COGNITIVE MODELS FOR LEARNING TO CONTROL DYNAMIC SYSTEMS

Russ Eberhart, Xiaohui Hu, Yaobin Chen

May 2008

Final Report

This work was funed by in whole or in part by Department of Air Force contract FA9550-07-C-0167. The U.S. Government has for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government

AF OFFICE OF SCIENTIFIC RESEARCH
875 NORTH RANDOLPH STREET, RM 3112
ARLINGTON VA 22203

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
05/30/2008	Final	09/30/2007-05/30/2008

4. TITLE AND SUBTITLE
Cognitive models for learning to control dynamic systems

5a. CONTRACT NUMBER
FA9550-07-C-0167

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Eberhart, Russell C.
Hu, Xiaohui
Chen, Yaobin

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
COMPUTELLIGENCE
212 W 10TH ST STE B-315
INDIANAPOLIS IN 46202-5401

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF OFFICE OF SCIENTIFIC RESEARCH/NL
875 NORTH RANDOLPH STREET, RM 3112
ARLINGTON VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-SR-AR-TR-08-0310

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution A: Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Report developed under STTR contract for topic “Cognitive models for learning to control dynamic systems" demonstrated a swarm intelligence learning algorithm and its application in unmanned aerial vehicle (UAV) mission planning. A new UAV assignment model was developed that reduces the dimension of the solution space and is easily adapted by computational intelligence algorithms. A version of particle swarm optimization (PSO) was applied to accomplish the mission optimization. Numerical experimental results illustrate that it efficiently achieves the optima and demonstrates the effectiveness of combining the model and PSO to solve complex UAV assignment problems. The time to complete mission plans for operationally realistic scenarios is reduced by 3-4 orders of magnitude compared with the mixed-integer linear programming approach being used by AFRL at WPAFB. A computer game was also developed to investigate how humans interact with swarm intelligence. The game is based on an NK landscape. It is concluded that the combination of a human-swarm team may have advantages in certain environments, such as dynamic decision making tasks.

15. SUBJECT TERMS
Unmanned aerial vehicle, UAV, mission planning, particle swarm optimization, PSO, swarm intelligence, NK landscape game

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Russ Eberhart
U	U	U	UU	79	19b. TELEPHONE NUMBER *(Include area code)* (317) 453-2120

Table of Contents

List of Figures

List of Tables

1. Executive Summary

Dynamic systems are comprised of the following main features: a series of control signals to achieve an optimum, a system that changes as a consequence of the control signals, and other internal/external factors. Thus the control signals are not independent, so that subsequent signals depend on earlier ones. The control signals can either be continuous or be a series of discrete actions. Real world examples include navigational control, battlefield decisions, logistics planning, *etc*.

The goal of analyzing dynamic systems is to develop the ability to characterize all possible trajectories of the system given all possible initial conditions. A state machine is one possible model of a dynamic system. The state machine comprises the set of all possible states of the system, where each state represents all relevant information at an instant in time, and all allowable state transitions that specify how states are updated each time step.

The systems for which we are developing cognitive-based model controllers are typically highly non-linear time-varying systems. Various approaches have been studied and various algorithms have been developed to solve these dynamic control problems, including stochastic (non-deterministic) and chaotic (deterministic) methods.

Our methodology focuses on swarm intelligence concepts, paradigms, algorithms, and implementations.

In the Phase I STTR work, the concept of swarm-powered optimization and planning systems for significantly improved unmanned aerial vehicle (UAV) mission planning and optimization was demonstrated. Feasibility was demonstrated for swarm intelligence based dynamic learning algorithms. Major accomplishments include:

a. Swarm intelligence based mission scheduling systems for multiple-UAV, multiple-task, multiple-target scenarios has been studied. The swarm-based approach is able to schedule optimal mission configurations two to four orders of magnitude faster than the mixed integer linear programming approach currently used by the Air Force Research Laboratory

b. A human-swarm hybrid model has been established and demonstrated.

c. Initial work on swarm diversity measurements has been finished and a group of diversity metrics have been investigated.

2. Introduction and Literature Review

2.1. Background

2.1.1. Machine learning and swarm intelligence

Since the dawn of the history, humans have tried to duplicate functions of the brain, or intelligence in an artificial form. Nowadays, the field is typically called artificial intelligence or machine learning. Traditional artificial intelligence researchers have focused more on individual mind processes or symbolic processes. A recent book "Swarm Intelligence" co-authored by the PI of this proposal, described a different point of view of "intelligence" [1], which is a result of social interaction rather than being a function and process of mind/brain. Our work rests upon and is inspired by the culture and cognition foundation built by Latané, Axelrod, and Kennedy.

Latané's dynamic social impact theory states that the behaviors of individuals can be explained in terms of the self-organizing properties of their social system, that clusters of individuals develop similar beliefs, and that subpopulations tend to diverge from one another. A summary of this theory is that individuals influence one another and in doing so become more similar, and that patterns of belief held by individuals tend to correlate within regions of a population. This model is consistent with findings in the fields of social psychology, sociology, economics, and anthropology [1].

Axelrod's culture model represents populations of individuals as strings of symbols, or features. The probability of interaction between two individuals is a function of their similarity, and individuals become more similar as a result of interactions. The observed dynamic in this model is *polarization*, homogeneous subpopulations that differ from one another [1].

Kennedy's adaptive culture model states that there is no effect of similarity on the probability of interaction, and that the effect of similarity is negative, in that it is dissimilarity that creates boundaries between cultural regions. Interaction between individuals occurs if *fitnesses* are different [1].

According to those social and cognition models, people adapt to complex environments by evaluating, comparing, and imitating one another, learning from experience and emulating the successful behaviors of others. The social interaction results in the development of the inherent intelligence of humans [1].

In summary, swarm intelligence is defined as the collective behavior of relatively simple individuals interacting locally with their environment which causes coherent functional global patterns to emerge.

2.1.2. Particle swarm optimization

Particle swarm optimization (PSO) is a swarm intelligence technique for optimizing nonlinear functions developed by Jim Kennedy (project advisor) and Russell Eberhart (project PI) [2, 3]. It was initially developed as a social psychological model that simulates behaviors of bird flocking. Later it was found that PSO was very effective as an optimization method and it has been applied in numerous applications since then [2, 3].

The particle swarm population is initialized by assigning random positions and velocities to each population member (particle). Potential solutions are then flown through hyperspace (the

problem space). Each particle keeps track of its "best" (highest fitness) position in hyperspace. This is called *pbest* for each individual particle, *gbest* for the best in the population (when a global model is used), and *lbest* for the best in a defined neighborhood (when a neighborhood model is used). At each time step, each particle is stochastically accelerated toward its *pbest* and *gbest* if the global model is used, or *pbest* and *lbest* if a neighborhood model is used. The neighborhood comprises the particle and a selected number of its topological neighbors. For example, a neighborhood of five includes the particle and its two topological neighbors on each side.

The particle swarm optimization process is as follows:

1. Initialize the population in hyperspace.
2. Evaluate the fitness of individual particles.
3. Modify the velocity of each particle based on its *pbest* and *gbest* (or *lbest*).
4. Terminate on some condition.
5. Go to step 2.

The velocity *vid* and position *xid* update equations for the global version of particle swarm optimization are

$$v_{id} = w_i v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

where *d* is the dimension, c_1 and c_2 are positive constants, *rand()* and *Rand()* are random functions, and *w* is the inertia weight [2, 3]. p_{id} is the *pbest* while p_{gd} is the *gbest*. (For the neighborhood version, change p_{gd} to p_{ld}, which is *lbest*)

The performance of each particle is measured according to a predefined fitness function. The inertia weight influences the tradeoff between global and local exploration. A common approach is to reduce the inertia weight during a run (*i.e.* from 0.9 to 0.4 over 1000 iterations), to set c_1 and c_2 to about 1.5 or 2.0, and to set the maximum allowable velocity *Vmax* to the dynamic range of each variable (on each dimension).

2.2. Dealing with dynamic environments with particle swarm

Many real-world systems involve dynamic environments that change state frequently; some change almost continuously. Prices fluctuating, route programming, target recognition, schedule changes, investment portfolio evaluations, data mining, etc., all pose dynamic optimization challenges for computation intelligence methods. Meanwhile, the requirement is no longer to locate optima in the search space, but to track their progression through the space as closely as possible.

2.2.1. Improvement of PSO for dynamic environments

The earliest research work with evolutionary computation methods in a dynamic environment can be traced back to 1966 [4]. Since year 2000, initial work on tracking dynamic systems with particle swarm optimization was reported by Carlisle and Dozier [5], and Eberhart and Shi [6].

Carlisle applied several versions of sentry PSO to dynamic problems [5, 7, 8]. “Sentry” is a selected point or a batch point, which is fixed or randomly picked in the search space, and used

to detect dynamic changes. Periodic and triggered replacements of best-known positions with current positions are implemented. The dynamic PSO by Carlisle [8] was applied to fuse information [9].

Based on an analogy of electrostatic energy, Blackwell [10, 11] proposed a charged PSO (CPSO), in which particles are given an added an extra acceleration term to experience repulsive effects from all other charged particles, maintain diversity and deal with changing dynamic environments. Moreover, another extension is called atomic PSO where half of the particles have the extra charge acceleration term and the other half are neutral. After adding a repulsion function to make balancing diversity, Jatmiko [12] applied it to an odor source localization task.

Multi-population evolutionary algorithms, like Self Organizing Scouts [13], are proposed to locate multiple peaks in a multi-modal landscape, and enhance search in a dynamic landscape. These algorithms use a number of smaller sub-populations to watch over the most promising peaks, and others for further exploration. Using the multi-population concept, multi-swarm PSO, speciation-based PSO etc., the swarm approach has been adapted to track multiple optima simultaneously with multiple swarms

Blackwell [14] designed multi-charged PSO and multi-quantum swarms. As a multi-population version of CPSO, a CPSO swarm is placed on each local optimum in a multi-modal environment. Those neutral sub-swarms keep searching and the surrounding charged particles maintain enough diversity to track dynamic changes. In the same paper, based on a quantum rather than classical picture of an atom, multi-quantum swarm optimization (multi-QSO) uses a quantum analogy for the dynamics of the charged particles. The positions of quantum particles are determined by a probability function centered on the swarm attractor, so particle-to-particle comparisons are thus avoided. The quantum atom has the advantages of lower complexity and an easily controllable distribution.

As a further elaborated work, the multi-swarm was integrated with the three diversity operators [15] as follows: Quantum particles (a diversity preserving technique, used in [14] before); Exclusion (a local interaction between colliding swarms, preventing swarms from settling on the same peak); Anti-convergence (a new operator, sharing information among swarm with the aim of allowing new peaks to be detected).

Another parallel sub-population model motivated by the multi-population idea is species-based PSO (SPSO), proposed by Parrott and Li [16]. According to their similarities, particles are adaptively divided into species at every iteration. Each species is grouped around a “seed”, a dominating particle used for speciation. And *P*max, a maximum number of particles allowed for a species, is set as a mechanism to avoid crowding. The model employs these mechanisms to encourage simultaneous tracking of multiple peaks by preventing overcrowding at peaks. Furthermore, Parrott [17] proposed an improved version of SPSO with a mechanism for removing redundant duplicate particles in a species for better performance efficiency. Li designed an extension of a speciation-based PSO [18], combined with several techniques: *P*max [16]; "neutral" and "quantum"" concepts of a quantum swarm model [14] to maintain a better diversity; a diversity threshold which triggers the anti-convergence[15] replacement process to the worst species; and, a particle diversification threshold within a species to judge particle relocation operation. Developing parallel species mechanism and improvement techniques encourage swarms to converge onto multiple local optima instead of a single global optimum.

Recently, Lung [19] proposed a hierarchical sub-population PSO, named Collaborative Evolutionary-Swarm Optimization (CESO), which consists of two populations of equal size. One is motivated by an evolutionary algorithm's diversity preservation mechanism and responsible for preserving diversity in order to prevent premature convergence, and the other tracks the global optimum.

The P-Best Decay PSO (PBDPSO) [20] is designed to solve a dynamic pricing problem. When response is triggered by any of the various changes, each personal fitness value is decayed by multiplying it by a decay rate. Similar to PBDPSO, Tracking Dynamic PSO (TDPSO) [21] , uses an evaporation constant, to evaporate (decrease) the personal fitness value and global fitness value over time. Both of them renew their memory without any centralized control and to maintain simplicity of each particle.

2.2.2. Environment change detection

Early research work on PSO has shown that PSO is effective for optimization problems in both static and dynamic environments. Fast converging to an optimum implies diversity loss that will make swarm easier to stagnate at a local extremum. For a classic PSO in dynamic environments, it is difficult to effectively respond to outside changes and restart exploration without enough population diversity.

So, one starting point is maintaining the swarm diversity and adapting particles to a landscape change throughout the entire run of the algorithm. The main advantage of such an approach is that it does not need a special operation to detect the environment changes [10, 21]. This approach is appropriate for continuous small changes in the objective function.

Another group of methods don't use detection operations. In [5], a response action will be triggered after a certain period of iterations, which means it assumes that a change has happened at that time. The *gbest* and second *gbest* value are monitored to check if they have no changes for 20 iterations in [22]. Similar to this, monitoring the *gbest* and 20 iteration intervals are adopted in [12].

Some techniques are designed to measure the changes and trigger a response. Most of them assume that the environment changes are known, or at least measurable and detectable, *e.g*., by a reevaluation of the objective function. Evaluating one or several points in the landscape and comparing with the previous fitness value is a popular way to test changes. Carlisle[5] selected a random point in the search space and only if its fitness change exceeds a setting allowable range, it will trigger a response operation. A sentry particle [8] is also a fixed point, randomly initialized, and is compared with its previous fitness at each iteration, which is also applied in [9]. Instead of a randomly selected point, *gbest* is used in [22]. Combined with [5] and [22], Mullen [20] used *gbest* as a detection point and a threshold for considering noise existing in a dynamic pricing problem. Blackwell [15] chooses the quantum particles of every swarm as the detected point. The best individual in the CRDE population, which is the first population of Collaborative Evolutionary-Swarm Optimization and is used to maintain diversity, is re-evaluated to detect the dynamic change [19].

Bird [23] proposed to monitor the five fittest particles for a standard constriction PSO with a von Neumann neighborhood model and record the *pbest* of the fittest particle in each of the top five species for species-based PSO, which is also adopted by [18]. Every iteration, reevaluating each particle's *pbest* fitness value at its recorded position is also a detection strategy for a species-

based PSO by Parrott [17], however, it does result in additional cost of fitness function evaluation.

2.2.3. Response strategy

Like any other computational intelligence methods (*e.g*. EA, EP, GA, etc.), the performance of PSO highly relies on the balance between exploration and exploitation, which is influenced by the swarm diversity. During optimization and evolution, particles follow the lead particle to search a target and tend to converge, so it is difficult for a swarm, losing diversity, to make a further exploration in the search space without any effective strategy for enhancing diversity. Especially in a dynamic optimization, the tracking behavior of a population with a low diversity is less desirable. Generally, there are two type mechanisms to deal with the diversity problem in dynamic environment: reactive and proactive [24]. Moreover, some algorithms combine the two mechanisms, such as some species-based PSO. So, we will review of PSO response strategies to dynamic changes from three aspects: Proactive strategy, Reactive Strategy and Hybrid Strategy.

Proactive strategy

The proactive strategy, in this report defined as the mechanism not based on the detection of dynamic change, prevents swarms from converging too fast and maintains the swarm diversity. It can be achieved by decreasing the selection pressure, diminishing outdated memory, periodically stimulating diversity or constructing multi-population swarms, etc. PSO algorithms based on the proactive strategy can adapt to a dynamic optimum, especially with small changes. However, by maintaining the diversity, a swarm can't converge as fast and be very close to a changing optimum.

When dynamic change happens, the swarm's memory (gBest and pBest) becomes outdated, which will misguide other particles. "Reset" means that particles' pBest and gBest have been cleared and set to new re-evaluated fitness value. It is one of the most popular mechanisms to deal with outdated information and diversity loss. For example, particles' personal best vectors are replaced with their current position vector on a regular frequency based on the iteration to forget their experience [5, 17, 18, 23, 25]. In [12], one strategy is designed to restart the *gbest* fitness value at the initial value (*gbest*=0) in order to make all particles jump out to follow the changes. It implies loss of information gathered during the search so far. Another strategy in this reference is to use charged PSO and add a repulsion function to balance diversity. Another method is to periodically re-randomize part of the swarm or gbest particle to stimulate diversity [5, 22].

