Anarchic Society Optimization: A human-inspired method

Conference Paper · June 2011

DOI: 10.1109/CEC.2011.5949940 · Source: DBLP

1 author:

Amir Ahmadi Javid
AUT | Tehran Polytechnic
57 PUBLICATIONS **915** CITATIONS

SEE PROFILE

Anarchic Society Optimization: A Human-Inspired Method

A. Ahmadi-Javid
Department of Industrial Engineering
Amirkabir University of Technology
Tehran, Iran
ahmadi_javid@aut.ac.ir

Abstract— **This paper introduces Anarchic Society Optimization (ASO), which is inspired by a social grouping in which members behave anarchically to improve their situations. The basis of ASO is a group of individuals who are fickle, adventurous, dislike stability, and frequently behave irrationally, moving toward inferior positions they have visited during the exploration phase. The level of anarchic behavior among members intensifies as the level of difference among members' situations increases. Using these anarchic members, ASO explores the solution space perfectly and avoids falling into local optimum traps. First we present a unified framework for ASO, which can easily be used for both continuous and discrete problems. Then, we show that Particle Swarm Optimization (PSO), for which a general introduction was initially implemented for continuous optimization problems, is a special case of this framework. To evaluate the performance of ASO for discrete optimization, we develop an ASO algorithm for a challenging scheduling problem. The numerical results show that the proposed ASO algorithm significantly outperforms other effective algorithms in the literature. Our study indicates that developing an ASO algorithm is basically straightforward for any problem to which a PSO or Genetic algorithm has been applied. Finally, it is shown that under mild conditions an ASO algorithm converges to a global optimum with probability one.**

Keywords- Swarm intelligenc; Anarchic Society Optimization (ASO); Combinatorial optimization; Continuous optimization; Particle Swarm Optimization (PSO); Genetic algorithms; Convergenc

I. INTRODUCTION

Over the past decades, population-based random-optimization methods have been widely employed to solve difficult combinatorial optimization problems. Genetic algorithms, evolutionary programming, evolution strategies and genetic programming are inspired by natural evolution. Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are the most important members of a wider class of swarm-intelligence methods that are used to solve optimization problems. The field of swarm intelligence is an emerging research area that draws on social insect colonies and animal groups to infer principles of self-organization and cooperation.

In this paper a new swarm-intelligence method is introduced. The main contributions are as follows:

1) It is shown how we can develop swarm-intelligence methods that are based on human societies. Unlike the existing swarm-intelligence methods designed on the basis of normal insect or animal social groupings, designing an algorithm by studying normal human societies can only have very limited success since normal and well-organized societies are controlled by large bodies of laws, and thus, generally, there is no possibility for society members to achieve their desires by way of individual approaches. In fact, members of a normal society are controlled by an authority, so no member of the society is really self-organized and able to radically improve his/her situation in a short period of time. This realization led us to look into an optimization method based on an abnormal human society. The proposed method is called Anarchic Society Optimization (ASO).

2) Despite PSO and ACO which were initialy introduced for continuous and discrete optimization, respectively, ASO is more general and can basically be used in both continuous and discrete optimization.

3) ASO can be applied in a straightforward manner to any problem that has been solved by a PSO or Genetic algorithm.

4) Under mild conditions, an ASO algorithm converges in the limit to a globally optimal solution with probability one. Our analysis indicates that the chracteristics of only one member may guarantee convergence of a swarm-intelligenc method.

The remainder of the paper is organized as follows. Section II introduces the framework of the new method. In Section III, it is shown that PSO is a particular case of this framework. Section IV develops an algorithm based on the new method for a discrete optimization problem. A convergence analysis is presented in Section V. Finally, Section VI concludes the paper.

II. ANARCHIC SOCIETY OPTIMIZATION (ASO)

The structure of a nature-inspired swarm intelligence method strongly depends on the personal and social characteristics of its population's members. Therefore,

selecting an appropriate underlying society is very important in designing such methods. To the best of our knowledge, no method considers human societies for this purpose. The fact that humankind has very special and unique characteristics motivated us to develop a human-inspired swarm intelligence method, called Anarchic Society Optimization (ASO). The new method is based on an abnormal human society instead of a swarm of birds or a colony of ants, which are the basis of PSO and ACO, respectively.

