01 Jan 2005

Fuzzy PSO: A Generalization of Particle Swarm Optimization

S. Abdelshahid

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Ashraf M. Abdelbar

Recommended Citation

S. Abdelshahid et al., "Fuzzy PSO: A Generalization of Particle Swarm Optimization," *Proceedings of the IEEE International Joint Conference on Neural Networks, 2005*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2005.
The definitive version is available at https://doi.org/10.1109/IJCNN.2005.1556004

Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005

Fuzzy PSO: A Generalization of Particle Swarm Optimization

Ashraf M. Abdelbar, Suzan Abdelshahid
Department of Computer Science
American University in Cairo

Donald C. Wunsch II
Department of Electrical and Computer Engineering
University of Missouri, Rolla

Abstract— **In standard particle swarm optimization (PSO), the best particle in each neighborhood exerts its influence over other particles in the neighborhood. In this paper, we propose fuzzy PSO, a generalization which differs from standard PSO in the following respect: charisma is defined to be a fuzzy variable, and more than one particle in each neighborhood can have a non-zero degree of charisma, and, consequently, is allowed to influence others to a degree that depends on its charisma. We evaluate our model on the weighted maximum satisfiability (max-sat) problem, comparing performance to standard PSO and to Walk-Sat.**

I. Introduction

Particle swarm optimization (PSO) [Kennedy and Eberhart, 1995; Kennedy and Eberhart, 2001] is a computational paradigm based on an analogy with models of the social behavior of groups of simple individuals. When PSO is used to solve a discrete optimization problem, a group, or *swarm*, of computational elements, or *particles*, is used to explore the solution space of a given instance $\mathcal{I}$ of the optimization problem. Each particle i stores a candidate solution vector x_i for $\mathcal{I}$, and stochastically modifies its candidate over time, based on the best solution found by the particle i itself, and based on the best solution of neighboring particles, where a neighborhood structure, defining particle adjacencies, is applied to the swarm.

In this paper, we propose *fuzzy PSO*, a generalization of PSO, which differs from standard PSO in only one respect: in each neighborhood, instead of only the best particle in the neighborhood being allowed to influence other particles in the neighborhood, several particles in each neighborhood can be allowed to influence others to a degree that depends on their degree of charisma, where charisma is defined to be a fuzzy variable. This will be described more fully in Section III.

We evaluate the performance of our proposed model on the weighted maximum satisifiability problem, comparing performance to standard PSO and to Walk-Sat [Selman *et al.*, 1994], a well-known satisfiability algorithm. We find that standard PSO does not perform well on Max-Sat in comparison to Walk-Sat. Adding fuzziness to standard PSO results in improved performance but still does not perform well compared to Walk-Sat. In previous work [Abdelbar and Abdelshahid, 2004], we introduced several modifications to the standard PSO model including the use of what we have called instinct factors, and of a neighborhood structure based on the hypercube topology. When these modifications are used with fuzzy PSO, performance compares favorably to Walk-Sat and to standard PSO with these same modifications, especially for larger problem sizes.

We begin by reviewing PSO more fully in Section II, and then introducing fuzzy PSO in more detail in Section III. Section IV reviews other modifications to the PSO model that we introduced in previous work. Section V presents experimental results on the weighted Max-Sat problem, and Section VI presents some conclusions and suggestions for future work.

II. Particle Swarm Optimization

Particle swarm optimization (PSO) [Kennedy and Eberhart, 1995; Kennedy and Eberhart, 2001] is a computational paradigm based on the phenomenon of collective intelligence exhibited by swarms of insects, and schools of fish, that has been applied to a variety of domains [Eberhart and Shi, 2001]. Recent applications of PSO include learning to play checkers [Franken and Engelbrecht, 2003], learning to play the Egyptian traditional board-game Seega [Abdelbar *et al.*, 2003; Abdelbar *et al.*, 2004], learning fuzzy cognitive maps [Parsopoulos *et al.*, 2003], detecting the existence of function roots [Parsopoulos and Vrahatis, 2003], data clustering [Van der Merwe and Engelbrecht, 2003], tracking extrema in dynamic environments [Li and Dam, 2003], and linear constrained function optimization [Paquet and Engelbrecht, 2003].

Like neural networks, computation in the PSO paradigm is based on a collection (called a *swarm*) of fairly-primitive processing elements (called *particles*). Connectivities are defined over the swarm, whereby each particle has an adjacency relationship with a subset of the other particles in the swarm. The neighborhood of each particle is the set of particles with which it is adjacent. The two most common neighborhood structures are *gbest*, in which the entire swarm is considered a single neighborhood, and *lbest*, in which the particles are arranged in a ring, and each particle's neighborhood consists of itself, its immediate ring-neighbor to the right, and its immediate ring-neighbor to the left.