The fundamental idea of a multi-swarm approach is to divide a swarm into a number of sub-swarms. It is an effective approach to preserve diversity. Each swarm watches different promising peaks of the landscape separately. Since all the particles don't converge together, when environment is changed, it can still have a search ability in some sense. From this perspective, a multi-swarm mechanism is a kind of proactive method. Without a detection action, a charged particle is used to produce the repulsive effects from all other charged particles for both charged swarm PSO or atomic swarm PSO [10]. Also, without a special method to deal with dynamic changes, the species-based PSO [16] adjusts the dynamic speciation by randomly reinitializing the extra particles of a speciation. In [14], it re-randomizes the worst swarm within a specified radius of each swarm for multi-charged PSO.

In addition to a multi-population structure to maintain diversity, multi-swarm PSO also works in conjunction with other improvement approaches, which are often reactive type operations based on change detection or proactive methods. So, this is regarded as a hybrid approach of proactive and active actions, and will be discussed in the third part of this section.

Reactive strategy

Reactive strategy is defined as the response mechanism based on the detection of dynamic change. Before a new change is detected, particles concentrate on searching the current optimum. Only when alerted by a signal will it respond. Its main disadvantage is the additional cost of fitness function evaluation because of the change detection procedure. Such an approach is appropriate for discontinuous changes.

"Reset" has been widely adopted to reactive response mechanisms. All particles reset their pbest values to their current position vectors by checking with defined threshold [20]. Instead of simply replacing all *pbest*s, an improvement of "reset" is to adopt re-evaluation step for *pbest* and replace those particles, only if their current location is better than *pbest* [8, 9]. It doesn't completely deny the value of a *pbest* particle's experience. Eberhart proposed that each *pbest* fitness value be updated in the changed environment and its position be retained [6].

Hu [22] also tested particle randomization as an active response method. The difficulty is to determine the randomized proportion of swarm, large proportion randomization loses too much previous information and is similar to restarting a new optimization, however, a small proportion being re-randomized cannot stimulate enough diversity. So experiments demonstrate that 10 percent re-randomization rate is a good choice for most cases. This method is also applied in [25], to compare PSO algorithms for tracking extrema in dynamic environments.

Instead of discarding all previous experience, the position value of each pbest of the P-Best Decay PSO [20] is decayed by multiplying a decay rate. This allows the particles to simultaneously make use of previous information while discounting possibly noisy or dynamic data.

Hybrid strategy

Hybrid strategy refers to those PSO algorithms that utilize mechanisms of both proactive and active strategies. Generally, each of them has its own advantages and disadvantages. For example, proactive approaches always converge slowly and not very close to a changing optimum, however active approaches result in extra computation cost. Designing improvements that have a fast convergence before new changes and maintaining a diversity so as to adapt to dynamic landscapes, is crucial.

Collaborative Evolutionary-Swarm Optimization (CESO) consists of two populations: SWARM (PSO)and CRDE (Crowding Differential Evolution). One of the populations is responsible for preserving diversity of the search and the other one tracks the global optimum. In case a change appears in the environment or if the distance between *cbest* and *gbest* is very small, SWARM will be reset by CRDE individuals [19] to enhance the diversity of SWARM.

If a change in the environment is detected, all particles' *pbest* are re-evaluated/re-initialized [15], which depends on the swarm status: Quantum particles, Exclusion, Anti-convergence. All of them are designed to control diversity separately. Exclude reinitializes the worst swarm of any two sub-swarms close to each other. Anti-convergence expels the worst swarm from its peak and reinitializes it in the search space. This can make sure that at least one swarm keeps searching.

2.2.4. Performance metric

Performance metrics are important to compare different algorithms and provide measurements for different strategies and approaches.

(1) Fitness Performance Metric

Some traditional measures of EA performance (uch as offline performance, online performance, time to convergence, and best-so-far curves, etc.) are still adopted. Any new metrics and modifications have been designed to compare performance for dynamic functions.

The **Best-so-far** curves are used to record and graph the best values which stand for the error of the optimum at each generation. Some researches argue that the curves are inappropriate, since after a landscape change, the previously fittest value may not be valid [26]. Nevertheless, as far as making an extension with respect to the new optimum and re-evaluating particles to select the fittest individual, extended best-so-far is still a simple and common metric. For instance, the real-time distance of the best particle from the movement beacon [5], in fact it is the same as the distance of the swarm's global best position at each iteration from the updated solution at that iteration [6, 8-10].

A curve of extended best-so-far is generally saw-tooth shape, which is simple, distinct and much easier to interpret, but lacks a quantitative description for the overall performance comparison.

The **Offline error** metric is the most popular and effective way to compare overall performance of algorithms. It computes an average of best the performance of individuals over the number of cycles. Branke and Schmeck introduce the offline error [10, 27] as a performance measure of dynamic optimization problems, e.g. used in species-PSO [18] and mCPSO, mQSO [15] for the moving peak benchmark problem. The cumulative mean revenue [20], based on the off-line error, is an average of the total revenue earned at a certain time. The normalized minimum error for each iteration and average minimum error over a run are used in [16]. Furthermore this is also used in [17], named global best offline error. Two of the offline errors to measure the performance are calculated as follows. The global best offline error is computed by the recorded best global error of every particle with respect of the global optimum since last change; the average local offline error is computed by the best average of best “local” errors since the last environment change across all visible optima.

However, it is pointed out that the offline error metric provides less measurement for exploration and exploitation characteristics of algorithms. The offline error is split into two orthogonal parts [23]. One is Best Known Peak Error (BKPE), which is used to measure convergence speed and another is Peak Cover, which is responsible for analyzing the optima or peaks quantity and also measuring population diversity and exploration status. The metric offers new insight into the convergence speed and diversity of an algorithm.

(2) Statistic Performance Analysis

Dozens of runs are usually executed for every experiment, so that the statistical performance is able to analyzed and reduce the variability of results, e.g., the average iterations needed to track the dynamic optimum, or the success rate to achieve a task under a certain time. It is essential to provide the statistical properties for all random-based optimization algorithms.

In [5], three variants are computed: the reliability of the algorithm in terms of solutions found, the efficiency in terms of average iterations per solution found, and the median iterations per run.

This gives a quick indication of the distribution of the solutions and comparison by considering variations in target speed and swarm topology.

Some simple ways exist to average a performance over dozens of runs. One is to average the extended best-so-far performance [25], such that each version PSO is run for 1000 generations and 50 runs. The offline error is averaged after 50 runs [18, 19] to test the effect of different parameters settings. The average of the cumulative mean revenue [20] is computed with considering several version PSO with different detection and response methods. The global best offline error, min, max errors and standard deviation for all the runs are recorded and averaged after 50 runs in [16].

In [22], it is indicated that there are two kinds of performance indexes considered. The first is the average number of iterations that PSO used to find the optimum. The second is the average number of iterations PSO required to follow the dynamic changes Both are used considering the difference of detection methods, changing severity and response strategy (different randomization rates). The second one also is adopted in [12].

There are many metrics, widely adopted by other EA algorithms, but not used often by PSO, e.g. the Collective Means fitness metrics, proposed by Morrison [26]. There are two average versions of it, one is the average of the best individual over sufficient generations, and another is further average over multiple runs. The strong point is it could be used for comparisons between algorithms, since it will give a stable performance value after a large number of runs. Applying this metric to PSO has not appeared in any research paper.

Other metrics include, mean tracking error, on-line performance, absolute performance, etc.

2.2.5. Experimental design

A real-world problem always has dynamic properties, characterized as changes in objectives, priorities, and/or resources. It is necessary to design algorithms, which can operate in a dynamic system, furthermore, abstract the real-world problem and design a fitness function. A real-world problem, transformed into a mathematical world, may be studied more conveniently. And those dynamic characteristics can be described, using the mathematic language. A dynamic problem consists of finding for each time moment *t,* the value *x**(*t*) which satisfied [28]:

$\tilde{f}(t,(x^*(t)) \geq f(t,x(t))$ for all $x \in R^n$ (for a maximum problem). Several characteristics of dynamic problems [29] should be considered, including frequency of change, severity of change, predictability of change and cycle length / cycle accuracy.

Dynamic PSO [9] is applied to an optimal fusion configuration of sensors in distributed detection network in a non-stationary binary symmetric channel. The main objective is to maximize probability of detection and minimize the probability of false alarms. The Dynamic pricing problem, a real-time machine learning problem with scarce prior data and a concrete learning cost, is depict by Kalyanam's log-linear demand model and studied by the P-Best Decay PSO[20]. The odor source localization problem in a dynamic environment with the odor Gaussian distribution model and the advanced turbulence model, is studied in the simulation optimization experiment [12]. A particle swarm model for swarm-based networked sensor systems is also applied to the Joint Battlespace Infosphere (JBI) [30].

Like benchmark functions in static environment (Sphere, Rosenbrock, Rastrigrin, and Griewank, etc,), it is necessary to construct general benchmark dynamic optimization problems. In 1999,

Branke [31] proposed Moving Peaks benchmark function and Morrison and De Jong [32] proposed DF1. There are several ways in which the benchmark problems can change over time [6, 32]:

Case 1. The simplest dynamic problems, the overall shape (morphology) of the fitness landscape is constant, the optimum location in problem space can move with a speed, randomly or linearly or periodically, etc. Tracking a beacon movement [5] at some constant velocity diagonally through the search space is a linear motion in this case. The simple parabolic function is special situation of this case. A 3-dim parabolic function is adopted as a test function with linear, circular and random movement in [7] and a 10-dim one with linear movement is used in [6, 22]. And a randomly moving optimal of the parabolic function also was used in [21]. Three types of dynamic optimum changes of DF1: small constant step changes, different large step changes, and chaotic step changes are used by Xiaodong [25] to compare several version PSO. A 5-dim and 5-cone landscape moving every 10 generations was studied in [26] for a performance measurement research.

Case 2. More complicated, the location remains constant, but landscape's morphology changes, including Peak heights, width, and shape. A species-based PSO [17] was run in 2-D environment dynamism with the height and shape changing periodically.

Case 3. Both will change. Third both the location of the optimum and landscape's morphology can vary. In a multidimensional system, these variations can occur on one or more dimensions, either independently or simultaneously. E.g. the plume changes randomly according the wind speed and wind direction, in [12]. The Moving Peaks Scenario 2 benchmark function is a typical experiment of case 3, which may has a settings of 5-dim and 10 peaks and a change of peaks' position, height, width in 5000 evaluations and the peak movements are uncorrelated. It was used to the performance metrics [23] research with different numbers of peaks from 1 to 190, and by an extended speciation-based particle swarm optimizer [18] and multi-Charged PSO, multi-Quantum Swarm PSO [15]. Collaborative Evolutionary-Swarm Optimization (CESO) [19] also use it as a benchmark experiment to compare with the Self Organizing Scouts (SOS), the Multi-swarm (MPSO) methods, the Particle Swarm with Speciation and Adaptation (SPSO). A parallel sub-population PSO with speciation and crowding mechanism [16] was applied to 2-dimension,3-peaks DF1 problem with peaks shifting position spatially, changing shape and weight periodically.

Besides the applications and benchmark functions mentioned above, other popular benchmark problems include: dynamic knapsack problem, dynamic bit-matching, scheduling with new jobs arriving over time, greenhouse control problem, etc.

2.3. Handling multiple-objectives with particle swarm

Most realistic optimization problems, particularly those in design, require the simultaneous optimization of more than one objective function. So the multi-objective optimization problem (MOP) is a class widespread optimization problem in real world, which contains several or some hard-solving characteristics, including high-dimensional, discontinuous, non-convex, multimodal, and/or NP-complete, etc. It arise a challenge for those deterministic methods, e.g. greedy and hill-climbing algorithms, branch and bound tree/graph search techniques, depth-and breadth-first search, best-first search, and calculus-based methods.

The earliest potential research of applying EA for MOP could be almost traced back to the late 1960s[33]. In Rosenberg's Ph.D. thesis, his research suggestion would have brought out the multi-objective optimization if it was carried out as mentioned. Not until the mid-1980s, the actual implementation of a multi-objective evolutionary algorithm was presented by Schaffer who proposed the Vector Evolution Genetic Algorithm (VEGA) for solving machine learning problems. Moore [34] firstly applied PSO to two-objectives MOP in 1999.

2.3.1. Basic concept

Multi-Objective Problem

In mathematical terms, the multi-objective problem can be described as:

$$\min_{x \in C} F(x) = [f_1(x), f_2(x), ..., f_m(x)] \tag{2-1}$$

where $\vec{x} = [x_1, x_2, ..., x_n]$ is the vector of feasible solution or decision variables. $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2..., n$ is the i-*th* objective function, $g(\cdot)$ and $h(\cdot)$ are the inequality and equality constrains, respectively. Instead of a single optimum, the solutions of MOP are always a solution set (the so-called Pareto optimal set), due to conflict between the objective functions.

Pareto Optimality

Pareto optimality, named after Vilfredo Pareto who used the concept in his studies of economic efficiency and income distribution, is an important concept in economics with broad applications in game theory, engineering and the social sciences. The concept of Pareto optimality is introduced in MOP that a vector $\vec{x}^* \in C$ in Equation 2-1 is said to be Pareto optimal for MOP, if all other vectors have a higher value for at least one of the objective functions, or else have the same value for all objectives. A few definitions are as below [35]:

Def. *Dominates*: Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$, it is said that $\vec{x} \le \vec{y}$ if $x_i \le y_i$ for $i = 1, \ldots k$ and that $\vec{x}$ **dominates** $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \le \vec{y}$ and $\vec{x} \ne \vec{y}$.

Def. *nondominated*: A vector of decision variables $\vec{x} \in \chi \subset \mathbb{R}^n$ is **nondominated** with respect to χ, if there does exist another $\vec{x}' \in \chi$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$

Def. Pareto-optimal: A vector of decision variables $\vec{x} \in F \subset \mathbb{R}^n$ (F is the feasible region) is **Pareto-optimal**, if it is nondominated with respect to F.

Definition Pareto Optimal Set: P^* is defined by: $P^* = \{\vec{x} \in F \mid \vec{x} \text{ is Pareto-optimal}\}$.

Def. Pareto Front: PF^* is defined by: $PF^* = \{\vec{f}(\vec{x}) \in \mathbb{R}^m \mid \vec{x} \in P^*\}$

Pareto optimal points are also known as efficient, non-dominated, or non-inferior points. The shape of Pareto front indicates the nature of the tradeoff between different objectives.

MOP requirement for PSO

The information sharing mechanism in POS is significantly different with GA. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only *gbest* (or *lbest*, *nbest*) provide the information with

others. It is blackboard information sharing model and one-way mechanism. All the particles tend to converge to the best solution quickly even in the local version[36].

approaches to guarantee that the solutions obtained are not only the Pareto optimal, but are also uniformly distributed in the Pareto front[34, 37-39].

2.3.2. Nondominated solution archive and best particle selection

In the case of a single objective optimization problem, *pbest* is the better one between the particle current position and previous personal best position; *gbest* is best position of all particles in the swarm and can be selected from the *pbest* set; the optimum is a point or vector in the solution space. However, the situation is changed for a multi-objective problem, the optimum solution is not a point, but a Pareto optimal set; the *gbest* have to be selected form several nondominated particles; the relationship between *pbest* and the current particles is no more a better or not, but dominate, nondominated or neither. And all non-dominated solutions are equally good. Thus, how to save the nondominated solution found by particles and how to select the best particle from the solution archive is the unique research work of PSO in multi-objective problems.

Nondominated solution archive

It is necessary to save all the found nondominated solutions in an archive, which may be internal or external. The internal archive means those nondominated solutions are reserved in the swarm, such as *pbest* set and *gbest*, without introducing other data set. In contrary, the external archive will utilize some data set to maintain search results. The internal archive is exactly a fixed size. Only a limited few papers reported adopting an external archive with an adaptive size. Most external archives are also fixed size. In term of a fixed archive, when the archive is full, some nondominated solutions have to be discarded, based on the density distribution, selection pressure, or clustering.