ASO is an innovative optimization method inspired by a human society whose members behave anarchically to improve their situations. In ASO the members are fickle, and their unpredictability increases as their situation worsens. They also behave irrationally and adventurously, moving toward the inferior positions they have visited. Using these anarchic members, ASO is able to search the solution space perfectly and avoid falling into local optimum traps. In the following, we present the ASO framework in general for any optimization problem.

A. Basic Assumptions and Notation

Let S be a solution space and $f : S \to \Re$ be a function needs to be minimized over S. Consider a society of N members searching within an unknown land, that is, the solution space, for the best place to live, that is, the global minimizer of f over S. The main characteristic of the society is that its members are adventurous and behave anarchically during their search procedure.

By $X_i(k)$ we denote the position of member i in iteration k of the exploring procedure. All of the members are aware of the best global position visited by the whole society in the first k iterations. This position is denoted by $G(k)$ and is called G-best. They also realize member i_k^* who occupies the best position in the society in iteration k. The best personal position previously visited by member i in iteration k is denoted by $P_i(k)$, called P-best.

The ASO framework is presented in Fig. 1. In the next subsections, the elements of the algorithm are explained.

B. Planning procedure for movement

Each member has a planning procedure to decide how it will move and change his/her position in the next iteration. To this end, each member provides three movement policies and then combines them to determine his/her position in the next iteration. These movement policies are described in the following subsections.

1) Selection of movement policy based on the current position

The first movement policy in iteration k is denoted by $\mathrm{MP}_i^{\mathrm{Current}}(k)$ and is chosen on the basis of the current position. In general, the movement policy $\mathrm{MP}_i^{\mathrm{Current}}(k)$ is a neighboring method. Each member can choose a different neighboring method, or all of them may prefer a common one. For member i_k^*, we suggest to use a neighboring method with a higher level of diversification, to avoid falling into local optima.

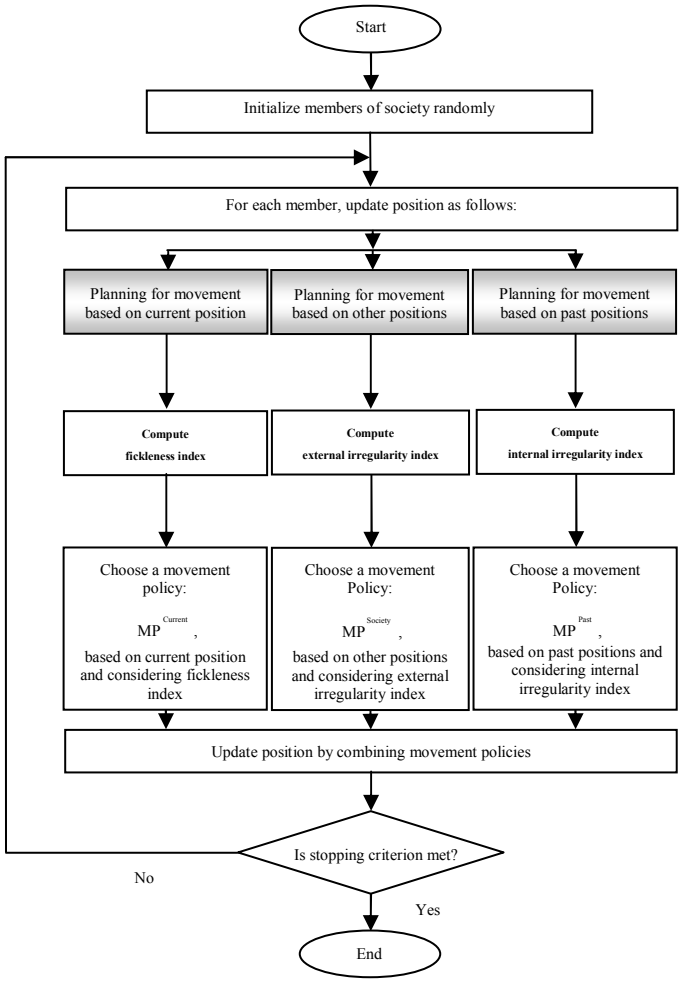

Figure 1. Framework of Anarchic Society Optimization (ASO)