Suppose we would like to use a swarm of M particles to solve a discrete combinatorial optimization problem whose candidate solutions can be represented as vectors of bits; let $\mathcal{I}$ be a given instance of such a problem. Let N denote the number of elements in the solution vector for $\mathcal{I}$. Each particle i would contain two N-dimensional vectors: a boolean vector x_i, which represents a candidate solution to $\mathcal{I}$ and is called particle i's state, and a real vector v_i, called the *velocity* of the

0-7803-9048-2/05/$20.00 ©2005 IEEE

particle. In the biological insect-swarm analogy, the velocity vector represents how fast, and in which direction, the particle is flying for each dimension of the problem being solved.

Let $\mathcal{N}(i)$ denote the neighbors of particle i, and let p_i denote the best solution ever found by particle i. In each time iteration, each particle i adjusts its velocity based on

$$v_i = \alpha v_i^{old} + \phi_1 r(\cdot)(p_i - x_i) + \phi_2 r(\cdot)(p_g - x_i) \ , \quad (1)$$

where α, called *inertia*, is a parameter within the range $[0,1]$ and is often decreased over time [Shi and Eberhart, 1998a]; ϕ_1 and ϕ_2 are two constants, often chosen so that $\phi_1 + \phi_2 = 4$ [Kennedy and Eberhart, 2001], which control the degree to which the particle "follows the herd" thus stressing exploitation (higher values of ϕ_2), or "goes its own way" thus stressing exploration (higher values of ϕ_1); $r(\cdot)$ is a uniformly random number generator function that returns values within the interval $(0,1)$; and g is the particle in i's neighborhood with the current neighborhood-best candidate solution.

For each dimension $d = 1, \ldots, N$, we then apply

$$v_{id} = \begin{cases} V_{max} & \text{if } v_{id} > V_{max} \\ v_{id} & \text{if } -V_{max} \leq v_{id} \leq V_{max} \\ -V_{max} & \text{if } v_{id} < -V_{max} \end{cases} \quad (2)$$

where V_{max} is a constant that limits the growth of velocity in either direction; then, we apply

$$Pr(x_{id} = 1) = \sigma(v_{id}) \ , \quad (3)$$

where σ is the sigmoidal function $\sigma(a) = \frac{1}{1+e^{-a}}$. Equation (3) can be implemented by generating a random number ρ within the interval $[0,1]$ and setting x_{id} to 1 if $\rho < \sigma(v_{id})$ and to 0 otherwise. We then determine the highest-quality solution within each neighborhood, according to a domain-dependent fitness measure, and adjust each particle's best neighbor pointer accordingly.

There has been some recent work [Clerc and Kennedy, 2002; Trelea, 2003; van den Bergh and Engelbrecht, 2002] on the convergence properties of PSO. Hierarchical neighborhood structures were introduced by Janson and Middendort [2003]; and dynamic neighborhoods were explored by Hu and Eberhart [2002]. Mendes *et al.* [2004] presented a fully-informed PSO variation, and the use of mutation in PSO was investigated by Stacey *et al.* [2003].

III. Fuzzy PSO

In this paper, we propose *fuzzy PSO* [Abdelshahid, 2004], a generalization of PSO, which differs from standard PSO in only one respect: in each neighborhood, instead of only the best particle in the neighborhood being allowed to influence its neighbors, several particles in each neighborhood can be allowed to influence others to a degree that depends on their degree of *charisma*, where charisma is a fuzzy variable. Several alternatives are possible for each of the following questions:

1) How many particles in each neighborhood will have non-zero charisma, and which particles will be selected to be charismatic?

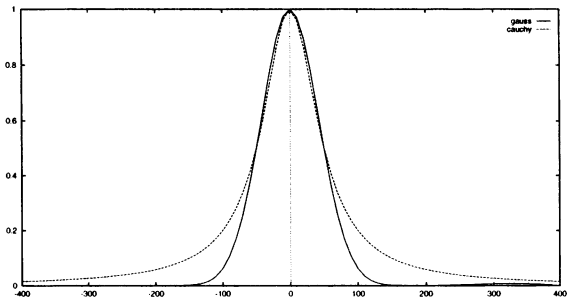

Fig. 1. A plot of a Gaussian function (solid line) with $\alpha = 0$ and $\sigma = 42.466$, and a Cauchy function (dashed line) with $\alpha = 0$, and $\beta = 50$. The two functions are aligned to have the same center and half-width. The two functions are very similar up to ± 50 (the half-width); beyond this point, the Cauchy function has a much lower rate of decay.