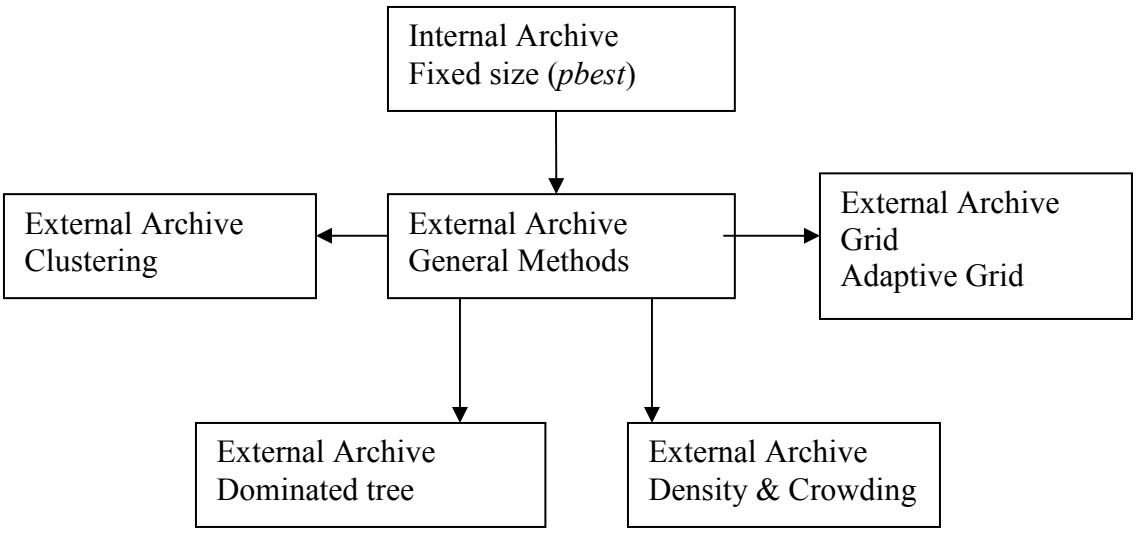

Figure 1 Nondominated solution archive

(1) Internal repository

The nondominated solutions found by particles are stored in the *pbest* set or a set with the equal size of the swarm size. It means that the number of non-dominated solutions is directly associated with the population size. So a larger population size is always preferred.

In [36], the earliest studies of PSO in MOP, Moore applied *p*-list (*pbest*) to keep track of all Pareto nondominated solution that a particle encountered as it explored the search space, so as to adapt PSO for multi-objective optimization. Hu[40] stored all the potential multiple optimal solution in the *pbest* with a large size swarm and find the nondominated solutions from the *pbest* set. In a further modification to significantly reduce the computation time, he introduced a external archive to take place the pbest to memorize the Pareto optimum solution[41, 42].

(2) External Archive

The historical record of best solutions, a set used to store nondominated solutions, is always called as repository, external archive or extended memory, etc. The archive may be fixed or adaptive size. The nondominated solutions should be inserted in the archive and any dominated solutions are eliminated from the repository.

□ ***General methods.*** Implementation of the external archive is set an external memory with a bounded size. There exits a tradeoff between the computation cost and storage capacity of nondominated solutions. The simplest way could customize an archive large enough and adopt elitist policy to update the archive [43]. e.g. Simple Multi-objective PSO (SMOPSO)[39]. In [44], an extended memory is applied to store all the Pareto optimal solutions, including those neglected solutions which do not dominate the current particle, but dominate other particles.

□ ***Cluster related methods.*** Mostaghim[45] set a fixed size archive to retain the nondominated solutions. The fitness space is divided into subspaces and every particle will find a nondominated solution with minimal σ distance in the archive. Clustering is applied on the elite particle in the archive, in the case the maximum archive size is reach and the extra particles will be moved. Hsieh[46] also introduced the clustering method to construct the external archive. In the further work, Mostaghim[47] added a multi-level subdivision scheme iterative division of the search space into subspaces (boxes). There is a restricted archive size at initial run and no restriction at the covering stage.

Instead of using one archive for gbest selection, Abido[48] employed two archives for both *gbest* and *pbest*. Up to the current time, the nondominated solutions obtained by all particles are stored in Nondominated global set and the nondominated solution obtained by personal particle is stored in nondominated local set, the hierarchical clustering algorithm used by SPEA is employed to guarantee the ember of the individuals stored in the set does not exceed the maximum size by means of an average linkage. For all the two sets, if the member of the individuals stored in the set exceeds the maximum size, reduce the set by means of an average linkage based. Zhang[49] also employed two sets to record pbest and gbest, respectively. The management of the sets included two stages. During initial stage of the evolution, when the cardinality of pbest set exceeds the given threshold, the nearest particle to the current one is deleted in order to explore more potential area. In contrast, during the final stage, the nearest one is preserved to enhance the process of convergence.

□ ***Density related methods.*** Praveen[50] also adopted a fixed size archive, but employed a sorting scheme based on ε-fuzzy dominance value. The external archive is used to store the top

best N nondominated solutions found so far from the solution in the current iteration and previous generation, based on the fuzzy dominance values.

LI [51]proposed a non-dominated sorting PSO combined with two parameter-free niching methods. All particles' personal bests and their offspring are compared and sorted together in the entire population, that will ensure more non-dominated solutions can be discovered and provide an appropriate selection pressure to guide the swarm population towards the Pareto-optimal front.

Carlo[52] incorporates the mechanism of crowding distance with Multi-objective PSO (MOPSO-CD). A fixed external archive is used to store nondominated solutions found. All new nondominated solutions are inserted into the archive, if they are not dominated by any of the stored solutions. And those dominated solution in the archive have to be deleted from the archive. If maximum size is reached, the most crowded particles will be removed. All nondominated solutions in the archive will be sorted in descending order in term of the calculated crowding distance value. And then a particle at the specified bottom portion will be randomly selected and removed.

□ ***Dominated tree methods.*** In order to find a better way to select a *gbest* than density based selection methods, Fieldsend[53, 54] adopted a new data structure, named dominated tree, into the external archive. The nondominated global solutions are stored in an unconstraint external archive and organized as a dominated tree. Moreover, a set of local best individuals found is also maintained for each swarm member in local hyper-set.

□ ***Grid methods.*** Not only using the external memory to store the nondominated solutions found during the search, Coello[55] adopted a grid to deposit every particle's flight experience into a hypercube. The adaptive grid uniformly distributes the solutions along the Pareto frontier, which also helps to maintain swam diversity. Nondominated solutions will be placed into a hypercube. The grid size is limited, so those particles located in less density cube have a higher priority for retention than those in the crowding area. In [56], similar to the adaptive grid procedure, an adaptive local archive, grouped by clustering algorithm, is applied to the multiple-swarm MOPSO to improve the well-distributed sections of Pareto front that associate with each sub-swarm.

As an enhanced archiving technique, Bartz-Beielstein [57] adopted the enhanced elitist technique and an adaptive grid as an external archive, which is resized in every generation by considering the selection pressure and particle density in a hypercube. A new solution is stored in the archive only when it is non-dominated by all the other solutions already stored in the archive. Depends on the deletion fitness value, the nondominated solutions are allowed to be replaced by new particles which may improve the distribution of the archive in a fixed archive.

□ ***Rank related methods.*** The non-dominated particles are ranked based on Pareto ranking. The particles is inserted in or removed from the set of leaders (SOL)[58], compared their nondominated rank based on the objective values with the average rank in an unconstrained problem, compared their nondominated rank based on the objective values and constrain matrix with the average rank in an constrained problem. The structure of SOL, cooperating with the selection strategy, results in a spread along the Pareto frontier.

Salazar-Lechuga[36] considered both the dominance and fitness sharing of a solution in the fixed size repository. When there is full of nondominated solution in the memory, the particle with worst fitness sharing is replaced by the new one.

Best particle selection

When solving a single objective problem, the *gbest* and *pbest* solutions are exactly dominated in term of a certain swarm topology. In the case of MOP, there are several potential leaders in the nondominated solution archive, different swarm topology and various strategies to select leaders in the swarm. Further more, some other computational techniques are introduced, e.g. tournament, roulette wheel, niching, etc. Thus, kinds of selection strategies have been reported on papers. In this part, gbest is use to stand for the swarm best solution, nbest and lbest stands for the best solution within neighbors. The selection strategies of pbest are similar in some papers and they will be summarized together.

□ ***Lexicographic methods.*** Hu[40] proposed a Dynamic Neighborhood Particle Swarm Optimization (DNPSO). The idea of *nbest* selection is similar to the Lexicographic method. The nearest neighbor particles is organized together according to the calculated distance of the first objective function's fitness value, which could be also considered as a cluster operation, furthermore, the local optima among the neighbors in terms of the fitness value of the second objective function. In [42, 43, 59], Hu used a extended memory as the candidate pool which has the same selection mechanism.

□ ***Random selection methods.*** The simplest selection way for those external archive based algorithm is *gbest* is randomly selected a non-dominated particle from the external archive [34, 39]. Every non-dominated solution in the repository has equal chance to be selected. It results in a selection pressure to the crowding region and is not good for distributing particles to locate the Pareto front and maintaining diversity. In [53, 60], all previous found nondominated solution are kept in the *p*-lists, the individual *pbest* is randomly selected from *p*-lists and the individual *nbest* in the neighborhood is the non-dominated solution within the neighborhood by comparing the nondominated solutions in the *p*-lists.

□ ***Probability methods based on density and crowding.*** The *nbest* selection of MOPSO [61] is based on density, which is proportional to the number of dominated solutions inside the hypercube. There are two steps for the geographically-based selection approaches. Firstly the roulette wheel selection is used to choose the hypercube in the external archive grid and the selection probability is inversely proportional to density of the hypercube. Secondly, the particle within such hypercube is randomly selected as the *nbest*. The grid concept is also adopted in [56] as a external archive. In [55, 62], *gBest* and *pBest* position are both selected from archive and in [41], both of them are selected from group leaders in the adaptive local archive through a roulette wheel selection,. A roulette wheel selection method is also applied in [57], *gBest* and *pBest* are chosen for a particle according to the weighted sums of their age variable and their fitness values. The roulette wheel scheme, a probabilistic crowding radius-based strategy, is applied for leader (*nbest*) selection in SOL[58]. Leaders with less number of individuals around them have a higher probability of being selected, thus allowing the strategy to explore new areas and maintain swarm diversity. Salazar[49] used the fitness sharing to measure the density of those non-dominated solution. Particles in the repository (Pareto Nondominated solutions) will be chosen based on their fitness sharing by roulette wheel.

Instead of roulette wheel selection, a binary tournament selection is adopted to decide a gbest solution, in [63]. LIU [50]used the tournament niche method to select gbest where a particle with a lower niche count is selected from the archive.

LI [51]designed the selection mechanisms based on a ranking strategy and two niching methods: Niche count and Crowding Distance Assignments. All particles and pbest are ranked first and the new population for the next step is selected from the ranked fronts in ascending order, that make a selection bias to the individuals in the populations. Based on the niche count method, those "less crowded" particles with smallest niche counts will be selected as *gbest*. According to the crowding distance assignments method, the crowding distance of every solution is calculated, that is the average distance of the two solutions on either side of this solution along each of the objectives. All particles of the current Pareto-solution Set are sorted in descending order and a particle is chosen randomly from the top part of the sorted list as *gbest* for each particle. Similar to the idea of sorting and randomly selection, MOPSO-CD[52] the gbest guide for each particle is randomly selected from a specified to portion (e.g. top 10%) of the extended archive, sorted in descending crowding distance values.

□ ***Distance methods.*** The dominated trees are utilized in [44] to improve the selection of an appropriate Pareto archive member as the global 'best', which is based upon its closeness in objective space to an individual in the nondominated set. Furthermore, the local best (personal best) individuals are uniformly selected from the local Pareto solution set, instead of a single best for each particle.

For σ-method based PSO[47], every particle will find a social leader with minimal σ distance to the particle, that means every particle should find a closet line decided by a social leader. The best *nbest* for each particle from the Pareto solution set will be selected. The σ-method encourage the particles flying directly towards the Pareto-optimal front.

In [45] two archives for both *gbest* and *pbest* are employed, the nondominated solutions in nondominated local set and global set give the minimum distance with the current particle, respectively, they are selected as the local best and the global best of the particle respectively.

Hsieh[42, 53, 58, 63, 64] designed the Solution exploration strategy (SES) which will uniform distribute each particle to a nondominated solution as their local guide which will lead movement of corresponding particle. Each nondominated solution in the external archive will be selected as a *gbest*. Thus particles in the swarm will distribute in a wide area.

□ ***pbest selection methods.*** The Pareto dominance based *pbest* selection is an easy and popular method, e.g.[36, 40, 45, 51]. Between the current particle position and the previous personal best position in memory, the nondominated one will be obtained as pbest, if neither of them is dominated by the other, then it is selected randomly. In [49], only a new solution dominates the current *pbest*, the *pbest* is updated. In [34, 39], if the solutions are incomparable or mutually non-dominant, the operation is a little bit different with previous papers, that the current position of the particle is used as the updated personal best.

2.3.3. Density metric and diversity maintaining

For the MO optimization problem, the task is to find the Pareto solution set, instead of an optimum in a single objective optimization problem. Furthermore, during the optimization process, all particles will follow the found nondominated solutions saved in the archive. In the early research of MOP by PSO[53], there is no consideration to the operation of diversity maintain. However, it is crucial to distribute these solutions in the fitness space, so as to locate the Pareto front as much and close as possible. Thus one mission is how to maintain diversity in the swarm. One way is to control diversity of the found nondominated solutions and propel

particles not to select those leaders in the crowding region; another way is to introduce some improvement into the PSO update equations.

Table 1 Comparison of some typical reference

	Archive	Update Equation	Other
Coello,2002	Grid		roulette wheel
HU,2002			Dynamic neighbor
Fieldsend, 2002	Dominated tree	Turbulence	
Ray, 2002	crowding radius		
Mostaghim, 2003	σ-method	Turbulence	
Bartz, 2003	Adaptive Grid		uniform selection anti-clustering
Mostaghim, 2003	σ-method, subswarm	Turbulence	
LI, 2003	the niche count crowding distance assignment		
Pulido,2004	Subswarm, Cluster		
Cagnina,2005	Grid	Uniform Mutation	
Carlo,2005	crowding distance	Mutation	deletion
ZHAO,2005	Grid		roulette wheel
Reyes-Sierra,2005	Subswarm, Cluster	Crossover, mutation	
Leong, 2006	adaptive local archive subswarm clustering		roulette wheel
Hsieh,2007	clustering	Turbulence	
Anbido,2007	clustering		

□ ***Grid based methods.*** Coello[43] designed a geographically-based approach strategy by dividing the external repository into hypercube. Fitness is given to each hypercube that contains the found solutions, which is equal to dividing 10 by the number of resident particles. A roulette wheel selection operation is applied to select the gbest vectors to guide particles. The selection probability is inversely proportional to the number of dominated solutions inside the cube (see Figure 3). In [56], an elitist policy is performed to store nondominated solutions in an external archive with grid structure and a uniform mutation is applied.

As an extended study, Bartz-Beielstein [55] improve the grid to be adaptive to divide the archive, as well as a relative distance metric (Equation 2-2) and a randomized approach to maintain a spread in the fitness space. To those particles in the archive, a uniform selection probability, anti-clustering selection technique, which is based on the deletion probability, are used to prevent particles in huge clusters to be chosen. The calculation of all pair distance metric depends on Equation 2-2. The constant c represents a selection pressure. $\max_i, \min_i$ are the maximal and minimal values reached by an archive member at the j–th objective function.

$$f_{del} = \sum_{\forall j \neq i} \frac{1}{D_{ij}^c}, D_{ij} = \sqrt{\sum_{k=1}^{\#objectives} \left(\frac{x_i - x_j}{\max_i - \min_i}\right)^2} \tag{2-2}$$

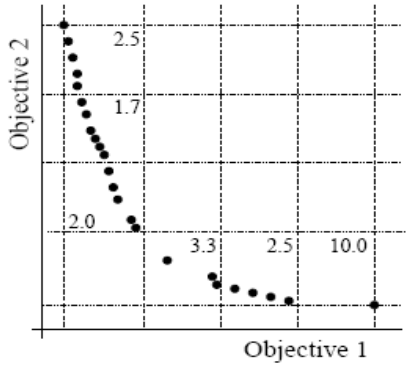

Figure 2 Hyper-cube method developed by Coello Coello

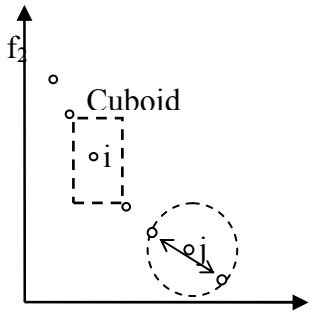

Figure 3 Crowding radius method developed by Ray

Leong [57] designed a crowdness indicator to monitor the density of a cell which is defined as the number of particles located in a cell and it involves the control of the population size at each cell. Similar to the adaptive grid procedure, an adaptive local archive is designed to improve the distributed solutions along the sections of the Pareto Front that associate with each sub-swarm, which is constructed by clustering algorithm, based upon leaders' position. A roulette wheel selection is applied to choose the group leaders.