We consider the *fickleness index* $FI_i(k)$ for member i in iteration k. This index is a measure of member i's dissatisfaction for his/her current situation, compared to other members' situations. When the objective function f is positive on S, $FI_i(k)$ may be defined as one of the following, for some nonnegative number α_i in $[0,1]$:

$$FI_i(k) = 1 - \alpha_i \frac{f(X_{i_k^*}(k))}{f(X_i(k))} - (1-\alpha_i)\frac{f(P_i(k))}{f(X_i(k))} \quad (1)$$

$$FI_i(k) = 1 - \alpha_i \frac{f(G(k))}{f(X_i(k))} - (1-\alpha_i)\frac{f(P_i(k))}{f(X_i(k))}, \quad (2)$$

which are numbers in the interval $[0,1]$.

In practice, depending on $FI_i(k)$, member i may choose different neighboring methods. It may be better for large values of $FI_i(k)$ to choose a neighboring method that changes the

current position more. Note that, for continuous problems, we can use this index more flexibly.

2) Selection of movement policy based on other members' positions

The second movement policy in iteration k is denoted by $\mathrm{MP}_i^{\mathrm{Society}}(k)$ and is chosen based on the positions of the other members. It is logical and regular that each member would generate his/her movement policy $\mathrm{MP}_i^{\mathrm{Society}}(k)$ based on G-best (or position of member i_k^*); however, because members are irregular and adventurous, they may select any one of the other members' positions (or a number of them) to generate a movement policy. Hence, we define the *external irregularity index* $EI_i(k)$ for member i in iteration k. This index can be used in two scenarios:

- In the first scenario, we consider $EI_i(k)$ as the probability that member i will behave irregularly and generate his/her movement policy based on another randomly selected member's position, which does not correspond to G-best.
- In the second scenario, we compare $EI_i(k)$ with a threshold. If it is greater than the threshold, then member i will behave irregularly.

The number $EI_i(k)$ can be defined on the basis of member i's situation relative to G-best (or the i_k^*th member's position) for some positive number θ_i:

$$EI_i(k) = 1 - e^{-\theta_i [f(X_i(k)) - f(G(k))]}. \quad (3)$$

However, we suggest to define $EI_i(k)$ using the level of diversity in the society; for example, consider the following external irregularity index:

$$EI_i(k) = 1 - e^{-\delta_i D(k)} \quad (4)$$

where δ_i is a positive number and $D(k)$ is a suitable dispersion measure, e.g., the coefficient of variation $CV(k)$ of the members' objective values when the objective function f is positive on S (coefficient of variation = standard-deviation/mean). This means that, if the diversity of the society increases, the members will tend to behave more irregularly, which is expected behavior for an anarchic society.

3) Selection of movement policy based on past positions

The third movement policy in iteration k is denoted by $\mathrm{MP}_i^{\mathrm{Past}}(k)$ and is chosen based on the past positions that were visited by each member. It would be more normal for each member to generate movement policy $\mathrm{MP}_i^{\mathrm{Past}}(k)$ based on P-best, but because the members are lawless, they may select any past position (or a number of them) to generate a movement policy. Hence, we define the *internal irregularity index* $II_i(k)$ for member i in iteration k, which can be used in the scenarios presented for $EI_i(k)$ in the previous subsection.

In selecting each type of movement policy, an important issue is how a member can generate a movement policy based on its current position, the other members' positions or its past positions. For continuous problems, there are several algebraic ideas about this; however, this matter requires more attention and innovation when it comes to discrete problems. A stimulating example is coding each solution of a discrete problem as a chromosome, and then using mutation or crossover operators as movement policies.

C. Combination rule

After choosing movement policies $\mathrm{MP}^{\mathrm{Current}}$, $\mathrm{MP}^{\mathrm{Society}}$, $\mathrm{MP}^{\mathrm{Past}}$, each member must combine these policies to move toward a new position, so he/she requires a combination rule. The simplest approach to this is to choose the movement policy that yields the best new position, which we term *elitism combination rule*. An alternative is that the policies be applied sequentially on the current position, this may be called the *sequential combination rule*. Other kinds of combination rules can be defined depending on the problem definition. For example, when each solution of a discrete problem is coded as a chromosome, we can use a crossover operator to combine all the positions obtained from the three movement policies. Note that members may use different combination rules.