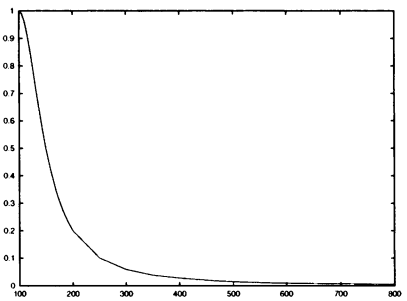

Fig. 2. A plot of $\psi(h)$ (y-axis) versus $f(p_h)$ (x-axis), based on $f(p_g) = 100$ and $\beta = 50$.

2) What membership function (MF) will be employed to determine level of charisma for each of the k selected particles?

In this paper, we use a simple rule to answer the first question: the k best particles in each neighborhood are selected to be charismatic, where k is a user-set parameter.

A number of alternatives are possible for the charisma MF. Popular MF choices include the triangular, trapezoidal, Gaussian, and Cauchy MFs [Jang *et al.*, 1996]. The Gaussian and Cauchy MFs, shown in Fig. 1, are specified by:

$$\text{gauss}(x; \alpha, \sigma) = e^{-\frac{1}{2}\left(\frac{x-\alpha}{\sigma}\right)^2} \ , \quad (4)$$

$$\text{cauchy}(x; \alpha, \beta) = \frac{1}{1 + \left(\frac{x-\alpha}{\beta}\right)^2} \ . \quad (5)$$

Compared to the Gaussian, we can see from the figure that the Cauchy function has a "wider tail;" we can illustrate this as follows. Consider a Gaussian function with center α_1 and half-width β_1, and a Cauchy function with center α_2 and half-width β_2, where the center α represents the point along the x-axis where the function output is 1, and $(\alpha \pm \beta)$ is the point along the x-axis at which the function output is 0.5. The output of the Gaussian function at the point $(\alpha_1 + 2\beta_1)$ is always 0.0625 while the output of the Cauchy function at $(\alpha_2 + 2\beta_2)$ is always 0.2, regardless of the values of the centers and half-widths. At the point $(\alpha_1 + 3\beta_1)$ and $(\alpha_2 + 3\beta_2)$, the output of the two functions is 0.002 and 0.1, respectively. At $(\alpha_1 + 10\beta_1)$ and $(\alpha_2 + 10\beta_2)$, the outputs are 7.8×10^{-31} and

0.01, respectively. The difference between the two functions continues to grow the further away from the center we move along the x-axis.

In this paper, we use an MF based on the Cauchy function. Let h be one of the k-best particles in a given neighborhood, and let $f(p_g)$ refer to the fitness of the very-best particle for the neighborhood under consideration. We compute the charisma $\psi(h)$ of particle h as

$$\psi(h) = \frac{1}{1 + \left(\frac{f(p_h) - f(p_g)}{\beta}\right)^2} . \quad (6)$$

Because $f(p_h) \not> f(p_g)$, $\psi(h)$ is actually half of a Cauchy function, as shown in Fig. 2. ψ is a decaying function that is 1 when $f(p_h) = f(p_g)$, and asymptotically approaches zero as $f(p_h)$ moves away from $f(p_g)$. $\psi(h)$'s output is 0.5 at $\alpha + \beta$. To avoid dependence on the scale of the fitness function, we set

$$\beta = \frac{f(p_g)}{\ell} , \quad (7)$$

where ℓ is a user-specified parameter. For a fixed $f(p_h)$, the larger the value of ℓ, the smaller the charisma $\psi(h)$. Figure 3 shows plots of ψ (y-axis) against $f(p_h)$ (x-axis), for several values of ℓ.

We employ an MF based on the Cauchy rather the Gaussian function because the amount of charisma allocated by the latter falls too quickly as we move away from $f(x_g)$. From our discussion above, a particle whose fitness is two units of half-width above the neighborhood-best is given a small charisma of 0.1 by equation (6) but would be given practically-zero charisma by a Gaussian-based MF. Therefore, the Cauchy will better promote search space exploration.