□ ***Distance based methods.*** Ray[51] proposed that a leader is probabilistically selected based on the crowding radius of the leaders in the objective space, so that it allows exploring new area and distributing along the Pareto front. A crowding radius is the average of the distance between its left and right neighbor in the objective space (see particle *j* in Figure 3). The less the number of individuals around leaders, the higher the probability of being selected they have. Carlo[44] employed the crowding distance value of a solution to estimate the density, which measures the size of the largest cuboids enclosing particle (see particle *i* in Figure 3) without including any other point. In fact, it is the average distance of its two neighbor solutions, the same as Ray's measurement. The set of solutions is sorted in ascending objective function values. The deletion operation, based on this sorted list, replaces the most crowded particles in the archive. As well as mutation is applied to enhance the information sharing.

Mostaghim[44] use σ method to divide the fitness space (shown in Figure 5). Every particle will find a social leader with minimal σ distance. A turbulence factor, which is a random value to the current position, is added to the position updated equation. Based on a further improvement is proposed in [46]. The optimization process is achieved in two steps[50]. At the covering step the swarm is divided into subswarm by σ method, with more particles and no restriction on the archive size.

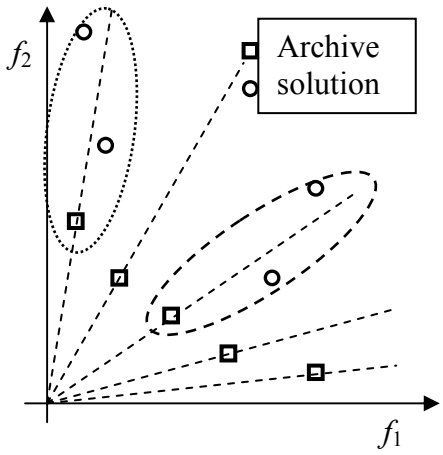

Figure 4 σ method developed by Mostaghim

Li [45] employed two parameter-free niching methods (the niche count and crowding distance assignments.) to measure the density in the archive. The niche count calculates the number of other particles within a certain distance (as shown in Equation 2-3 and Figure 5). The crowding distance of a point is the average distance of the two points on either side of this point along each of the objectives, which is the same as the methods of Ray and Carlo).

$$\sigma_{share} = \frac{u_2 - l_2 + u_1 - l_1}{N - 1} \tag{2-3}$$

To promote diversity, those particles in crowded areas (the largest niche count or the smallest crowding distance value) will be removed and replaced by randomly generated particles.

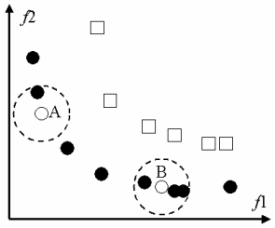

Figure 5 Niche method developed by Li

□ ***Clustering based methods.*** Hsieh[47] used the archive cluster method to keep desired diversity from the non-dominated solution set found so far. Each particle is assigned a cluster radius. Particles within the radius of cluster center will be discarded (shown in Figure 4). The radius of cluster is defined as:

$$r = \frac{\sum_{i=1}^{m-1} d\{(f_i^{\min}, f_{i+1}^{\max}), (f_{i+1}^{\max}, f_i^{\min})\}}{2s} \tag{2-4}$$

where s is the number of nondominated solution in the archive, m is the number of objective, d is the Euclidean distance, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum of the i-th objective in nondominated solutions. A disturbance operation, mutation-like evolutionary strategy, is also introduced to enhance particle's searching ability. The Gaussian noise is put into randomly selected particles' moving vectors (velocities). It is helpful to particles jump out from local search and explores more un-searched area.

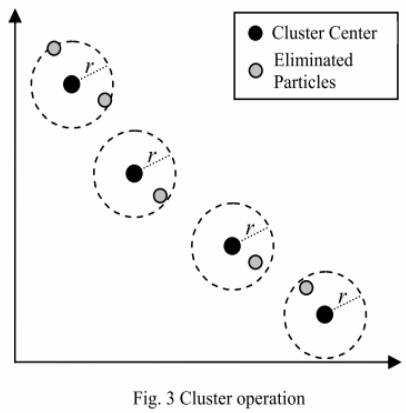

Fig. 3 Cluster operation

Figure 6 Cluster method developed by Hsieh

In [61], an average linkage based hierarchical clustering algorithm is employed to reduce the nondominated local and global set size if either exceeds the bounded size.

□ ***Subswarm based methods.***

Reyes-Sierra[59] take advantage of some co-evolutionary concepts. He integrated several diversity techniques into PSO for solving MOP. Particles are split into subswarm, which also cooperate and compete among themselves and can adjust their own size based on their contribution to the current Pareto front; the adaptive grid is used to store the obtained nondominated solutions and enforcing a more uniform distribution along the Pareto front; a clustering is performed to analyze the promising regions; crossover and mutation, operators of GA, are introduced, three types of mutation (no mutation, uniform mutation and non-uniform mutation) are applied to particles in the swarm, separately.

In [36], a PSO algorithm is run in each subswarm and at some point the sub-swarms exchange information. Johnson's cluster algorithm is performed to assign leaders into groups to guide the flight direction

□ ***Other methods.*** HU[40] designed a dynamic neighborhood strategy. At each iterations, a particle finds the nearest particles as neighbors, based on the distance which is calculated in the fitness space of the first fixed objective function. The dynamic neighborhood encourages the information sharing of *pbest* and not concentrates on a single *gbest* and *pbest*. As a further study, HU [52]introduced an external repository. Cooperated with the dynamic neighborhood strategy, it propel particles to uniformly distribution the solutions along the Pareto frontier and enhance the diversity. Fieldsend[65] proposed dominated tree for storing the particles and a turbulence variable to velocity update equation. The global best for an individual is based upon its closeness and calculated by quantitative procedure but not any random selection involved. The turbulence variable is a random number, equivalent to perturbation in ES.

HO [66], make a modification of the update equation, considering the affection of personal and social experience, velocity reverse and suddenly direction change of birds. Four additional random variables are added to the update equation of velocity. One introduced random parameter, named craziness variable, is a turbulence operation and used to maintain the diversity of the particles in an optimization algorithm. Except these four random parameters, an age variable is assigned to each solution in the repository, aiming to further maintain the diversity of the

algorithm. The roulette wheel selection mechanism is used to pick a solution according to the weighted sums of their age variables and their fitness values.

2.3.4. Strategies related to objectives fitness function

In this section, several strategies involved the manipulation will be surveyed. The first one is an approach to transform a MO problem to a single objective problem by a weighted aggregation approach. The second one is introduced the lexicographic method to consider the objectives separately according to the order at each step. The last one attempt to estimate the particle's fitness by some simple modeling operation, in order to reduce the computation cost.

Weighted Aggregation Approach

According the weighted aggregation approach, all objectives are summed with a weighted combination and a MO problem is transformed into a single objective problem. The weights are considered by normalizing and processing the contributions to the objective function. It is almost the easiest way for coping with MO problem and the combination of the fitness functions $f_i(x)$ is presented as below:

$$F = \sum_{i=1}^{k} \omega_i f_i(x) \tag{2-5}$$

where ω_i , usually assumed that $\sum_{i=1}^{k} \omega_i = 1$,can be fixed or dynamically adjusted by some strategies during the optimization. So it can be classified into 3 type techniques as following and Parsopoulos adopts the 3 fixed or adaptive weights Weighted Aggregation methods to convert a multi-objective problem to single objective problem, compared and analyzed the performance in [67].

(1) Conventional Weighted Aggregation (CWA) is a fixed weights approach, which requires a priori knowledge to choose the appropriate weights. And it has to be repeated several times search to obtain a number of Pareto Optimal points, however, in most case, we don't know how many Pareto optimal points there is. Furthermore, due to time limitations and heavy repeated computational costs, it is rare reported adopting the CWA. Xia [68] applied the linear weight aggregation to 3-objectives Job-Shop problems and got one acceptable solution for each problem. ZHANG[65] use the fixed weight aggregation PSO to design a PID controller.

(2) Bang-Bang Weights Aggregation (BWA) can adjust weights during the optimization, based on the equations: $\omega_1(t) = sign(\sin(2\pi t / F))$, $\omega_1(t) = 1 - \omega_1(t)$. The weights are changed abruptly. As an improvement of CWA and a prototype of DWA, there is no paper reported that BWA is combined with PSO for MO problems, except the comparison experiments in [69].

(3) Dynamic Weighted Aggregations (DWA) calculates the value of weights to forces the optimizer to keep moving on the Pareto Front during the optimization. Liu[70] use a modified fuzzy-Chebyshev programming (MFCP) to generate the weight or quantify the level of importance for each objective based on its satisfaction level. Marandi[71] applied an ordered weighted averaging operator to transform a 3-objectives MO problem to a one objective cost function and a Mamdani fuzzy inference system to calculate weights for objectives. Baumgartner[36, 40], adopted a gradient technique based approach to the weights adjustment. The swarm is equally partitioned in subswarm. Each sub-swarm is belonging to a weighted sum and evolving into the direction of its own swarm leader.

Only one optimal solution and fixed weight among the objectives are generally not sufficient to a multi-objective problem. So there are still existing three main problems, the first one is how to keep the found solutions and find more; the second one is whether the dynamic weighted aggregation could describe the objectives' relationship and how to design the dynamic strategy to push the optimizer close to Pareto front; the last one is with the number increase of objectives, the combination of weights and solution space dramatically increase, whether it is available and computable to considered the MO problem as one objective cost function with a dynamic weights. All weighted aggregation versions can be deduced by Kuhn-Tucker conditions, which are necessary for a nondominated solution and require the Pareto front is convex. In fact, it is not an efficient way to study the essential characteristic of MOP.

Lexicographic Method

In the lexicographic method, the objectives are ordered according to some considerations, e.g. importance, difficulty, priority, etc. The solution is obtained by optimizing the objectives separately according to the order. HU[72] introduced the lexicographic idea into the MOP-PSO. One objective is defined as the optimization objective and fixed, other objectives are defined neighborhood objectives. The first (maybe the most important, the highest priority, etc.) objective optimization is started firstly, and then, based on the first result, the second objectives will be optimized at each iteration. It is a low computation cost and computational complexity. However, selecting the fixed objective must be done by having a priori knowledge about the objective functions, due to the algorithm are sensitive to the order of objectives. And it is based on an assumption that the objectives must be orthogonal. Only two-objective functions are tested, with the increase of objectives number N (more than three), the computational complexity will have a exponential growth as $O(N^{N-1})$.

4.3 Fitness Enhancement Techniques

In real world applications, the increase of fitness function evaluation is accompanied by the dramatic increase in computation cost. Not only in real world problems, even if the mathematic description is much more complicated and calculation time-consuming, frequently fitness function evaluations will result in a very low efficiency optimization. Thus, in order to reduce the computational cost of fitness evaluation and improve the performance, fitness enhancement techniques are introduced in EA. The fitness heritance and fitness approximation are two typical enhancement techniques. Recent years, they are adopted in the POS to solve MOP.

The use of the fitness heritance is calculated the fitness value of particle at the next iteration. There are two ways[72, 73] to implemented by taking the average fitness of the two parents or taking a weighted average of the fitness of the two parents. The parents are the current particle, pbest or gbest. Reyes-Sierra[74] proposed 15 kinds Fitness Inheritance methods. The linear Combination Based on Distance methods calculates the new fitness of a particle by means of linear combinations of the positions of the particles. There are 3 variant kind versions, based on the different selection of a particle's parents. The flight formula on objective space methods is similar to the position and velocity update equation. And also has 3 versions through considering different particles' fitness. The combination using flight factors method contains two types' methods: the non-linear combination is similar to the velocity update equation. There are 6 kinds equations to calculate the new fitness; the linear combination, Similar to NLC, but is a linear combination of the 3 part. The performance of weighted average fitness inheritance is tested on a well-known test suite of multi-objective optimization problems[75]. A study [42]of the

inheritance proportion indicated that it could reduce the computational cost by 32% without affecting the quality of the obtained Pareto front and only when savings of more than a 50% of the total number of evaluations the quality on the results is significantly affected.

Table 2 Fifteen fitness inheritance methods

			I	II	III
Linear Combination Based on Distance (LCBD)			$f_{x_i}\ f_g$	$f_{x_i}\ f_p$	$f_{x_i}\ f_p\ f_g$
Flight Formula on Objective Space (FFOS)			$v\ f_{x_i}\ f_p\ f_g$	$f_{x_i}\ f_p\ f_g$	$v\ f_{x_i}\ f_g$
Combination using Flight Factors.	Non-linear Combination (NLC)	I	$f_{x_i}\ f_p\ f_g$	$f_p\ f_g$	$f_{x_i}\ f_g$
		II	$f_{x_i}/0.5\ , f_p/2\ , f_g/2$	$f_p/2\ , f_g/2$	$f_{x_i}/0.5\ , f_g/2$
	Linear combination (LC)		$f_{x_i}\ f_p\ f_g$	$f_p\ f_g$	$f_{x_i}\ f_g$

Fitness approximation techniques are employed to estimate a particle's fitness depends on the previously calculated fitness of its neighbors. The simplest way is that the fitness of a new particle is assigned to the same value of the closest particles' fitness, which is look like the operation of fitness inherence. In fact, it is estimated based on an approximate mode of fitness landscape, including, polynomials, neural network and interpolation and regression model, etc.

Yapicioglu applied the idea of fitness approximation idea in [76]. PSO is utilized to find an initial set of non-dominated solutions. And then, a general regression neural network is constructed using these non-dominated solutions. The neural network will generate a considerably larger set of non-dominated solutions.

Other fitness related modification

LI [77] introduced the Maximin strategy in game theory into the MOPSO and proposed an extension PSO, named *maximin*PSO. The fitness function of *maximin*PSO is derived from the maximin strategy to determine Pareto-domination and rank individual in the swarm instead of the popular non-dominated sorting method. No additional clustering or niching techniques is needed.

2.3.5. Performance metrics

Different of the metrics of single objective problem, the definition of quality is substantially more complex in the case of multi-objective optimization. Generally, the metrics should be considered [49] :

1. The distance of the resulting non-dominated solution set to the Pareto front should be minimized;
2. A good (in most cases uniform) distribution of the resulting nondominated solution set should occur;
3. A wide range of the Pareto front should be covered by the resulting nondominated solutions set.

Some popular performance metrics are presented as following. And the statistics of metrics and reference is summarized in Table 3.

□ **Generational Distance (GD).** It is the average distance of the current solution from the Pareto front. It addresses the closeness between the found nondominated vectors and those in the Pareto optimal set. It can be calculated by[34, 39, 41, 50, 78] :

$$GD = \frac{(\sum_{i=1}^{n'} d_i^p)^{1/p}}{n'}, d_i = \min_{j=1}^{n} \sqrt{\sum_{k=1}^{M} (f_k^i - f_k^j)^2} \tag{2-6}$$

n The number of nondominated solutions in the known Pareto front set

n' The number of current nondominated solution in the found set by algorithm

M The number of objectives

f_k^i The fitness of the i-th individual for objective k

□ **Inverted Generational Distance (IGD).** It is also used to measure how far the true Pareto front from the obtained Pareto front is and gives an idea of how close and widely spread is the obtained Pareto front with respect to the true Pareto front.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2-7}$$

where n is the number of elements in the actual Pareto front, d_i is the Euclidean distance, the same in Equation 2-6.

□ **Spacing(S).** A metric allows measuring the distribution of vectors. In addition to comparing the convergence to the true Pareto front, it is used to measure the spread (distribution) of vectors throughout the nondominated vectors found so far.

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2}, \bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{2-8}$$

□ **Error Ratio (ER)**. This metric is proposed by Veldhuizen to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set, as shown in Equation 2-9. $e_i = 0$, if the i th solution is a member of the Pareto optimal set, and $e_i = 1$ otherwise.

$$ER = \frac{\sum_{i=1}^{n'} e_i}{n'} \tag{2-9}$$

□ **Success Counting (SCC).** The measure counts the number of elements of the Pareto front obtained, that belong to the true Pareto front of the problem. $SCC = \sum_{i=1}^{n'} s_i$, where n' is the number of vectors in the obtained nondominated set

□ **Two Set Coverage (SC).**

It is also called C metric. $SC(X', X'')$ map the two set X', X'' to the interval $[0,1]$, which gives the ratio of points in X' that are dominated by at least on point in the X''. It could be presented as:

$$SC(X', X'') \equiv \frac{|\{a'' \in X''; \exists a' \in X' : a' \preceq a''\}|}{|X''|} \tag{2-10}$$

If all points in the set X' dominate or are equal to all point in X'', then $SC = 1$, $SC = 0$ implies the opposite.