III. PSO AS A SPECIAL CASE OF ASO

After studying the social behavior of birds, Kennedy and Eberhart [1] originated PSO in 1995. The main idea behind PSO is that each bird, called particle, flies through the solution space of the optimization problem to search for the optimal solution. Each particle determines its velocity based on personal experience and information gained through interaction with the swarm.

The framework of PSO is basically presented for unconstrained continuous optimization in $\Re^d$. In a continuous solution space, each dimension of the position vector corresponds to the value of a decision variable for the problem. In other words, the position of each particle is a potential solution to the problem at hand, and the fitness of this particle can be calculated by putting these values into a predetermined objective function. When the fitness is more desirable in terms of the objective function, the particle's position is better. The mathematical description of PSO is as follows.

Suppose a swarm of N particles searching for the globally optimal solution within a d-dimensional solution space. Two d-dimensional vectors are assigned to particle i in iteration k as follows:

$X_i(k)$: Position

$V_i(k)$: Velocity.

The d-dimensional vectors $P_i(k)$ and $G(k)$ are defined similarly as in ASO.

The new velocity of each particle is calculated as follows:

$$V_i(k+1) = \omega V_i(k) + \lambda_1 r_1(k)[G(k) - X_i(k)] + \lambda_2 r_2(k)[P_i(k) - X_i(k)] \quad (5)$$

where λ_1 and λ_2 are positive constants called acceleration coefficients, ω is a positive constant called the inertia factor, and $r_1(k)$ and $r_2(k)$ are two numbers that are independently generated from a given probability distribution over the interval $(0,1)$ in iteration k. Finally, the position of each particle is updated using the following equation:

$$X_i(k+1) = X_i(k) + V_i(k+1). \quad (6)$$

The basic steps of the standard PSO are presented below:

Step 1: Initialize a population of particles with random positions and velocities.

Step 2: Determine P-best and G-best in the current iteration.

Step 3: Update the velocity and position of each particle in the newest iteration according to Equations (5) and (6).

Step 4: If the stopping criterion is met, then stop; otherwise, go to Step 2.

It is easy to show that PSO is a special case of the ASO framework. Let us define the following velocities:

$$V_i^{\text{Current}}(k) = \omega V_i^{\text{Sum}}(k)$$

$$V_i^{\text{Society}}(k) = \lambda_1 r_1(k)[G(k) - X_i(k)]$$

$$V_i^{\text{Past}}(k) = \lambda_2 r_2(k)[P_i(k) - X_i(k)]$$

where $V_i^{\text{Sum}}(k) = V_i^{\text{Current}}(k-1) + V_i^{\text{Society}}(k-1) + V_i^{\text{Past}}(k-1)$.

Consider the following movement policies and combination rule.

$\text{MP}_i^{\text{Current}}(k)$: move with velocity $V_i^{\text{Current}}(k)$ for one unit of time,

$\text{MP}_i^{\text{Society}}(k)$: move with velocity $V_i^{\text{Society}}(k)$ for one unit of time,

$\text{MP}_i^{\text{Past}}(k)$: move with velocity $V_i^{\text{Past}}(k)$ for one unit of time.

Combination rule: sequentially apply the above movement policies.

The new position of member i after applying the above movement policies is similar to the case he/she moves with velocity $V_i^{\text{Sum}}(k+1) = V_i^{\text{Current}}(k) + V_i^{\text{Society}}(k) + V_i^{\text{Past}}(k)$ for one unit of time. Thus, we have $X_i(k+1) = X_i(k) + V_i(k+1)$. This shows that PSO is a particular case of ASO; therefore, the proposed ASO framework can be used for all problems recently tackled by PSO algorithms, especially continuous optimization problems.

IV. Implementation of ASO for Hybrid Flowshops

To show how ASO can be implemented for discrete problems, this section presents an ASO algorithm for the sequence-dependent setup-time hybrid-flowshop scheduling problem (SDST-HFSP) whose objective is to minimize the makespan. SDST-HFSP is a challenging NP-hard problem which has recently been studied by several authors. See [2] for an excellent review of this problem.

In the following, we briefly describe the proposed ASO algorithm. For more details on this algorithm and the numerical results refer to [3].