In fuzzy PSO, equation (1) is replaced by:

$$\begin{aligned} v_i &= \alpha v_i^{old} + \phi_1 r(\cdot)(p_i - x_i) \\ &\quad + \sum_{h \in \mathcal{B}(i,k)} \phi_2 r(\cdot)\psi(h)(p_h - x_i) , \end{aligned} \quad (8)$$

where $\mathcal{B}(i,k)$ denotes the k-best particles in the neighborhood of particle i. Each particle i is influenced by its own best solution p_i and the best solutions obtained by the k charismatic particles in its neighborhood, with the effect of each weighted by its charisma ψ. Of course, if k is taken as 1, this reduces to the standard PSO model.

IV. Other Modifications to PSO

In this section, we review a number of modifications to the standard PSO model, that we have presented in previous work [Abdelbar and Abdelshahid, 2004]. In this paper, we will refer to these modifications collectively as PSO+.

A. Instinct Factors

We propose a modification to PSO based on an idea from ant colony systems [Dorigo *et al.*, 1996]. Instead of the velocity-update Equation (1), we use

$$\begin{aligned} v_{id} &= \alpha v_{id}^{old} + \phi_1 r(\cdot)(p_{id} - x_{id}) \\ &\quad + \phi_2 r(\cdot)(p_{gd} - x_{id}) + \phi_3 \theta_d(x_i) , \end{aligned} \quad (9)$$

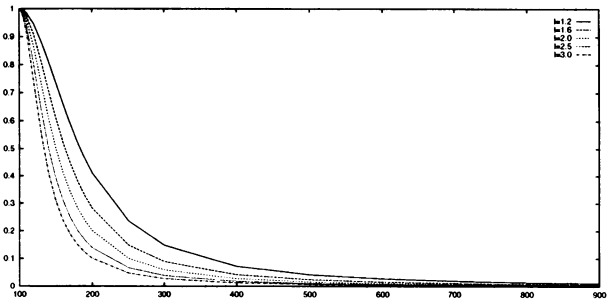

Fig. 3. Plots of $\psi(h)$ (y-axis) versus $f(p_h)$ (x-axis) for different values of ℓ, based on $f(p_g) = 100$.

where $\theta_d(x_i)$ is an *instinct* function that determines the particle's natural tendency towards the d^{th} bit of the solution vector, conditioned on the current solution string x_i. In the biological metaphor, this can be seen as modeling the primitive instinctive intelligence that even an insect might have. We will illustrate this in the context of the weighted max-sat problem.

An instance of weighted max-sat consists of a set U of n propositional variables, and a set C of m clauses, where each clause consists of a weight and a disjunction of literals, where each literal is a variable or its negation. The objective is to find a truth assignment to the n variables that satisfies the maximum-weight subset of the clauses. For example, consider the following instance:

$$\begin{array}{lll} 40: \ \overline{a} \vee b & 80: \ \overline{b} \vee c & 75: \ a \vee \overline{c} \\ 90: \ b \vee c & 50: \ \overline{a} \vee \overline{b} \vee \overline{c} & \end{array}$$

The optimal assignment for this instance, $\{a \rightarrow true, b \rightarrow false, c \rightarrow true\}$, leaves one clause unsatisfied: $(\overline{a} \vee b)$.

In modeling satisfiability problems with PSO, we would let the number of dimensions be equal to the number of propositional variables, and a value of 1, or 0, for a given dimension would indicate an assignment of true, or false, respectively, for the corresponding variable.

Let τ be an arbitrary truth assignment for U, and let v be an arbitrary member of U. Let C_v refer to the subset of clauses in which v participates. Then let $h(\tau, v \rightarrow true)$ refer to the weight of clauses in the set C_v left unsatisfied when v is forced to true and all other variables of τ remain unchanged; let $h(\tau, v \rightarrow false)$ be analogously defined. We then define $\theta_v(\tau)$ as

$$\theta_v(\tau) = 2 \times \frac{h(\tau, v \rightarrow false)}{h(\tau, v \rightarrow true) + h(\tau, v \rightarrow false)} - 1 . \quad (10)$$

The reasoning behind this equation is as follows. The function $\theta_v(\tau)$ measures the degree to which it is a good idea to set v to true assuming nothing else changes. For example, suppose a hypothesis h participates in a set of clauses C_v of total weight 2000. Now, suppose that if v is clamped to false, and all other hypotheses retain their values under τ, the weight of unsatisfied clauses within C_v is $h(\tau, v \rightarrow false) = 200$. Suppose the corresponding value for true, $h(\tau, v \rightarrow true)$, is 300. (Of course, $h(\tau, v \rightarrow true) + h(\tau, v \rightarrow false)$ will not necessarily equal the total weight of C_v because there could