□ **Diversity (Δ).** It is a diversity metric, which measures the extent of distribution achieved among the obtained solutions and is defined as

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n'-1} |d_i - \bar{d}|}{d_f + d_l + (s-1)\bar{d}}, \bar{d} = \frac{\sum_{i=1}^{n'-1} d_i}{s-1} \tag{2-11}$$

where d_f, d_l are the Euclidean distance between the extreme solutions and the boundary solutions of the obtained nondominated set.

□ **Maximum Spread (MS).** It gives a value which represents the maximum extension between the farthest solutions in the obtained non-dominated set.

$$MS = \sqrt{\frac{1}{m}\sum_{k=1}^{m}\left\{\frac{\min(f_k^{\max}, F_k^{\max}) - \max(f_k^{\min}, F_k^{\min})}{F_k^{\max} - F_k^{\min}}\right\}^2} \tag{2-12}$$

$f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum of the k th objectives in nondominated solution found so far, respectively; and $F_k^{\max}$ and $F_k^{\min}$ are the extreme in the actual Pareto optimal set. In a problem with two objectives, the value will be the Euclidean distance between the two farther solutions. In this metric a bigger, value indicates better performance.

□ **R-metrics.** The R-metrics compare two non-dominated sets on the basis of some utility functions and determine the expected number of occasions where the solutions of on set are better than the other.

□ **Attainment surface.** Attainment surface use a set of non-dominated solutions to define a surface that delineates the objective space into a dominated and non-dominated region.

□ **S-metirc.** The S-metric calculates a hyper-volume of a multi-dimentional region enclosed by the non-dominated set to be assessed and reference point to measure the diversity and the convergence of the obtained non-dominated set.

Table 3 Statistics of metrics and reference

Category	Reference
Number of nondominated solution found	[43, 45, 48-50, 55, 58, 59, 61, 63, 79]
Generational Distance (GD)	[72, 74, 75]
IGD	[43, 48, 51, 53, 55, 58, 59, 61, 63, 79]
Spacing (SP)	[48, 59, 61, 79]
Error Ratio (ER)	[72, 74, 75]
Success Counting (SCC)	[40, 44, 46, 51, 55, 61, 74]
Two Set Coverage (SC)	[45, 50]
Diversity (Δ)	[45, 58, 63]
Maximum Spread (MS)	[56]
R-metric and S-metric	[79]

2.3.6. Application and test suits

Since MO issues have been studied for a long time, no matter by deterministic methods or stochastic approaches, there are many classic application cases and test functions investigated by EAs and GAs. All of them provide a shot cut for researching and testing PSO. Furthermore, based on the previous proposed test functions, some comparison research papers have been published, e.g. the pure comparative studies by Coello[78], Emma [80-82], etc, and comparative analysis with a new proposed PSO algorithms in the paper, which is summarized in Table 4. Some tested functions are popular and often adopted in an experiment and it is summarized in Table 5. In addition, some resources can be found online[55, 72, 74].

Table 4 Compared MO optimization algorithms

Compared algorithms	Reference
Cluster-MOPSO	[61]
CO-MOEA	[40]
original Dynamic Neighborhood PSO	[79]
Micro-Genetic Algorithm for MO (microGA)	[43]
Multi-Objective Genetic Algorithm 2 (MOGA2)	[48, 49, 51, 52]
MOPSO	[45, 48, 50, 53, 58, 61, 78, 79]
Non-dominated Sorting Genetic Algorithm (NSGA II)	[55]
Non-dominated Sorting PSO	[43, 53, 58, 78, 79]
Pareto Archived Evolution Strategy (PAES)	[52]
Partitioned Quasi Random Selection (PQRS)	[45, 55, 72, 74]
sigma-MOPOS	[47, 61]
Strength Pareto Evolutionary Algorithm (SPEA)	[44, 45, 78]
SPEA II	[55, 72]

Table 5 MO functions tested in references

Test functions	Num-OBJ, Pareto Front properties	Reference
DTLZ2	3-OBJ, 12 decision variable, a surface Pareto front	[72]
DTLZ6	3-OBJ	[49, 51, 56, 58]
Kursawe	2-OBJ, discontinued, three disconnected Pareto curves	[41, 43]
MOP5	3-OBJ, unconstrained disconnected, asymmetric	[43]
MOPC1	2-OBJ, constrains, convex	[47, 53]
TP1 (Schaffer)	2-OBJ, discontinuous	[47]
TP2		[44-50, 72]
ZDT1 (TP3)	2-OBJ, Convex, continuous, uniform distribution of solutions Difficulty: Large number of parameters	[45, 50, 72]
ZDT2	2- OBJ, non-convex	[43-50, 53, 57, 58, 72, 83]
ZDT3 (TP4, Func1 by Deb, MOP6)	2-OBJ, non-convex, unconstrained, discontinuous , consistent of 4 Pareto curves	[44, 46, 48-50, 53, 55, 57, 72]
ZDT4 (Func2 by Deb)	2-OBJ, convex, continuous Difficulty: Large number of local Pareto-optimal fronts, 60 local Pareto fronts	[46, 48, 49]
ZDT6	Non-convex, disconnected-continuous, non-uniform	[84]

	density of solutions Difficulties: adverse density of solutions, non-convex front and discontinuous front

There are several different names of the same test functions in various papers, such as TP4, ZDT3, MOP6, and one function by Deb[84] stand for the same function. TP3 and ZDT1 are the same function, ZDT4 and another function by Deb[57] are the same. The Schaffer function is the same as TP1.

Besides the research work on the test suits, some other application cases are also investigated, e.g. a welded beam problem subject to several constraints and a design work of a multiple disc brake, by Ray[69], Design of reconfigurable machine tools (RMTs) involves several objectives, by LIU [85], an optimization problem of the thermodynamic parameters of the Na2O-SiO2 system in the mineralogy by Halter [86], an I-beam problem with the bending stress constrain by Eric[70], a hard multi-objective electromagnetic problem of a planar antenna for using in two-way satellite Internet services in the Ku-band, by Marandi [42], a non-linear constrained multi-objective economic load dispatch problem in power system with 3-objectives and constrains, by Yapicioglu[62], the optimal control problem for interplanetary trajectories design to minimize both the propellant mass to maneuver and the transfer time for the Earth to the target planet, by Mich [87], the molecular docking problem to find a good position and orientation for docking a small molecule to a larger receptor molecule by Stefan [87].

2.4. Handing constraints with particle swarm

Most realistic optimization problems involve a number of constraints and should be minimized or maximized subjected to those constrains on the possible values of the independent variable, which can also be equality constraints and inequality constraints. The constraint problems can be addressed using either deterministic or stochastic methods. Deterministic methods such as Feasible Direction, Gradient-descent methods, Gradient-projection method, Quadratic Programming, Sequential Quadratic Programming and Lagrange multiplier methods, etc, usually require objective or constraint functions should be differentiable, continuous and the feasible domain is convex. Unfortunately, lots of problems don't meet these requirements.

Compared with the deterministic methods, the evolutionary computation techniques depend less on characteristics of the constraint problem and push themselves to a strong support in the research of constraint problem. Many research results and surveys about constraint handling techniques in evolutionary computation methods, i.e. Evolutionary Algorithm (EA), Genetic Algorithm (GA), etc., have been published [88-93].

In the recent ten years, Particle Swarm Optimization (PSO) has aroused considerable attention regarding its potential as an efficient algorithm for constraint problems. In this review, constrained handling methods for Constrained Optimization Problems (COP) are surveyed in detail and for the Constraint Satisfaction Problem (CSP) are also discussed.

2.4.1. Constraint problem

A constrained optimization problem is usually written as a nonlinear programming (NLP) problem as following:

$$\min f(x), x \in R^n$$
$$subject \quad to \quad \begin{matrix} g_i(x) \geq 0, i = 1,2,...l \\ h_j(x) = 0, j = 1,2,...m \end{matrix} \tag{2-13}$$

where $f(\cdot)$ is the objective fitness function, $x \in S \cap F$, $S \subseteq R^n$ is the *n*-dimension search space and F is the feasible region satisfying the inequality constraints $g_j(x)$ and equality constraints $h_i(x)$. In constrained optimization, the general aim is to guide solutions in the infeasible domain to feasible area. Large classes of early methods transform the constrained problem to a basic unconstrained problem that can be easier solved, by using a penalty function which were proposed in 1940s and later improved by many researchers. The penalty functions utilize the amount of constraint violation to penalize the infeasible solutions and to favor feasible solutions. They simply penalize constraints and build a single objective function. However there are lots of limitations, such as the selection of appropriate penalty coefficient is not an easy task. Penalty functions are classified into two main categories: stationary and non-stationary. Stationary penalty functions use fixed penalty values throughout the minimization, while in contrast, in non-stationary penalty functions, the penalty values are dynamically or adaptively modified. In order to overcome the limitations of , many alternative methods are proposed, such as rejecting infeasible solutions, repairing the infeasible solutions, separation strategies of constraints and objectives, Pareto principle related constraint handling techniques, designing special mutation and crossover operators[94], etc.

2.4.2. Constraint penalties and objective approaches

Static Penalty

An individual of penalty category methods can be evaluated by the following formula [88]:

$$F_i(x) = f_i(x) + \sum_{j=1}^{l+m} r_{k,j} \phi_j^2(x) \tag{2-14}$$

where $r_{k,j}$ is a penalty coefficient for the j-th constraint and the k-th level of violation which is defined by user. $\phi_j^2(x)$ impose a penalty controlled by a sequence of penalty coefficients.

Zhu[95] concerned on solving Vehicle Routing Problem with Time Windows (VRPTW) problem. Some large positive numbers are employed as the punishment coefficients $r_{k,j}$ to deal with vehicle capacity, travel time and arrival time feasibility constraints. The constraint problem is converted to an unconstraint problem with these static coefficients incorporated into the object function

The static penalty approach belongs to stationary penalty category. The penalty factors $r_{k,j}$ remain constant during the entire evolutionary process. The description of the approach is simple, although there are lots of coefficients to be set up in some cases. Whereas, generally penalty coefficients are problem dependent and real time related with the optimization process. If the coefficient value is not penalized enough, the proportion of penalty in the objective function is too small to have an infeasible solution; on the contrary, if the value is too large, it will lead to premature convergence to a local optimum and is difficult to move from one feasible region to another one, which are disconnected each other. In general, penalty function approaches heavily dependent on the coefficient settings which require a strong a priori and are different for variant issues.

Dynamic Penalty

Under this category, penalty coefficients $r_{k,j}$ are the function of the generation number. Normally, it is designed as an increase function over the generation number (i.e., proportional) so as to allow individuals gradually converge to feasible region.

Parsopoulos[96] proposed a non-stationary multi-stage assignment penalty function, defined by adding the sum of violation of all constraint functions to the objective functions as penalty, defined as equation 2-15.

$$F_i(x) = f_i(x) + r_k \phi(x)$$
$$\phi(x) = \sum_{j=1}^{l+m} \theta(q_j(x)) q_j(x)^{\gamma(q_j(x))} \tag{2-15}$$

where $q_j(x) = \max\{0, g_j(x)\}$ is a relative violated function of constraints and $\gamma(\cdot)$ is the power of $q_j(x)$; $f_i(x)$ is the objective function and $g_j(x)$ are the constraints in Equation 2-15. The weight r_k of each violation is a function of iteration number k and designed as two versions $r_k = \sqrt{k}$ and $r_k = k\sqrt{k}$ for different problems. The $h(\cdot)$, $\theta(\cdot)$ and $\gamma(\cdot)$ are problem dependent[96]. The main difficulty of the dynamic penalty function method is also the difficulty of selecting an appropriate value for the penalty coefficients that adjusts the strength of the penalty[97]. Parsopoulos claimed it outperformed other different evolutionary algorithms, such as Evolution strategies and genetic algorithms.

Xu[98] studied the realistic portfolio selection problem with transaction costs, tax and buy-in threshold, based on Markowitz mean-variance model. Constraints are handled by dynamic penalty factor $r_k = C \cdot k$, and the violation of both equality and inequality constraints are considered as Equation 2-16.

$$q_j(x) = \begin{cases} \max\{0, g_j(x)\} & 1 \le j \le l \\ abs(h_{j-l}(x)) & l+1 \le j \le l+m \end{cases} \tag{2-16}$$

The algorithm is sensitive to the value of constant C in penalty factor, but there is no any suggestion how to select in the paper.

In some papers, it is reported that penalties work better than static penalties. In fact, it works well for some simple problems, but failed for more difficult problems [99]. The value of $r_{k,j}$ is usually predefined monotonically increasing and problem dependent, thus the convergence process is irreversible and once the swarm is trapped in a feasible local optimum, it may stay there forever.

The crux of dynamic penalties is the configuration of those sensitive problem dependent coefficients.

Adaptive Penalty

Both dynamic penalty and adaptive penalty approaches belong to non-stationary penalty function methods. Compared to dynamic penalty methods, $r_{k,j}$ is adjusted automatically and adaptively relies on the feedback information from optimization process.

Liang[94] designed the new constraint handling strategy for Dynamic Multi-swarm Optimizer (DMS-PSO). For each sub-swarm according to the adaptive constraints combination, the roulette selection is used to assign the objective function or a single constraint as its target. The average of weighted constraints is to balance the impacts of different constraints (as Equation 2-17). The constraints that are more difficult will have more sub-swarms work for it, while the easier ones will have less or even no sub-swarm working for it.

$$\phi(x_i) = \sum_{j=1}^{l} (\lambda_j \cdot g_j(x_i)) \quad \lambda_j = \frac{1/g_{j\max}}{\sum_{j=1}^{l} 1/g_{j\max}} \quad j = 1, 2...l \tag{2-17}$$

He[100] developed a c o-evolutionary particle swarm optimization approach (CPSO) for constrained optimization problems and employed the notion of co-evolution to adapt penalty factors. The PSO is applied with two kinds of swarms for evolutionary exploration and exploitation in spaces of both solutions and penalty factors.

Ismael[101] calculated the constraint violation separately as Equation 2-18. Where $[h_i(x)]_+ = \max\{0, h_i(x)\}$, $H : R^n \to [1, +\infty]$

$$H(x) = e^{\left(\sum_{i=1}^{l} \log(1+g_i(x)) + \sum_{i}^{m} \log(1+[h_i(x)]_+)\right)} \tag{2-18}$$

Although different problems lead to a different adjust strategies for $r_{k,j}$, adaptive penalty methods relatively require less prior knowledge of problems. More recent constraint-handling approaches in EA pay a lot of attention to this issue [92].

2.4.3. Separation strategies of constraints and objectives

Instead of using penalty functions for constraint-handling what is the most common approach, some strategies consider the constraints and objectives separately. In some cases, the separation strategies are employed to compare individuals based on the objective fitness and eliminate some infeasible solutions. And in other cases, the constraint violation is calculated as a measurement to the feasible region of a particle, so as to be used to sort or rank.

Separation for Death Penalty

The death penalty method is a popular and probably the easiest way to handle objectives and constraints separately. It utilizes the interior penalty rules to discard infeasible solutions from a population that means the heuristic information from infeasible points is neglected. It works well in term of linear constraint problem. Whereas, the drawback of the method is the limitations for problems, that it might be stagnated, when the feasible search space is non-convex or disconnected, or small. Especially for some problem with small feasible region, such as equality constraint problem, it is almost too hard to generate new feasible solutions, especially at initialization stage.

Amin[102] applied particle swarm optimization to the hardware/software partitioning problem which is to minimize the area and communication costs between software and hardware blocks subject to execution timing constraint requirement. The fitness value of feasible solution is defined as summation of communication cast and hardware cost, otherwise it is set to infinite if the timing constraint can be satisfied. In some cases, It will take much more time for a fixed number of generations, since looking for feasible solutions.

Hu[103, 104] employed the preservation of feasible solutions method (FSM) for constraint handling with PSO. Constraints are only used to see whether a solution is feasible or not[105]. PSO is started with forcing all particles into the feasible space before any evaluation of the objective function has begun. And feasibility function is used to check if the new explored solutions satisfy all the constraints. Only those particles in feasible space are counted for update of *pBest* and *gBest* values (for both global and local version), the current position of some particles can be infeasible that keeps the possibility to move from one feasible region to another one, in term the disconnected feasible region distributed in the search space.