The most important issue in applying ASO to scheduling problems is to find a suitable method to relate job sequences and the members' positions. A suitable method is to represent each scheduling solution by a chromosome. This enables us to use genetic operators such as crossover and mutation to easily generate movement policies. To represent each solution of the SDST-HFSP, it is enough to code only the sequence of all the jobs at the first stage.

An example for the representation is shown in Fig. 2. In this example, there are ten jobs and three machines in the first stage. Based on Fig. 2, jobs 1, 4 and 8 with order 4→8→1 are assigned to machine 1; jobs 2 and 3 with order 2→3 are assigned to machine 2; jobs 9, 10, 5, 7 and 6 with order 9→10→5→7→6 are assigned to machine 3.

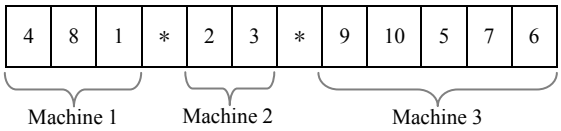

Figure 2. An example of solution representation

The elements of the proposed ASO algorithm are described in Subsections A and B. Subsection C presents the numerical study.

A. Movement policies and associated parameters

Here, the three movement policies are described and their associated parameters are set. We attempted to choose simple forms of the settings.

1) Movement policy based on current position: $\text{MP}_i^{\text{Current}}(k)$

For each member, the fickleness index is defined as

$$FI_i(k) = 1 - \frac{f(X_{i_k^*}(k))}{f(X_i(k))}$$

where the objective function f is the makespan of the decoded solution. This index is obtained from Equation (1) by setting $\alpha_i = 1$ for all members. We now consider the two following cases.

Case A: Member $i \neq i_k^*$

Based on $FI_i(k)$, generate the movement policy $\text{MP}_i^{\text{Current}}(k)$ as follows:

$$\mathrm{MP}_i^{\mathrm{Current}}(k)=\begin{cases}\text{Apply swap mutation operator} & 0\le FI_i(k)<0.5\\ \text{Apply insertion mutation operator} & 0.5\le FI_i(k)\le 1.\end{cases}$$

Case B: Member $i=i_k^*$

Randomly select one of the following movement policies:

a. 1-$\mathrm{CMP}_{i_k^*}^{\mathrm{Current}}(k)$:

- randomly choose a machine t
- randomly choose two jobs j_1 and j_2 for machine t
- swap jobs j_1 and j_2.

b. 2-$\mathrm{CMP}_{i_k^*}^{\mathrm{Current}}(k)$:

- randomly choose two machines t_1 and t_2
- randomly choose one job j_1 for machine t_1 and one job j_2 for machine t_2
- swap jobs j_1 and j_2.

c. 3-$\mathrm{CMP}_{i_k^*}^{\mathrm{Current}}(k)$:

- randomly choose one job j_1 and one machine t, where job j_1 does not belong to machine t
- choose randomly a valid position k in machine t
- transfer job j_1 to machine t at position k.

2) Movement policy based on other members: $\mathrm{MP}_i^{\mathrm{Society}}(k)$

We use the second scenario presented in Section II.B.2 to define $\mathrm{MP}_i^{\mathrm{Society}}(k)$ based on $EI_i(k)$ as follows:

$$\mathrm{MP}_i^{\mathrm{Society}}(k)=\begin{cases}\text{Apply two-point crossover operator with position of } i_k^* & 0\le EI_i(k)<0.25\\ \text{Apply uniform crossover operator with position of } i_k^* & 0.25\le EI_i(k)<0.5\\ \text{Apply two-point crossover operator with position of a randomly selected } i\ne i_k^* & 0.5\le EI_i(k)<0.75\\ \text{Apply uniform crossover operator with position of a randomly selected } i\ne i_k^* & 0.75\le EI_i(k)\le 1.\end{cases}$$

For each member, we define the external irregularity index $EI_i(k)$ as

$$EI_i(k)=1-e^{-CV(k)},$$

which is obtained from (4) by setting $\delta_i=1$ and $D(k)=CV(k)$.

3) Movement policy based on past positions: $\mathrm{MP}_i^{\mathrm{Past}}(k)$

For all members, the internal irregularity indices $II_i(k)$ are zeros, which means that members generate policies $\mathrm{MP}_i^{\mathrm{Past}}(k)$ based only on their own P-bests. The movement policy $\mathrm{MP}_i^{\mathrm{Past}}(k)$ is defined as

$\mathrm{MP}_i^{\mathrm{Past}}(k)=$ apply uniform crossover operator with P-best.