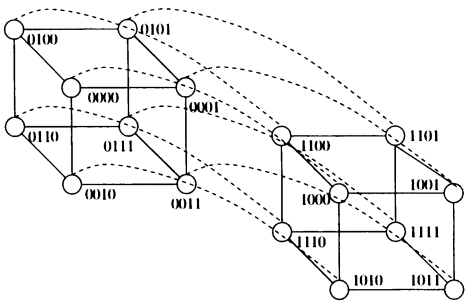

Fig. 4. An example of a hypercube topology (with number of dimensions equal to 4). Two nodes are adjacent if and only if the binary representation of their indices differ in only one bit position.

be clauses in C_v that are not satisfied under either truth value of v.) Then, the value of the θ function in our example would be equal to $(\frac{200}{500} \times 2) - 1 = -0.2$, and the effect of $\theta_d(x_{id})$ in equation (9) would be to move v_{id} downward.

Note that the instinct function described in this section is a modified version of the function used in previous work [Abdelbar and Abdelshahid, 2004].

In the case of fuzzy PSO+, equation (9) would be replaced by

$$\begin{aligned} v_{id} &= \alpha v_{id}^{old} + \phi_1 r(\cdot)(p_{id} - x_{id}) \\ &\quad + \sum_{h \in \mathcal{B}(i,k)} \phi_2 r(\cdot)\psi(h)(p_{hd} - x_{id}) \\ &\quad + \phi_3 \theta_d(x_i) \; . \end{aligned} \tag{11}$$

B. Hypercube Topology

With the addition of the instinct component, we have found that better performance is obtained with a hypercube topology than with the *gbest* topology. In a hypercube (see example in Fig. 4), the number of particles must be a power of 2; each particle is given an index between 0 and $2^k - 1$, for some k. Two particles are neighbors if the binary representations of their indices differ in exactly one bit position. The hypercube topology has been extensively studied as an interconnection network structure for parallel processing [Kumar *et al.*, 1994]. It has a number of desirable properties (let 2^k be the number of particles):

- each neighborhood contains exactly k particles,
- the maximum distance between any two particles (called the diameter) is exactly k,
- if particle i and particle j are neighbors, then i and j will not have any neighbors in common except for each other.

These properties suggest that the hypercube topology provides a good balance between effective communication (thus facilitating exploitation) and allowing independent solution trajectories to develop (thus facilitating exploration).

C. Stochastic Local Search

At the end of each iteration, the current state vector x_i for each particle i is used as the seed of a stochastic local search, as follows:

1) We repeatedly select a random bit $1 \leq d \leq N$ of i's state vector, x_{id}. This process is repeated N times until all bits have been selected.
2) Let $\ell_d(x_i)$ refer to the truth assignment produced by flipping the truth value of the selected bit x_{id} to the negation of its current value.
3) If the weight of clauses satisfied by $\ell_d(x_i)$ is greater than the weight of clauses satisfied by x_i, then x_i takes the value of $\ell_d(x_i)$. This new assignment takes effect immediately before we go back to step 1 to process the next bit.

The results of this process of course will depend on the random order in which the bits are selected in step 1. Therefore, for the same state vector, the results could be different for different runs of the stochastic search process.

V. Experimental Results

In this section, we present experimental results comparing five models: PSO, PSO+, Fuzzy PSO, Fuzzy PSO+, and Walk-Sat.

A. Experimental Suite and Parameter Settings

The number of particles used in PSO+ (and Fuzzy PSO+) is 64 particles. Since the computation of θ and the stochastic local search cause these methods to take more time than standard PSO, the number of particles in PSO (and Fuzzy PSO) is adjusted to 256 particles, to give the different methods roughly the same amount of time. We employ a suite of four large and ten small weighted max-sat instances to compare the different models:

- `auc-1`, `auc-2`, and `auc-3`: Three instances that we randomly generated with $(n; m)$ equal to (4,000;32,000), (6,000;48,000), and (10,000;80,000), respectively. These were randomly generated with the number of variables per clause varying from 3 to 6, and the clause-weight varying from 10 to 200. (Available from `www.aucegypt.edu/faculty/abdelbar/aucsat/`)
- `rndw2000`: A 2000-variable, 8500-clause instance obtained from `satlib` [Holger and Stützle, 2000].
- Ten 100-variable, 900-clause instances, (`jnh301` through `jnh310`), obtained from [Resende *et al.*, 1997].