Talal[106] applied PSO to optimize the repairable-item inventory model with state-dependent repair and failure rates as well as steady-state environment. PSO is modified to handle stochastic constraints and discrete decision variables. A variance of the estimated expected performance constraint is considered as a supplementary item to inequality constraints that original constraints can be transformed into a manageable. Only those records of feasible solutions are kept.

Goldbarg[107] studied the bi-objective degree-constrained minimum spanning tree problem. A tree is represented with the edge-set representation. Particles are initialized based on a depth first search. The local search procedure and the path-relinking operation with the back and forward strategy are utilized as the velocity update operators. If constraint is violated, the current edge is discarded. A non-dominated solution, represented by the set of edges constituting the tree, is kept in an archive with size limited. As a constraint problem, those infeasible solutions are dealt with death penalty methods.

There are no predefined parameters and preprocessing to manipulate the objective and constraints which are handled separately. Thus it is an effective method for handling constrains. Coath[108] reported that the rate of convergence and accuracy of those methods proposed by Hu[103] and Parsopoulos[96],separately, were very competitive at finding near-optimal solutions. However the obvious drawback is that it may have a very high computational cost to initialize a group of feasible solutions in some functions, such as the constrained nonlinear optimization problems with extremely small feasible spaces or equity constraints optimization problems. It may take a impractical long time until the initialization process is completed. In some case in [104], even the generation of one million of random points was insufficient to produce a single feasible solution[109].

Separation for Sorting

In this category, the total violation of constraints for each infeasible particle will be counted as a distance measurement to the feasible region, instead of discarded directly. All solutions are sorted and selected based on the objective fitness and violation summation. The common procedure to compare two solutions is following the criteria (also termed as "feasibility and dominance" (FAD), see Figure 7) and the Individuals are evaluated using Equation 2-19, where $\phi_j(\cdot)$ stands for all constraints, including both equality and inequality constraints.

1) Between two feasible particles, the one that has a better fitness value wins;
2) A feasible particle is always preferred over an infeasible one;
3) Between two infeasible particles, the one having lower total violation of constraints wins.

Figure 7 Separation sorting procedure

$$F_i(x) = \begin{cases} f_i(x) & \text{if feasible} \\ \sum_{j=1}^{l+m} \phi_j(x) & \text{if infeasible} \end{cases} \quad (2\text{-}19)$$

Zhang[110] proposed a hybrid particle swarm with differential evolution operator, named DEPSO. DEPSO obeys the separation sorting rules in Figure 7 and Equation 2-19. The bell-shaped mutations, similar to that used in differential evolution, keep consensus on the population diversity along with the evolution. It is applied only on the *pBest* so as to keep the self-organization. The *gBest* also influences the mutation that might enhance the social learning capability and speed up convergence. The black-box problems and problems with small feasible region are still difficulties for DPSO. The approach is sensitive to the values of crossover constant unless the crossover constant could be decided in term of the correlation of the parameters.

Angel[111] designed particle evolutionary swarm optimization algorithm (PESO). The constraint handling and selection mechanism are based upon the "feasibility and dominance" rules (in Figure 7). The optimization process is divided into several stages. Two new perturbation operators "c-perturbation" and "m-perturbation" are implemented in different stage so as to enhance the diversity and deal with the premature convergence problem. The "c-perturbation"(similar to the reproduction operator in differential evolution algorithm) is added to every particle, but in DEPSO[110] this operator is only applied to *pBest*. "m-perturbation" is performed on each dimension of decision variable vector with some probability.

Pulido[109] introduced a normalized sorting selection for processing infeasible individuals to lie on the closeness to the feasible region. Based on Equation 2-19, the infeasible points are calculated as:

$$F_i(x) = \sum_{j=1}^{m} \frac{\phi(x)}{\| \phi \|} \quad \text{if infeasible} \quad (2\text{-}20)$$

The total violation of constraints, defined in Equation 2-20, sums normalized violation with respect to the largest violation of each constraint. A turbulence operator, incorporated to prevent trapping in local optima and improve the exploratory capabilities, is calculated depends on the probability which is related to the current number of iteration. It means there is a much higher probability at the beginning and tends to stable over time. It is claimed[109] that the PSO approach does not require any user defined parameters and less objective evaluations, however, it does not provide any suggestion to decide the largest violation of constraints.

In [112], it adopted almost the same the decision-making scheme in [110], but not the normalized constraint violations, Zielinski just assigned a sum of constraint violations to infeasible solutions. If no feasible individual is found, the search is guided only by the amount of constraint violations. In the paper, 24 constrained single-objective optimization problems are tested and it successfully accomplished the optimization of test functions with disjoint feasible regions, or with many inequality constraints and different types of objective functions. However, as the other optimization techniques, it is also a challenge for handling those problems with extremely small feasible space, such as the problems with equality constraints. It is also declared that a high dimensionality cause difficulties and active constraints at the optimum could not be sufficiently analyzed since containing equality constraints or high dimensionality, simultaneously.

Lexicographic Separation

Takahama developed the α[113] and the ε[97, 114] constrained methods, which consider the objective and constraints separately and allow all solutions are comparable with each other by replacing the ordinal comparisons with the α level and the ε level comparisons. With the lexicographic order of the ε level comparisons, the violation of constraints is more prior than fitness, the formula goes as follows:

$$(f_1,\phi_1) <_\varepsilon (f_2,\phi_2) \Leftrightarrow \begin{cases} f_1 < f_2 & if\ \phi_1,\phi_2 \le \varepsilon \\ f_1 < f_2 & if\ \phi_1 = \phi_2 \\ \phi_1 < \phi_2 & otherwise \end{cases} \tag{2-21}$$

where ϕ_i is the constraint violation which can be calculated as the maximum of all constraints (as Equation 2-22) or the summation of all constraints (as Equation 2-23).

$$\phi_i(x) = \max\{\max_j\{0, g_j(x)\}, \max_j |\, h_j(x)\,|\} \tag{2-22}$$

$$\phi_i(x) = \sum_j \| \max\{\max_j\{0, g_j(x)\}\|^p + \sum_j \| h_j(x)\|^p \tag{2-23}$$

Based on lexicographic order mechanism of Equation 2-21, constraint problems are converted to unconstraint problems, since it is comparable among solutions. Equality constraints can be easily handled. Furthermore, Takahama[115] proposed an adaptive strategy to maintain the maximum velocity $V_{\max}$, which is a Lipschitz condition used to guarantee the input and output is boundary[116]. The value influence the stability and performance of algorithm and is also problem dependent. The swarm is divided into several sub-swarms which have their own settings about the maximum velocity. The worst group will adjust the value in term of the best sub-swarm. It is also claimed that εPSO could find much better solutions that had been never found by other methods and on average for all problems[116]. However, only the results that the ε level is assigned to 0 are reported, in this case, the constraint handling approach is the same as lexicographic orders in which the constraint violation precedes the fitness. The study of other settings of variable ε is the further research work.

Liang[94] combined Dynamic Multi-swarm Optimizer (DMS-PSO) and Sequential Quadratic Programming (SQP) method. SQP is used to achieve a better local search performance, which can also be regarded as a kind of hybrid method in this sense. The novel constraint-handling mechanism is incorporated into the DMS-PSO, that the population is periodically divided into sub-swarms which are adaptively assigned to explore different constraints according to their difficulties. The constraints that are more difficult will have more sub-swarms work for it, while less or even no sub-swarm will be assigned to easier ones. The priority relations are similar with Equation 2-21, except the ϕ_i of DMS-PSO is defined as Equation 2-17. Thus, it is classified to a dynamic lexicographic method.

Ismael[101] incorporated the dominance concept into PSO to address nonlinear constrained optimization problems. Both objective function and the constraint violation (calculated as Equation 2-18) are regarded in order to extend PSO to constraint problems. The dominated comparison is implemented as Equation 2-21,, except ϕ_i.

2.4.4. Pareto principle related constraint handling methods

In the past few years, some researchers incorporated the concepts of multi-objective optimization into the constraint-handling techniques. Pareto Principle based methods are effective for the research of multi-objective optimization. And it also produced a considerable interest to adopt the Pareto concept in constraint handling. The N objective functions of a constrained optimization problem, incorporated with the $m+l$ constraints are organized as a $N+m+l$ multi-objective function. Generally, constraint problems with a single objective function are redefined as a $1+m+l$ multi-objective function. The research of multi-objective constraint problems are gradually increasing. Both cases are treated as belong to the same class problems which employ the Pareto principle based methods in this section.

Ray[117] utilized a Pareto ranking scheme to handle constraints, which is a concept widely used in multi-objective optimization scenarios. The equality constraints are transformed to a set of inequalities with a tolerance and all constraints are presented in a unified formulation. An effective multilevel information sharing strategy is implemented by constructing a CONSTRAINT matrix corresponding to each constraint of every particle; moreover all elements in the constraint matrix are sorted based on the non-dominated ranking scheme. All these mechanisms enforce the selection pressure to the set BPL. Instead of the regular update equation, the simple generational operator, which can generates a new variable between a individuals and leaders, is useful to avoid premature convergence. It is claimed the proposed algorithm converged fast benefiting from the reduction of fitness evaluation. Although the methods to handle equality constraint are designed, there are no test functions with equality constraints studied in the paper. The Pareto ranking and the generation of unique individuals' process are two computationally expensive to the multilevel information sharing mechanism.

Wang[118] took into account three design constraints and three objectives for generating system adequacy assessment, in order to achieve reliability assurance during system operations. A constrained multi-objective particle swarm optimization (CMOPSO) algorithm is proposed to derive a set of Pareto-optimal system configurations. The non-dominated solutions are stored in a archive and gbest is produced by the binary tournament selection from the archive. The fuzzified global best selection, niching and fitness sharing, and disturbance factor are applied in order to enhance solution diversity. A rejecting strategy is adopted to have the constraint check to the solutions. If all of the constraints are satisfied, the solution is eligible to compare with the non-dominated solutions in the archive. As long as any constraint is violated, the candidate solution is considered as infeasible.

Ji[119] developed the Divided Range Multi-objective Particle Swarm Optimization (DRMPSO) which utilize a Multi-swarm Particle Swarm Optimization for multi-objective and multi-constraint optimization problem. At each generation, according to every particle's fitness, population is dynamically spitted into sub-swarms whose number is the same as the number of objective function. That means each swarm works for an objective function at each generation. Populations are consisted of feasible and infeasible individuals. In each sub-swarm, the feasible individuals fly towards Pareto front guided by objective functions and the infeasible individuals is guided towards feasibility by an unfeasibility evaluation function, generally the constrained function or the weighted constrained functions. If the proportion of infeasible individuals exceeds the threshold, the information from the best global feasible individual is adopted to update the infeasible particles iteratively until they become feasible individuals. Moreover, the

threshold reduces gradually and the number of non-dominated solutions is directly linked to the population size, so a larger population size is preferred.

But in many cases, solving multi-objective optimization problems is a more difficult and expensive task than solving single objective optimization problems[120].

2.4.5. Linear constraints

In the search space, the linear constraints define a convex polyhedron, named the feasible region. Since locating the feasible region and considering the linear equality constraints and the objective functions, the linear class PSO algorithm always moves inside the region. The feasible region of linear constraints is relative small in the search space and a high computational time will be required to the initialization stage and repair those infeasible solutions into the feasible region during optimization.

Paquet [121, 122] proposed the Linear PSO (LPSO) and the Converging Linear PSO (CLPSO) to the optimization problems with linear constraints $Ax = b$. All individuals are initialized in the feasible region and velocity vectors are set to zero. So as to guarantee the updates of velocity are linear combination of previous position and velocity vectors, the acceleration factors are kept constant in Linear PSO. Thus, the following calculation of position vectors must be feasible solutions. The Guaranteed Convergence Particle Swarm Optimizer (GCPSO), developed by Bergh, is introduced to improve the *gBest* for the Converging Linear PSO and only particles with feasible direction are updated. Except initialization process, these methods save more computation cost than other methods to solve constraint problems. Yet, the acceleration factors are set as constants may weaken the searching ability of algorithms.

Halter[123] investigated an optimization problem of the thermodynamic parameters of the Na2O-SiO2 system in the mineralogy as a Bi-level Optimization problem, consisted of upper and lower levels problems. The Linear Multi-Objective PSO (LMOPSO) is designed to solve linearly constrained multi-objective optimization problems in the lower level, which contains three objective functions and two linear constraints. LMOPSO is efficient to preserve and guarantee the feasibility of the solutions during the optimization, when solving linearly constrained problems. The population members are evaluated and the non-dominated solutions are found and stored in the archive. In fact, since the boundary conditions constructed by inequality constraints, two feasible solutions may lead some of the particles to the infeasible part to inequality constraints. Thus, a feasibility preserving method has to be designed to move the infeasible particles back to a random feasible position or to the boundary. The rescaled velocity is as below:

$$\begin{aligned} \vec{x}_{final} &= \vec{x} + \lambda \vec{v} \\ \vec{v}_{final} &= -\lambda \vec{v} \qquad \lambda \in [0,1] \end{aligned} \tag{2-24}$$

2.4.6. Constraint satisfaction problem

A constraint satisfaction problem (CSP) is defined on a nonempty domain of possible values and a set of constraints. A set of variables to a CSP, taken from the domain, is a complete assignment that satisfies all the constraints. Furthermore, some CSPs also require a solution that maximizes or minimizes an objective function. Problems that can be expressed as constraint satisfaction problems are the a warm-up problem, Pythagorean triple problem, *n*-queens problem, map

coloring problems, the Sudoku, the boolean satisfiability problem, send-more-money puzzle, scheduling problems and so on.

Schoofs[124] applied a discrete particle swarm for solving binary constraint satisfaction problems. The binary PSO proposed by Clerc is adapted and several improvements are applied in the velocity update equation: the previous velocity vector is neglected and only current states are considered; a deflection operator, similar with the mutation operator in Gas, a deflection operator is used to enhance the exploitation power; the number of conflicts a variable causes in the candidate solution are considered in the multiplication of a coefficient with a velocity; no-hope/re-hope system are employed to drive exploration force. The number of conflict checks until termination and the success rate are used in the binary constraints problem experiments. Compared with ant colonies and genetic algorithms, Schoofs reported that the proposed particle swarm method is outperformed. One of the reasons that the algorithms perform more effectively is a particle will not change its assignment to the variable, only if it is in conflict by taking constraint violation information into account for each variable[125].

Lin[125] made a further research work based on Schoofs' work. The deflection operation is modified as a probability threshold rather than a Boolean switch. A no-hope and hop technique is added to fix constraint violations or to minimize the constraint violations when the regular swarm stops improving the search. A conflict count function and a distance estimation function are employed as penalty functions to guide the particles. The conflict function counts constraint violations as a penalty score when the constraint and the distance estimation function compute the distance from a potential solution to a feasible solution.

Yang[126] added the max-degree static variable ordering of variables to the fitness function of [124], so that some variable satisfied constraint firstly with high probability and guides the direction of the whole swarm by selecting the gbest and pbest particles. The algorithm is similar to the Backtracking search in solving problems

Alberto[127] used the geometric framework to design the Geometric particle swarm optimization (GPSO) to solve the Sudoku puzzle. Deferent with standard PSO, no velocity is used, the update of position is the convex combination, there is mutation and the parameters ω, $\varphi 1$, and $\varphi 2$ are positive and sum up to one in GPSO. The 4 types of constraints in the Sudoku constraint satisfaction problem are divided into hard constraints, which all solutions have to be respected, and soft constraints that can be only partially fulfilled and the level of fulfillment is the fitness of the solution. The fitness function is consisted of three parts summation: the number of unique elements in each row, the number of unique elements in each column and the number of unique elements in each box. It also could be considered as a special case of static penalty. Compared with those methods using brute force trial-and-error search employing back-tracking, it is more efficient. It is claimed in the paper that "the 9x9 Sudoku puzzle of any difficulty can be solved very quickly by a computer" and "this is the first time that a PSO algorithm has been successfully applied to a non-trivial combinatorial space".

One difficulty for PSO, GA and other swarm intelligence methods is there is no explicit objective function of those problems[124]. The simplest way to design a fitness function is counts the number of constraint violations by assignments, which leads to no difference could be made, if assignments has an equal number of constraint violations. More repairing methods should be used as infeasible solutions or unsatisfiable potential solutions handling strategies for further enhancing the algorithms.