B. Combination rule

We can consider three combination rules:

1. elitism
2. sequential; $\mathrm{MP}^{\mathrm{Current}}\rightarrow\mathrm{MP}^{\mathrm{Past}}\rightarrow\mathrm{MP}^{\mathrm{Society}}$
3. crossover.

The numerical results show that, for the problem under consideration here, the best choice is the elitism rule.

C. Numerical results

We compare the proposed ASO algorithm with the following algorithms:

- the Simulated Annealing (SA) algorithm proposed in [4]
- the PSO algorithm proposed in [5]
- the hybridized Tabu Search (HTS) algorithm proposed in [6].

To the best of our knowledge, these algorithms are the best ones introduced in the literature for SDST-HFSP. All algorithms are coded in C++ and run with an Intel Pentium IV Core 2 Duo 2.5 GHz processor on 1024 MB RAM under a Windows operating system. The four algorithms are tested six times on 252 instances, which are considered in Kurz and Askin's article [7].

For each instance we run the four algorithms and then compute the *relative percentage deviation* (RPD) from the best solution for each algorithm. RPD for algorithm A (PSO, HTS, SA or ASO) is computed as follows:

$$\mathrm{RPD}=\frac{\mathrm{MS}_{\mathrm{A}}-\mathrm{MS}^*}{\mathrm{MS}^*}100\%$$

where MS_{A} is the average of the objective values of the solutions obtained by algorithm A in six runs, and MS^* is the best average objective value obtained by the four algorithms.

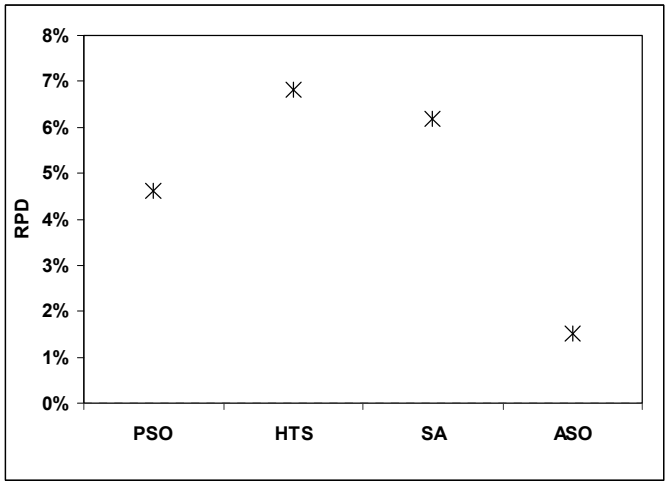

Figure 3. Averages of RPDs for four algorithms

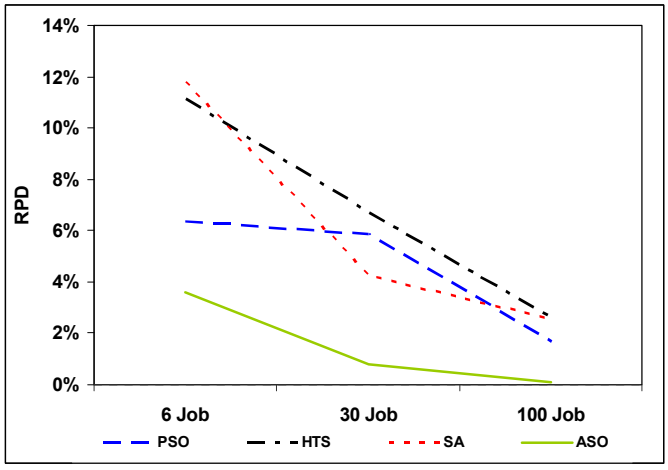

Figure 4. Averages of RPDs for four algorithms vs. different numbers of jobs

The averages of 252 RPDs for PSO, HTS, SA and ASO algorithms are depicted in Fig. 3, which demonstrates that the ASO algorithm is the most effective. In Fig. 4, we also show the averages of RPDs over the number of the jobs. These results prove that the ASO algorithm performs considerably better than the existing algorithms proposed in the literature.