The following experiments were executed for each instance:

- The PSO model was applied to each of these instances with $\phi_1 = \phi_2 = 2$, as recommended in [Shi and Eberhart, 1998b]. Ten runs were executed for each of the large instances. For the smaller `jnh` instances, 20 runs were executed.
- The PSO+ model was applied to each instance with $\phi_1 = 0.6$, and $\phi_2 = 1.1$. The parameter ϕ_3 was set to an initial value $\phi_3^{(0)}$ of 1.0 and its value $\phi_3(t)$ at iteration t was made equal to

$$\phi_3^{(t)} = \phi_3^{(0)} \cdot \left(1 - \frac{t}{1000}\right) \; . \tag{12}$$

These PSO+ parameters are based on experiments described in [Abdelbar and Abdelshahid, 2003]. The number of runs for each instance is the same as described for the PSO model above.

- The Fuzzy PSO model was applied for different values of the parameters k and ℓ. For each of the larger instances, the model was run 5 times for each value of $k \in \{3,4,5\}$, and for ℓ varying from $1.2, 1.4, \ldots, 3.0$ (a total of 30 parameter pair settings). For the smaller instances, the model was run 20 times for each parameter pair setting.
- The Fuzzy PSO+ model was also applied for different values of k and ℓ. The number of runs for each parameter pair is the same as described for Fuzzy PSO above. Based on preliminary experimentation, the ϕ_1 and ϕ_2 parameters were set to 0.6 and 0.5, respectively. The ϕ_3 parameter was set as described for the PSO+ model above.
- The Walk-Sat algorithm was run 5 times for each of the large instances and 20 times for each of the `jnh` instances, with its maximum number of steps set so that its run-time is commensurate with the other methods. We used Walser's well-known publicly-available (http://www.ps.uni-sb.de/~walser) implementation `wsatoip`.

B. Analysis of Results

Table I shows the average score (weight of clauses left unsatisfied) of the different methods on the various instances. For FPSO and FPSO+, performance of course depends on the setting of k and ℓ. We used the FPSO+ results of the four larger instances to determine a "default" parameter pair ($k = 5; \ell = 2.4$); we will refer to FPSO+ operating with this default parameter pair as dFPSO+. Table I presents the average score for the following (in order of column): pure PSO, FPSO averaged over all values of k and ℓ, PSO+, average FPSO+ for $k = 3$, average FPSO+ for $k = 4$, average FPSO+ for $k = 5$, dFPSO+, and Walk-Sat.

Let us examine the results, first, for the four large instances. For `auc-3`, we can see that:

- FPSO (averaged over all parameter pairs) is a small improvement over standard PSO;
- PSO+ dramatically improves on FPSO and also improves over Walksat;
- for FPSO+, performance is better on average the higher the value of k, with PSO+ performing better than the FPSO+ average for $k = 3$ and $k = 4$;
- dFPSO+ improves over PSO+ by over 12%.

The pattern is similar for the other large instances, except for `auc-2` where PSO+ is slightly better than dFPSO+ (however, if the k and ℓ parameters are individually optimized for `auc-2`, the FPSO+ score with the parameters $k = 5$ and $\ell = 1.6$ would be slightly better than PSO+).

The pattern for the smaller `jnh` instances is different:

- the performance difference between FPSO and PSO is more substantial than for the larger instances;
- the performance difference between PSO+ and FPSO is large but not as dramatic as for the larger instances;

TABLE I

This table shows the multiple-run average score (weight of unsatisfied clauses) of each of the following methods (in order of column): pure PSO (P), FPSO averaged over all parameter settings (FP), PSO+ (P+), FPSO+ averaged for $k = 3$ (F3+), FPSO+ for $k = 4$ (F4+), FPSO+ averaged for $k = 5$ (F5+), dFPSO+ (dF+), and Walk-Sat (WS). The term dFPSO+ refers to FPSO+ under the default parameter settings of $k = 5$ and $\ell = 2.4$.

name	P	FP	P+	F3+	F4+	F5+	dF+	WS
auc-3	238,918	229,657	1,955	2,471	2,063	1,752	1,714	4,194
auc-2	135,309	126,206	901	1,107	1,041	974	905	1,918
auc-1	85,361	77,981	395	427	385	343	300	509
rndw	38,710	34,398	917	866	786	784	774	1,341
jnh301	1,741	735	62	62	63	65	54	0
jnh302	3,299	1,914	695	723	755	775	774	395
jnh303	2,736	1,575	512	501	506	513	509	355
jnh304	3,261	1,803	426	433	444	466	467	321
jnh305	3,918	2,494	1,116	1,152	1,195	1,228	1,221	816
jnh306	2,432	1,195	30	38	43	48	40	16
jnh307	3,272	1,734	679	685	722	749	736	540
jnh308	2,930	1,369	289	288	290	297	293	130
jnh309	2,802	1,305	282	279	281	280	283	276
jnh310	3,526	2,135	607	628	667	696	508	463