3. Swarm-Powered Systems

3.1. Fundamental framework

It shows PSO can successfully track the changing optima in an efficient way with different dynamic tracking strategies. The results are very promising and show that PSO can be used to implement fast learning algorithms to control dynamic systems.

Currently the information processing of PSO is focused on two levels of intelligence. In the velocity update formula, one term is the individual cognition component and the other is the social communication component. The individual cognition component represents the search ability of the particle itself. (c_1, p_{id}, v_{id}, V_{max}, w). The social communication component represents the influence from the social environemnt (c_2, p_{gd}).

However, for more complex applications, such as dynamic environments , higher level intelligence should be integrated into the algorithm. This kind of intelligence either cannot be furnished by the particles themselves or is not easy to accomplish without losing performance. Environment detection and response are new intelligence methods which cannot be achieved by individual particles. This is population-level intelligence, shown in Figure 8; at this level, we can integrate more human knowledge and intelligence to the PSO to accommodate complex systems. For example, for dynamic systems, we can make PSO maintain a certain level of diversity instead of converging to the optimial area quickly.

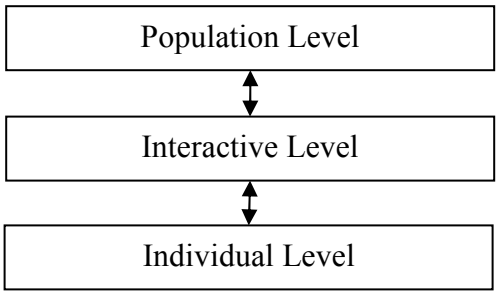

Figure 8 Three levels in the swarm model

3.2. PSO diversity measurement

3.2.1. Introduction

Background

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart in 1995 [128-130]. It is being researched and utilized in over 30 countries.

The process for implementing the *global* version of PSO is as follows:

1. Initialize a population (array) of particles with random positions and velocities on *n* dimensions in the problem space.
2. For each particle, evaluate the desired optimization fitness function in *n* variables.

3. Compare particle's fitness evaluation with particle's *pbest*. If current value is better than *pbest*, then set *pbest* value equal to the current value, and the *pbest* location equal to the current location in *n*-dimensional space.
4. Compare fitness evaluation with the population's overall previous best. If current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.
5. Change the velocity and position of the particle according to (1) and (2), respectively [131-133]:

$$v_{id} = w * v_{id} + c_1 * \text{rand}( ) * (p_{id} - x_{id}) + c_2 * \text{Rand}( ) * (p_{gd} - x_{id}) \quad (3\text{-}1)$$

$$x_{id} = x_{id} + v_{id} \quad (3\text{-}2)$$

After step 5, loop to step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

Particle swarm optimization is especially useful for obtaining answers to problems involving multiple objectives and multiple constraints. It has outperformed other algorithms in a number of benchmark tests, and a number of researchers have developed methodologies for its utilization for these types of problems.

There is also a *local* version of PSO in which, in addition to *pbest*, each particle keeps track of the best solution, called *lbest*, attained within a *local* topological neighborhood of particles.

The positions and velocities of a population of particles can be represented in vector format as follows, where *m* is the population size of the swarm, and *n* is the number of dimensions (variables) for each particle.

$$x_i = \{x_{i2}, x_{i2}, ..., x_{in}\}, i = 1,2,..., m \quad (3\text{-}3)$$

$$v_i = \{v_{i2}, v_{i2}, ..., v_{in}\}, i = 1,2,..., m \quad (3\text{-}4)$$

The population of particles' positions and velocities can be represented in matrix form as follows:

$$X = (x_i, x_2, \cdots, x_m)^T = \begin{pmatrix} x_{11} x_{12} \cdots x_{1n} \\ x_{21} x_{22} \cdots x_{2n} \\ \vdots \quad \ddots \\ x_{m1} x_{m2} \cdots x_{mn} \end{pmatrix} \quad (3\text{-}5)$$

$$V = (v_i, v_2, \cdots, v_m)^T = \begin{pmatrix} v_{11} v_{12} \cdots v_{1n} \\ v_{21} v_{22} \cdots v_{2n} \\ \vdots \quad \ddots \\ v_{m1} v_{m2} \cdots v_{mn} \end{pmatrix} \quad (3\text{-}6)$$

Fitness-based Population Diversity

In the literature, including a paper co-authored by the authors of this paper [134], the diversity of a genetic algorithm (GA) population has been calculated using the standard deviation of the individual fitness values of population members. This is, however, an indirect metric for population diversity; fitnesses are attributes of population behavior (phenotype), rather than direct diversity measures at the information-theoretic level. The diversity of a GA population

has also been calculated as the average Euclidian distance from the GA population's average location vector [135].

The authors propose to view diversity from more of an information-theoretic perspective. Such an approach should be based on something more closely related to entropy. For a particle swarm, the positions and velocities of the particles provide such a basis. In the following sections, we present basic metrics for population position diversity and population velocity diversity, and then discuss ways to combine them into a unified diversity metric for particle swarm diversity. With minor modifications, this approach is applicable to other evolutionary algorithms.

3.2.2. Position based population diversity

Element-wise Population Position Diversity

Based on previous work done on GA population diversity [135, 136], the elements of all dimensions in an individual can be equally weighted. (It should be noted that in the two cited references, each element is a binary gene, having a value of either 0 or 1, rather than a real value.)

The superscript index *p* in D^p (Equation 3-8) stands for the population position diversity with respect to the particle's position x_i. In subsequent sections, the superscript *v* stands for population velocity diversity.

$$\bar{x} = \frac{1}{n \times m} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \tag{3-7}$$

$$D^p = \frac{1}{n \times m} \sum_{i=1}^{m} \sum_{j=1}^{n} [x_{ij} - \bar{x}]^2 \tag{3-8}$$

Euclidean Distance-based Population Diversity

In this method, the Euclidean distance is measured between pairs of population members for all possible combinations.

$$\begin{aligned} d^p(x_i, x_j) &= \left\| x_i - x_j \right\| \\ \tilde{d}^p(x_i, x_j) &= \frac{d^p(x_i, x_j)}{\left\| a - b \right\|} \end{aligned} \tag{3-9}$$

$$D_{ED}{}^p = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \tilde{d}(x_i, x_j) \tag{3-10}$$

In this measure, [a, b] is the dynamic range of the particle's position, that is, the particles are limited to "fly" within the range [a, b]. This type of diversity metric was discussed in Wen *et al.* [137] and Wen *et al.*[138].

Dimension-wise Population Position Diversity

For a particle swarm, instead of considering whether the particles are close to each other by looking at a single distance measure separating them, we measure the diversity of each dimension's positions by considering the position values on each dimension of the particles in the population. In the equations below, *m* is the population size, and the subscript *I* is a

dimension which varies from 1 to *m*. Also, *n* is the number of dimensions (variables) in each particle, and the subscript *j* is an index that varies from 1 to *n*.

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$
$$D_j^p = \frac{1}{m}\sum_{i=1}^{m}[x_{ij} - \bar{x}_j]^2 \tag{3-11}$$

Therefore, we have a dimension-wise position diversity vector $(D_1^p, D_2^p, \cdots, D_n^p)$. Based on this *n*-dimensional diversity measure, there are several ways to measure the swarm population position diversity.

4) Weighted Summation Position Diversity

In the first position diversity measure we examine, the individual dimensions (parameters) are assigned different weights, resulting in the weighted summation position diversity over all dimensions as defined in (3-12).

$$D_{WS}^p = \sum_{j=1}^{n} w_j D^p{}_j \tag{3-12}$$

Each w_j is a positive value (weight) less than or equal to 1. If all the dimensions are treated equally, we have $w_j = 1/n$ for all values of *j*. Then (3-12) can be revised as (3-13) for weighted summations with equal weights.

$$D_{WSew}^p = \frac{1}{n}\sum_{j=1}^{n} D^p{}_j \tag{3-13}$$

2) Weighted Maximization Position Diversity

Another measure that might be of interest is a weighted maximum position diversity, which is the weighted maximum value calculated on any individual dimension, as indicated in (3-14).

$$D_{WM}^p = \max\{w_j D^p{}_j\} \qquad j = 1, \cdots, n \tag{3-14}$$

Again, w_j is the weight for the D_j^p position diversity for one dimension defined in (11b).

3) Position Diversity Vector Length

It may be useful to calculate the Euclidean length of the position diversity vector, as illustrated in (3-15).

$$D_L^p = \sqrt{\sum_{j=1}^{n} D_j^{p\,2}} \qquad j = 1, \cdots, n \tag{3-15}$$

4) Normalized Dimension-wise Population Position Diversities

Another method to calculate the position diversity values analogous to (12)–(15) is to normalize the position values before calculating the position diversity.

In most implementations of PSO, the particles are restricted to an initial range of values (dynamic range) for each parameter (dimension). Assuming the particle is initially limited to the dynamic range $[a_j, b_j]$ on dimension *j*, then the normalized particle position value is given in (16).

$$\hat{x}_{ij} = \frac{x_{ij}}{|b_j - a_j|} \qquad a_j \le x_{ij} \le b_j; \qquad i = 1, \cdots, m; \qquad j = 1, \cdots n \tag{3-16}$$

The normalized dimension-wise position diversity can then be calculated as in (3-17).

$$\begin{aligned} \bar{x}_j &= \frac{1}{m}\sum_{i=1}^{m} \hat{x}_{ij} \\ D_{jN}^{p} &= \frac{1}{m}\sum_{i=1}^{m} [\hat{x}_{ij} - \bar{x}_j]^2 \end{aligned} \tag{3-17}$$

The population position normalized diversity can then be obtained by using the weighted summation, weighted maximization, or diversity vector length approach defined in (3-12)–(3-15).

3.2.3. Population velocity diversity

A feature of PSO not found in other evolutionary algorithms is that there is a velocity, in addition to a position, associated with each population member (particle). Therefore, we should not only consider the population diversity from the perspective of position, but also from the perspective of velocity.

To distinguish between position and velocity, we use the superscript index *p* to represent diversity with regards to the particles' positions, and the superscript index *v* to represent diversity with regards to the particles' velocities.

Velocity has two components: speed and direction. We consider particles' velocity from two perspectives: dimension by dimension, where we calculate speed diversity; and particle by particle, for which we calculate direction diversity. Since particles move along dimensions, it is logical to use intra-dimensional movement to calculate speed diversity. However, there is no intra-dimensional direction diversity. (See (1) and (2), from which it can be seen that calculations are done on a dimension-by-dimension basis.) We calculate direction diversity using the difference in angle between each particle's overall velocity vector and the average velocity vector for the swarm population.

Dimensional Speed Diversity

To calculate the dimensional speed diversity, each particle's speed is first normalized using the unit length over all population members along a single dimension *j*.

$$v_{ij}^{nor\_\dim} = \frac{v_{ij}}{\sqrt{\sum_{i=1}^{m} v_{ij}^{\ 2}}} \tag{3-18}$$

Then in a manner similar to (11), we calculate dimension-wise speed diversity, as shown in (3-19), where D_j^{v-ds} is the speed diversity on dimension *j*.

$$\bar{v}_j^{nor\_\dim} = \frac{1}{m}\sum_{i=1}^{m} v_{ij}^{nor\_\dim}$$
$$D_j^{v\_ds} = \frac{1}{m}\sum_{i=1}^{m}[v_{ij}^{nor\_\dim} - \bar{v}_j^{nor\_\dim}]^2 \quad (3\text{-}19)$$

Weighted Summation Dimensional Speed Diversity

In a manner similar to (3-12), we can calculate the weighted summation dimensional speed diversity over all dimensions.

$$D_{WS}^{v\_ds} = \sum_{j=1}^{n} w_j D_j^{v\_ds} \quad (3\text{-}20)$$

As before, we define w_i as a positive value less than or equal to 1. If all the dimensions are treated the same, we obtain $w_i = 1/n$. Then (20) can be written as (3-21).

$$D_{WSew}^{v\_ds} = \frac{1}{n}\sum_{j=1}^{n} D_j^{v\_ds} \quad (3\text{-}21)$$

Weighted Maximization Dimensional Speed Diversity

Analogous to the position diversity calculation of (3-14), we can calculate the weighted maximum dimensional speed diversity, shown in (3-22).

$$D_{WM}^{v\_ds} = \max\{w_j D_j^{v\_ds}\} \qquad j = 1,\cdots,n \quad (3\text{-}22)$$

Each w_j is the weight for $D_j^{v\_ds}$ as defined in the weighted summation calculation of (3-20).

Dimensional Speed Diversity Vector Length

It may be useful to calculate the Euclidean length of the dimensional speed diversity vector, as illustrated in (3-23).

$$D_L^{v\_ds} = \sqrt{\sum_{j=1}^{n} D_i^{v\_ds\,2}} \qquad i = 1,\cdots,n \quad (3\text{-}23)$$

Particle Direction Diversity

To calculate the particle direction diversity, each particle's velocity elements are normalized using the unit length over all dimensions for that particle as shown in (3-24).

$$v_{ij}^{nor\_par} = \frac{v_{ij}}{\sqrt{\sum_{j=1}^{n} v_{ij}^{\,2}}} \quad (3\text{-}24)$$

We now calculate the normalized average velocity vector for the population of particles. We first average the velocity elements on each dimension, as shown in (3-25). This gives us the dimension-by-dimension elements of the average velocity vector.

$$v_j^{\dim\_avg} = \frac{1}{m}\sum_{i=1}^{m} v_{ij} \quad (3\text{-}25)$$

We then calculate the normalized average velocity vector elements (dimension-by-dimension) as shown in (3-26).

$$v_j^{nor\_avg} = \frac{v_j^{\dim\_avg}}{\sqrt{\sum_{j=1}^{n} v_j^{\dim\_avg\,2}}} \tag{3-26}$$

We can now calculate the cosine of the angle θ between each normalized particle velocity vector and the normalized average particle velocity vector as in (3-27). The particle population direction diversity is then defined in (3-28).

$$\cos(\theta_i) = \sum_{j=1}^{n} \left(v_{ij}^{nor-par} \bullet v_j^{avg\_nor} \right) \tag{3-27}$$

$$D^{v\_dir} = \frac{1}{m} \sum_{i=1}^{m} \theta_i^{\,2} \tag{3-28}$$

3.2.4. Discussion

We should consider the position, speed, and direction diversities together, not only one or two of them at a time. There is a trade-off between position based population and velocity based population diversities. A way to calculate the overall diversity is shown in (29).

$$D = w_p D^p + w_{v\_ds} D^{v-ds} + w_{v\_dir} D^{v-dir} \tag{3-29}$$

The quantity D^{v-ds} can be any of the speed diversity metrics such as D_L^{v-ds} of (23). Without a loss of generality, the velocity, speed and direction diversities can be treated equally, that is, the weights w_p, w_{v_ds} and w_{v_dir} can be set to be equal. Other values are possible and will depend on the problem to be solved. The weights can also be a function of PSO performance and the diversities themselves. They can also be adjusted or evolved to reflect dynamic changes during the PSO search process.

Pair-wise calculations of diversities (position, speed, and/or direction) similar in concept to the calculation of (10) for distance-based population diversity would be possible. However, the approaches we have used, such as (28), are significantly simpler and don't become computationally intensive for realistic problems. We acknowledge that some users might prefer, for particular reasons, to make pair-wise calculations for all three diversity measures, but we believe that the methods we propose will prove more effective for most applications.

Other metrics are enabled by the three diversity measures we propose. For example, the standard deviations of the diversity measurements may provide insight into system performance, and the optimal values of these metrics may also change as iterations progress.

This method is also adaptable to other evolutionary algorithms such as genetic algorithms. For example, although GAs do not have a velocity, the measurement of their population "velocity" diversity can be approached by considering the changes in location of population members from one generation (iteration) to the next in a manner analogous to the velocities in PSO.

This methodology for measuring diversity can facilitate many areas of investigation for PSO and other evolutionary algorithms. Among them are:

a. How should diversity be managed?

b. How does varying parameters and features of PSO such as using the global versus local model, adjusting V_{max}, c_1, c_2, etc. (see Equations 1 and 2) affect diversity?

c. What is the best way to increase or decrease diversity by a predictable quantity?

d. How should each of the three metrics be weighted?

e. Which (if any) statistical attributes of the three metrics are significant and useful?

3.2.5. Conclusion

It is important to note that optimal diversity varies with the problem and over the course of a run of an evolutionary algorithm such as PSO. There has been much discussion in the literature of the importance of population diversity. This paper proposes a method to measure it for particle swarm optimization. The method is adaptable for use with other evolutionary algorithms.