V. Convergence Analysis

One of the important issues needs to be addressed is the convergence of ASO algorithms. Here we present a convergence property of ASO for combinatorial optimization problems, for which the search space S is finite. The result can be extended for continuous optimization problems.

Suppose the following assumptions:

A1. Search space S is finite.

A2. f is bounded over S.

A3. There is a member q in the society with the following characteristics.

i. The movement policy $\mathrm{MP}_q^{\mathrm{Current}}(k)$ is the neighboring method NM_q. $\mathrm{NM}_q(X)$ denotes the position generated by this method, and $N(X)$ is the set of all possible neighbor solutions of $X \in S$ by considering NM_q.

ii. The directed graph $G=(S,E)$ with

$$E=\{(X,Y): X\in S,\ Y\in N(X)\}$$

is connected. This means that every point in the solution space is reachable from any other point by a finite number of applying the neighboring method NM_q.

iii. The combination rule of this member is as follows:

1- Randomly generate r from uniform distribution on interval $(0,1)$.

2- Set $\lambda=\min\left\{1,\exp\left(\frac{-\Delta f}{T_k}\right)\right\}$ with

$$\Delta f = f(\mathrm{NM}_q(X(k)))-f(X(k))$$

$$T_k=\frac{\gamma}{\log(k+\alpha)}$$

for constants $\alpha\geq 2$ and $\gamma\geq R\times L$ where R is the radius of graph $G=(S,E)$, and L is the maximum local slope, i.e.,

$$L=\max_{X\in S,Y\in N(X)}\{f(Y)-f(X)\}.$$

3- If $r\leq\lambda$ move toward position $\mathrm{NM}_q(X)$; otherwise, stay in the current position and do not move.

Note the combination rule depends on iteration k, and uses only $\mathrm{MP}_q^{\mathrm{Current}}(k)$ to generate the position in the next iteration.

Theorem 1. Under assumptions A1-A3, with probability one an ASO algorithm converges to a globally optimal solution.

Proof. The proof follows from the convergence analysis presented in Theorem 5.1 of [8] for a special SA algorithm.

An interesting observation is that the above convergence result needs only a special member to exist in the society.

VI. Conclusion

This paper introduces Anarchic Society Optimization (ASO), which is inspired by an abnormal human society whose members behave anarchically and adventurously to improve

their situations. Our main contribution to the literature is to show that human societies may be a wide and good source of inspiration for designing new swarm-intelligence methods, instead of insects or animal social groupings.

We show that ASO includes PSO as a special case; therefore, it can be used successfully for continuous optimization problems. We also developed an efficient ASO algorithm for a challenging scheduling problem to illustrate how ASO can be applied to solve a discrete optimization problem by exploiting operators used in Genetic algorithms. These show that developing an ASO algorithm is easy for problems to which a PSO or Genetic algorithm has been applied.

Finally, we prove that an ASO algorithm converges to a globally optimal solution with probability one if the ASO population involves a member with specific characteristics. This shows that the convergence analysis of a swarm-intelligence algorithm does not necessarily depend on the characteristics of the entire society.

For future research, we suggest applying ASO to other problems, and providing convergence results that specify a time limit within which an ASO algorithm is guaranteed to converge with a given probability. Developing other human-inspired swarm-intelligence methods is another interesting research direction.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of the IEEE International Conference on Neural Networks (Perth, Australia) 1942–1948, 1995.

[2] R. Ruiz, J.A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, 205 (1): 1–18, 2010.

[3] A. Ahmadi-Javid, J. Behnamian, "An anarchic society optimization algorithm for the flowshop scheduling problem with sequence-dependent setup times," working paper.

[4] Z. Jina, Z. Yang, and T. Ito, "Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem," *International Journal of Production Economics*, 100 (2): 322–334, 2006.

[5] C. T. Tseng and C. J. Liao, "A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *International journal of production research*, 46(17): 4655–4670, 2008.

[6] X. Wang and L. Tang, "A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers," *Computers & Operations Research*, 36 (3): 907–918, 2009.

[7] M. E. Kurz and R. G. Askin, "Scheduling flexible flow lines with sequence-dependent setup times," *European Journal of Operational Research*, 159 (1): 66–82, 2004.

[8] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Advances in Applied Probability*, 18(3): 747-771, 1986.