- FPSO+ performs better the smaller the value of k, unlike for the large instances;
- however, FPSO+, even with ($k = 3$), does not improve over PSO+ for the smaller `jnh` instances;
- since the default parameters were chosen with the large instances in mind, their performance is expectedly poor against, say, the average for ($k = 3$);
- for the smaller instances, Walk-Sat outperforms all the PSO methods.

These observations tend to suggest that fuzzy PSO tends to be more useful the harder the problem instance. They also tend to suggest that larger values of k are more appropriate for larger problem sizes.

C. Sensitivity to Parameter Selction

To explore how sensitive FPSO+ is to parameter selection, we computed the score-ratio of the instance-specific worst parameter pair (within our experimental range) to the instance-specific best parameter pair; of the average over all parameter pairs to the instance-best parameter pair; and of the default pair to the instance-best pair. Table II shows these ratios for each instance, with the first column specifying the value of the instance-best parameter pair. The final row gives the ratio averages for the ten smaller `jnh` instances; the $(5; 2.4)$ parameter pair is used as the default for the smaller instances even though it performs poorly for them. From the table, we note that the ratio of the average over all parameter pairs to the instance-specific best setting is no more than 128%, and even the ratio of the worst to the best setting is no more than 172%.

VI. Conclusions and Future Work

We have presented a fuzzy generalization of particle swarm optimization, based on allowing more than one particle in each neighborhood to influence its neighbors, depending on its degree of charisma. Our experimental results on max-sat suggest that our proposed generalization improves performance, especially for larger problem instances. The results

TABLE II

How sensitive is FPSO+ to parameter selection? This table shows, for each of the four larger instances in our test suite, the instance-specific best parameter pair, the ratio of the average score over all parameter pairs to the score of the instance-best parameter pair, the ratio of the score of the instance-specific worst pair to the score of the instance-best pair, and the ratio of the default pair $(5; 2.4)$ to the instance-best pair. We can see that the ratio of the average over all pairs to the instance-best pair is never more than 128%, and even the ratio of the worst to the best setting is no more than 172%.

	$(k;\ell)$	average / best	worst / best	default / best
auc-3	(5;2.0)	1.24	1.57	1.02
auc-2	(5;1.6)	1.16	1.37	1.01
auc-1	(5;2.4)	1.28	1.72	1.00
rndw	(5;1.6)	1.12	1.27	1.06
jnh (avg)	(3;1.6)	1.09	1.22	1.10

also suggest that although performance is affected by the settings of the two parameters k and ℓ, good performance is possible with default parameter settings.

An extension we would like to consider in the future is the use of the generalized bell function (of which the Cauchy function is a special case) in the charisma MF. The generalized bell has the form

$$\text{gbell}(x;\alpha,\beta,\gamma) = \frac{1}{1+\left(\frac{x-\alpha}{\beta}\right)^{2\gamma}}, \qquad (13)$$

where increasing the additional parameter γ has the effect of making the curve flatter at the top and narrower at the bottom. Another possible extension is to allow the value of k and ℓ (and γ if the generalized bell is used) for each particle to change dynamically, based on the relative performance of the particle.

An interesting alternative for selecting which particles to be charismatic is to choose high-fitness particles whose proposed solution differs from that of the neighborhood-best by more than a certain threshold.

ACKNOWLEDGEMENTS

We would like to acknowledge Taysir El-Hawary and Karim El-Gebaly for programming some of the features described in this paper, for running some of the experiments, and for contributing to the design of the θ function described in Section IV.A.

REFERENCES

[Abdelbar and Abdelshahid, 2003] A.M. Abdelbar, and S. Abdelshahid, Swarm optimization with instinct-driven particles. *Proceedings IEEE Congress on Evolutionary Computation*, 2003.

[Abdelbar and Abdelshahid, 2004] A.M. Abdelbar, and S. Abdelshahid, Instinct-based PSO with local search applied to satisfiability. *Proceedings International Joint Conference on Neural Networks*, 2004.

