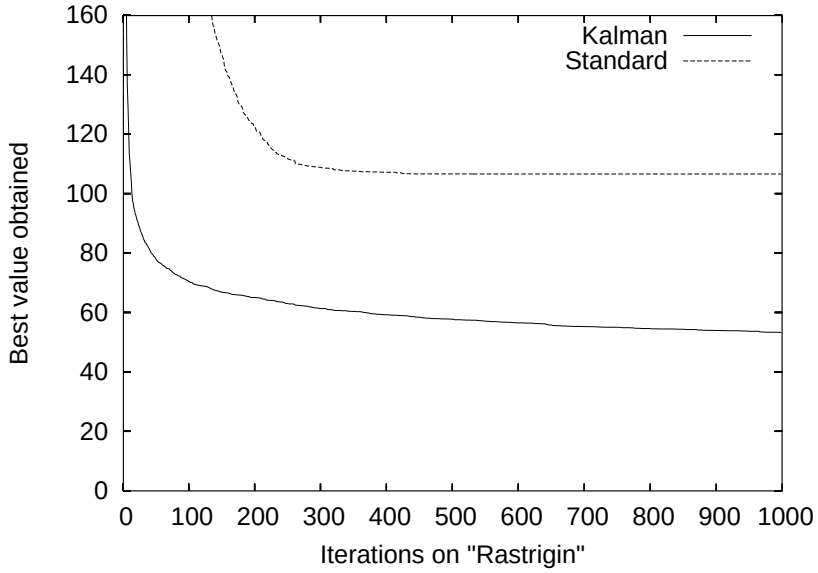

Fig. 5. Rastrigin

of the matrix operations, the complexity is $O(d^3)$ in the number of dimensions ($d = 30$ in our experiments). The importance of this increased complexity, however, appears to diminish when compared to the apparent exponential improvement in the number of iterations required by the algorithm. Additionally, the complexity can be drastically reduced by using matrices that are mostly diagonal or by approximating the essential characteristics of Kalman behavior in a simpler way.

6 Conclusions and Future Work

It remains to be seen how KSwarm performs against diversity-increasing approaches, but preliminary work indicates that it will do well in that arena, especially with regard to convergence speed. Since many methods which increase diversity do not fundamentally change particle motion update equations, combining this approach with those methods is simple. It can allow KSwarm to not only find solutions faster, but also to avoid the stagnation to which it is still prone.

Work remains to be done on alternative system transition matrices. The transition model chosen for the motion presented in this work is not the only possible model; other models may produce useful behaviors. Additionally, the complexity of the algorithm should be addressed. It is likely to be easy to improve by simple optimization of matrix manipulation, taking advantage of the simplicity of the model. More work remains to be done in this area.

KSwarm fundamentally changes particle motion as outlined in PSO while retaining its key properties of sociality, momentum, exploration, and stability. It represents a substantial improvement over the basic algorithm not only in the resulting solutions, but also in the speed with which they are found.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: International Conference on Neural Networks, IV (Perth, Australia), Piscataway, NJ, IEEE Service Center (1995) 1942–1948
2. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics, Piscataway, New Jersey (1997) 4104–4109
3. Kennedy, J., Spears, W.: Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Anchorage, Alaska (1998)
4. Riget, J., Vesterstroem, J.S.: A diversity-guided particle swarm optimizer - the ARPSO. Technical Report 2002-02, Department of Computer Science, University of Aarhus (2002)
5. Løvbjerg, M.: Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality. Master's thesis, Department of Computer Science, University of Aarhus (2002)

6. Richards, M., Ventura, D.: Dynamic sociometry in particle swarm optimization. In: International Conference on Computational Intelligence and Natural Computing. (2003)
7. Vesterstroem, J.S., Riget, J., Krink, T.: Division of labor in particle swarm optimisation. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii (2002)
8. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, New York (1998) 591–600
9. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii (2002)
10. Kennedy, J., Mendes, R.: Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. In: Proceedings of the 2003 IEEE SMC Workshop on Soft Computing in Industrial Applications (SMCia03), Binghamton, New York, IEEE Computer Society (2003)
11. Kennedy, J.: Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, Z., eds.: Proceedings of the Congress of Evolutionary Computation. Volume 3., IEEE Press (1999) 1931–1938
12. Mendes, R., Kennedy, J., Neves, J.: Watch thy neighbor or how the swarm can learn from its environment. In: Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana (2003) 88–94
13. Krink, T., Vestertroem, J.S., Riget, J.: Particle swarm optimisation with spatial particle extension. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii (2002)
14. Angeline, P.J.: Using selection to improve particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Anchorage, Alaska (1998)
15. Clerc, M., Kennedy, J.: The particle swarm: Explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6** (2002) 58–73
16. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana (2003) 80–87
17. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering **82** (1960) 35–45
18. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Second edn. Prentice Hall, Englewood Cliffs, New Jersey (2003)