[Abdelbar *et al.*, 2003] A.M. Abdelbar, S. Ragab, and S. Mitri. Applying co-evolutionary particle swarm optimization to the Egyptian board game Seega. *Proc.* **CEC-03** *Workshop on Genetic Programming*, 2003, p. 9-15.

[Abdelbar *et al.*, 2004] A.M. Abdelbar, S. Ragab, and S. Mitri. Co-evolutionary particle swarm optimization applied to the 7×7 Seega game. *Proceedings International Joint Conference on Neural Networks*, 2004.

[Abdelshahid, 2004] S. Abdelshahid. Variations of particle swarm optimization and their experimental evaluation on maximum satisfiability. M.S. Thesis (Ashraf Abdelbar, advisor). Department of Computer Science, American University in Cairo, May 2004.

[Clerc and Kennedy] M. Clerc and J. Kennedy. The particle swarm—explosion, stability, and convergence in multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, Vol. 6, 2002, pp. 58-73.

[Dorigo *et al.*, 1996] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, pp. 29-41, 1996.

[Eberhart and Shi, 2001] R.C. Eberhart, and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings IEEE International Conference on Evolutionary Computation*, pp. 81-86, 2001.

[Franken and Engelbrecht, 2003] N. Franken, and A.P. Engelbrecht. Comparing PSO structures to learn the game of checkers from zero knowledge. *Proceedings 2003 IEEE Congress on Evolutionary Computation*, 2003.

[Holger and Stützle, 2000] H.H. Hoos, and T. Stützle. SATLIB: An online resource for research on SAT. In: I.P. Gent, H.V. Maaren, T. Walsh, eds., *SAT 2000*. IOS Press, 2000, pp. 283-292.

[Hu and Eberhart, 2002] X. Hu, and R.C. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings Congress on Evolutionary Computation*, pp. 1677-1681, 2002.

[Jang *et al.*, 1996] J.-S.R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, 1996.

[Janson and Middendorf, 2003] S. Janson, and M. Middendorf. A hierarchical particle swarm optimizer. *Proceedings 2003 IEEE Congress on Evolutionary Computation*, 2003.

[Kennedy and Eberhart, 1995] J. Kennedy, and R.C. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, Vol. IV, pp. 1942-1948, 1995.

[Kennedy and Eberhart, 2001] J. Kennedy, and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[Li and Dam, 2003] X. Li, and K.H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. *IEEE 2003 Congress on Evolutionary Computation*, 2003.

[Mendes *et al.*, 2004] R. Mendes, J. Kennedy, and J. Neves, The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 4, 2004.

[Paquet and Engelbrecht, 2003] U. Paquet, and A.P. Engelbrecht. A new particle swarm optimiser for linearly constrained optimization. *IEEE Congress on Evolutionary Computation*, 2003.

[Parsopoulos and Vrahatis, 2003] K.E. Parsopoulos, and M.N. Vrahatis. Investigating the existence of function roots using particle swarm optimization. *IEEE Congress on Evolutionary Computation*, 2003.

[Parsopoulos *et al.*, 2003] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, and M.N. Vrahatis. A first study of fuzzy cognitive maps learning using particle swarm optimization. *IEEE Congress on Evolutionary Computation*, 2003.

[Resende *et al.*, 1997] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, Vol. 35, pp. 393-405, 1997.

[Selman *et al.*, 1994] B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings National Conference on Artificial Intelligence*, pp. 337-343, 1994.

[Shi and Eberhart, 1998a] Y. Shi, and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings IEEE International Conference on Evolutionary Computation*, pp. 69-73, 1998.

[Shi and Eberhart, 1998b] Y. Shi, and R.C. Eberhart. Parameter selection in particle swarm optimization. In *Proceedings Annual Conference on Evolutionary Programming*, 1998.

[Stacey *et al.*, 2003] A. Stacey, M. Jancic, and I. Grundy. Particle swarm optimization with mutation. In *Proceedings IEEE Congress on Evolutionary Computation*, 2003.

[Trelea, 2003] I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, Vol. 85, 2003, pp. 317-325.

[van den Bergh and Engelbrecht, 2002] F. van den Bergh, and A.P. Engelbrecht. A new locally convergent particle swarm optimizer. *Proceedings IEEE International Conference on Systems, Man and Cybernetics*, pp. 94-99, 2002.

[van der Merwe and Engelbrecht, 2003] D.W. van der Merwe, and A.P. Engelbrecht. Data clustering using particle swarm optimization. *Proceedings IEEE Congress on Evolutionary Computation*, 2003.