The measurement of diversity is only the first step in the eventual management and optimization of diversity. We have much to learn regarding how to determine optimal diversity values, and how these values vary over the process of a run of an evolutionary algorithm. In order to manage diversity, however, we must be first able to measure it. This paper proposes a method to provide this first and important step.

3.3. Human in the swarm: an NK landscape game

3.3.1. Background

Computers are faster, and in some ways more powerful, than human beings. On the other hand, human beings are generally better at learning, making decisions, and adapting to dynamic environments. In this paper, we examine ways to combine the strengths of computers and humans to solve real world problems. Specifically, the objective of this paper is to begin to investigate how humans interact with swarm intelligence. The swarm intelligence paradigm used is particle swarm optimization (PSO) (Eberhart and Kennedy 1995, Kennedy and Eberhart 1995, Kennedy *et al*. 2001).

Kennedy (2001) designed particle swarm optimization programs using pattern matching with NK landscapes (Kauffman 1995) to study how humans perform as particle. His preliminary results show that the introduction of a human only marginally (at best) helps find an optimum and the swarm always beats the human. However, the underlying assumption is that the human and the particle swarm have the same amount of information about the problem. What if the human has more knowledge than the particles? Following is an introduction to an NK-landscape game developed by the authors which is an extension of the game described in Kennedy's paper.

3.3.2. Methods

The NK landscape game program first defines a globally optimal bitstring. This bitstring is random for the Hard level of the game, and complies with constraints associated with the Medium and Easy levels of the game (levels are described below). The global bitstring determines whether each square on the checkerboard pattern will be red or white. Then the program defines K connections for each node (square), makes sure that each node has inputs from exactly K other nodes, and that each node connects to exactly K others. (It randomly assigns them rather than using adjacent neighbors as is usually done by Kaufmann (1995).)

Based on the state (color) of each node (square) and the K nodes connected to it the program has established the local pattern which corresponds to the global optimum. This local pattern is assigned a fitness value of 1.0. So if a square is the correct color and all of the squares upon which it depends are the correct color, the square's bitstring matches the global optimum, and that square (node) will return a fitness value of 1.0.

The fitness values of non-optimum bitstrings are determined using a "short-cut" method (Kennedy 2001, Kennedy 2007). A short-cut is desirable because as N and K increase, fitness evaluation tables grow exponentially. Therefore, in this game a random number seed is generated for each non-optimal bitstring based on a random integer generated when the program is initialized, the node's index, and the binary value of the node and its K inputs. Fitness values are then generated by seeding the random number generator and reading its output. (A similar approach was described by Lee Altenberg in the 1997 Handbook of Evolutionary Computation.)

Note that if K=0, creating a unimodal landscape, hill-climbing algorithms will always find the optimum by flipping bits and keeping the new value if the fitness improves. As K increases, complexity increases quickly, and even low values of K result in difficult problem surfaces. Local optima quickly complicate the error surface, and finding solutions is difficult even with K=3 (the default value in the NK Landscape Game).

After selecting the game configuration in Target Settings (rows, K neighbors, and difficulty level) and PSO settings (20 particles with a neighborhood size of 3 is default), a player makes a move by clicking on one or more squares in the top left box (see Figs. 1-3) to change square color. When the player is satisfied, the "Step" box is checked, and the program accepts the player's input and implements one iteration of the particle swarm algorithm (assuming 1 iteration per step is selected). The program can also execute 10 or 100 iterations of PSO per human input step by clicking on the appropriate radio button. The lower left box is the player's own previous best pattern, and the lower right box is the swarm's best (this may be either the human particle's best, or the best of one of the particles managed by the program).

Three scenarios have been implemented in the NK-landscape pattern matching game as shown in Table 6. In each game session, a target pattern is randomly generated based on the levels and conditions defined in Table 6. Particle swarm optimization (PSO) is used to find the matched pattern based on the NK-landscape information. The particle swarm does not know how the pattern is generated beyond the fact that the pattern is completely random, i.e., the additional constraints in the easy and medium levels are not coded into the PSO fitness function. However, the human knows how the patterns are generated. For example, at the easy level, the human user knows that only one block in each column is red, and all blocks are linked to each other (the pattern is a spline).

Table 6 NK Landscape Game Levels

Levels	Target Pattern
Easy	Only one red block in each column, and only spline patterns.
Medium	Multiple red blocks, but no isolated links; spline patterns only.
Hard	Totally random patterns in N*N blocks

Figure 9, Figure 10, and Figure 11 show screen shots taken during a game session with the computer-generated pattern to be matched by the human shown by checking the "show target" box. The "show target" box is generally not shown during a game session. It is used for player orientation to the game, or if a player gives up and wants to see the correct pattern.

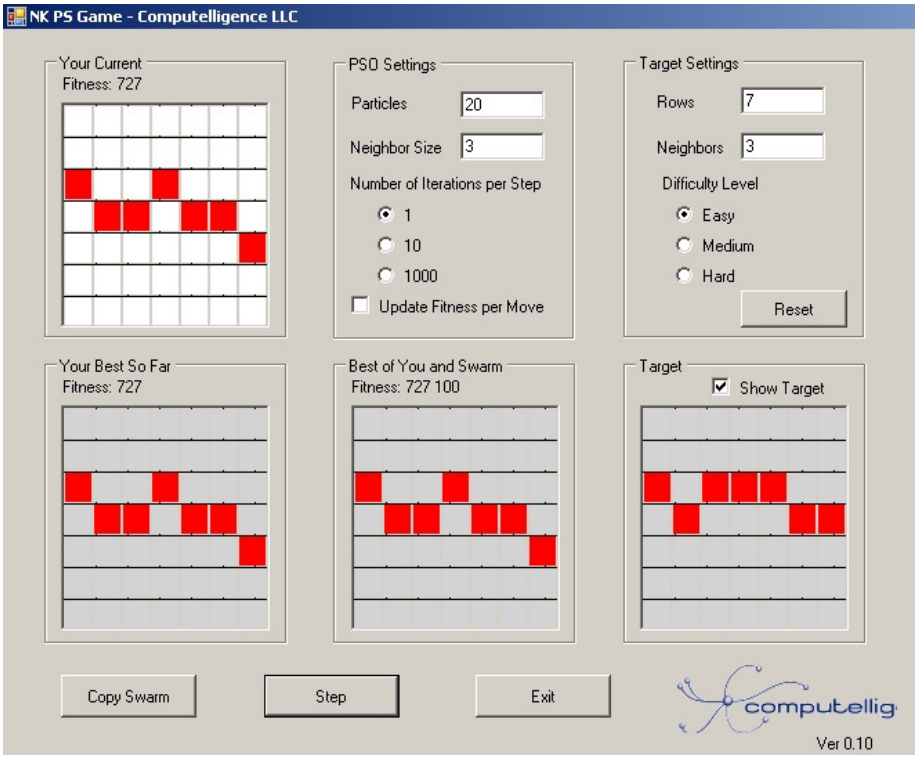

Figure 9 NK landscape game screenshot for the Easy level

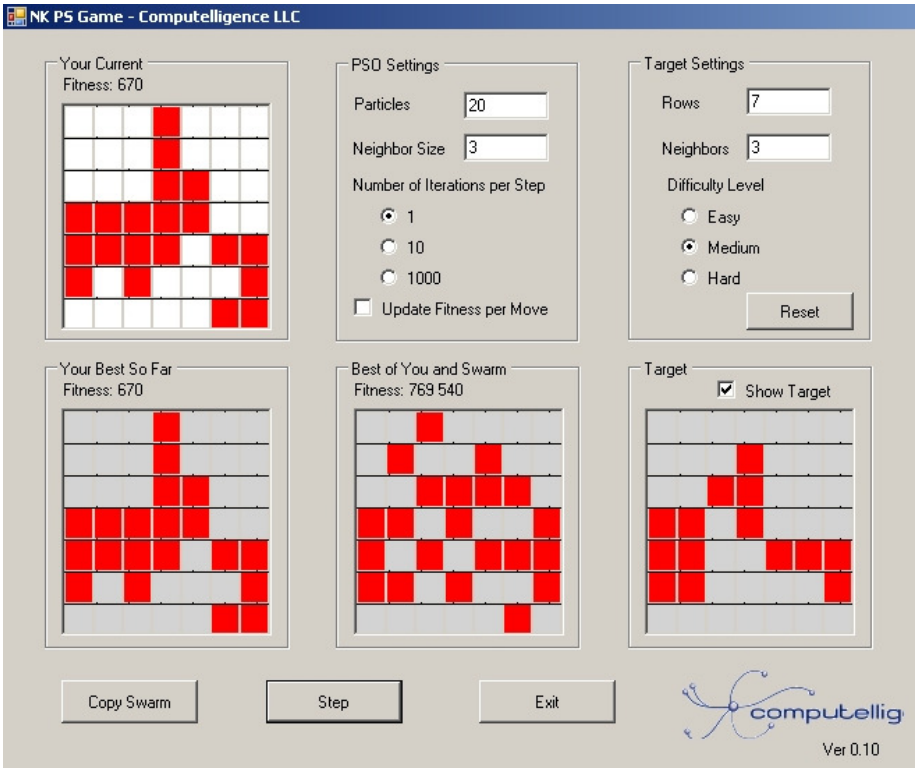

Figure 10 NK landscape game screenshot for the Medium level

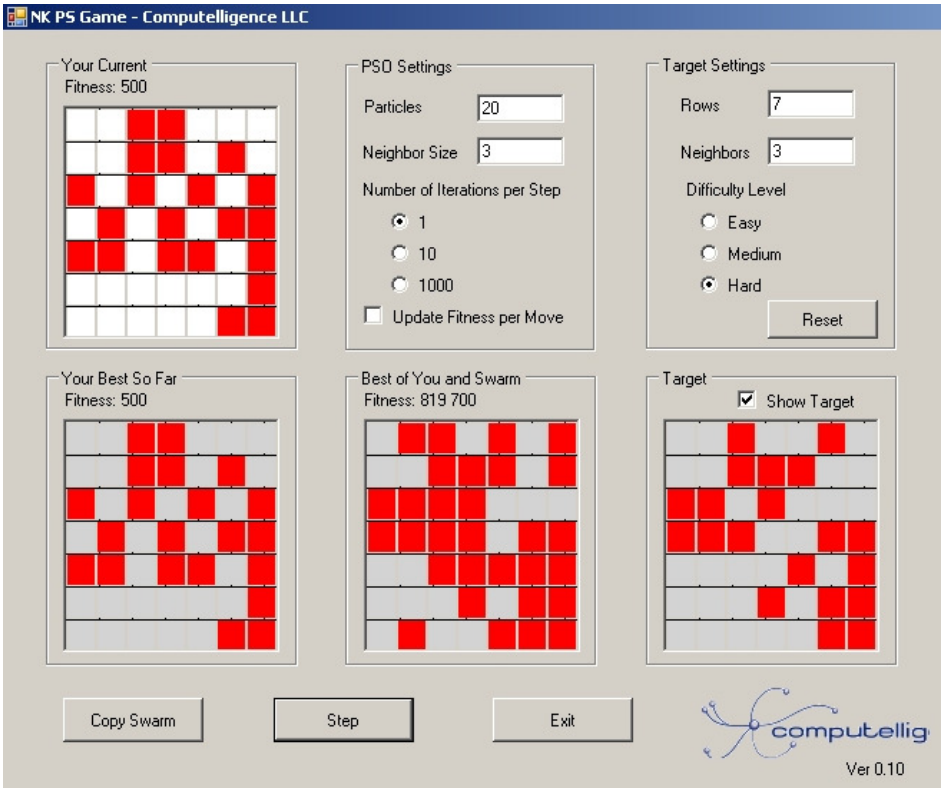

Figure 11 NK landscape game screenshot for the Hard level

3.3.3. Results

In a test of the game, the PSO settings were as follows: population size was 20, neighbor size was 3, c1 and c2 were 1.49445 and w was [0.5+Rnd/2.0]. The NK-landscape parameters were, N = 7 and K = 3. Ten runs were executed for PSO without a human particle, and for PSO with a human particle. The results are displayed in Table 7. Results depend on the skill of the player, but preliminary indications are that the results in Table 7 are typical of those obtained by a computer-savvy player adept at electronic games.

Table 7 Performance comparison between pure PSO and human-in-the loop PSO

(average iterations used to find the pattern over 10 runs)

Levels	Without Human Particle	With Human Particle	Who found solution
Easy	146.6	36.9	Human 9 – Swarm 1
Medium	103.8	69.0	Human 2 – Swarm 8
Hard	90.0	-	Human 0 – Swarm 10

3.3.4. Discussion

From the results, it can be seen that:

1. When human has significant knowledge that swarm does not have, the human-swarm team can beat the swarm-only-team.

2. The less random the pattern, the slower the particle swarm finds the optimum.

3. If the extra knowledge that the human has is easy to implement, the human can usually beat the computer (the human was first to find the solution 90% of the time in the Easy case). If the

extra knowledge the human has is hard to implement, the computer will usually beat the human (the computer was first to find the solution 80% of the time in the Medium case).

3.3.5. Conclusion

If the swarm knows the constraints for the Easy and Medium levels, it might solve the problem faster than a human. However, in order to incorporate constraints there are three challenges that need to be addressed. First, the constraints must be encoded in software, which sometimes is difficult. Second, the algorithm may require significantly more computation to deal with the complexity added by those constraints. Third, constraints may appear dynamically in a way relatively obvious to humans but not to computers, and recoding to include them during a run is not always feasible.

Thus, the combination of the human-swarm team may have advantages in certain environments, such as dynamic decision making tasks. The team approach can combine computer computational power with human intuitive knowledge to provide fast response for dynamic and complex tasks.

4. Case Study – UAV scheduling

4.1. Background

With the innovation of advanced material, engine and sensor technology, it improves the tactics performance of Unmanned Air Vehicles (UAVs); the rapid development of communication, information and image processing technology make it efficiently to deal with large amounts of information and data; artificial intelligence and computational intelligence methods enable UAV more intelligent, and better adapt to the changes in the battlefield. The roles and complexities of UAV are growing and research approaches are evolving.

UAV assignment and task allocation problems, among the most studied topics, decompose the optimization of air-to-ground operations into several parts and study how to allocate and schedule the UAVs to perform tasks so as to maximize effectiveness of the overall mission., involving different geographical locations, goal assignment, trajectory optimization, and time or task requirements., The complexity and computation cost are dramatically increase with the increase of problem size[139, 140]. 'Multi-tasking Multi-assignment problem' consists of two aspects: modeling and solving task assignment problem[141].

(1) Modeling of the multi-task and multi-assignment problem.

Based on classic assignment and scheduling problems, some models are adapted in the research of UAV assignment problem, including Multiple Traveling Salesman Problem (MTSP)[142, 143], Vehicle Routing Problem (VRP);, Mixed-Integer Linear Programming (MILP)[140, 144, 145], Dynamic Network Flow Optimization (DNFO)[146], Multiple Processors Resource Allocation (MPRA), and so on.

(2) Solving of the multi-task and multi-assignment problem.

Approaches applied to solve the UAVs assignment problems can be classified into two categories: deterministic and stochastic methods. Under some assumption, the assignment problems are simplified as a certain type mathematic model (e.g. linear programming model), and those deterministic methods, well-developed in the optimal theory and operational research, such as Hungarian algorithm[147],dynamic programming[148], branch and bound and decision tree[149], etc, can arrive at optimal or good decisions efficiently. However, known as a typical NP-problem, the complexity and computation cost rapidly aggravate subject to the complexity increase of the assignment problem.

Stochastic methods, especially heuristic algorithms, don't spend polynomial time cost in solving assignment problem with the combinatorial complexity. They try to optimize a global or sub-optimal in a limit computation cost. Genetic Algorithm[150-152], Ant Colony Algorithm [153], Tabu Search[140, 143, 154], Auction Algorithms[139], Particle Swarm Optimizer[142], etc, are widely utilized to achieve the optimization task.

4.2. Problem description

Obviously, the UAV assignment problem is similar and related to Traveling Salesman Problem (TSP) and Linear ordering Problem (LOP). There are some popular models of them are introduced to the assignment problems, such as the integer programming formulation of TSP proposed by Dantzig-Fulkerson-Johnson and Miller-Tucker-Zemlin[150].

The Mixed Integer Linear Programming (MILP) model, one of the most studied and widely recognized models, is applied to describe the UAV assignment problem and GLPK program package in [144]. A scenario, with N targets, M UAVs and K tasks, is studied. There are $N(N-1)MK+NMK+2NM+NK+2M+1$ decision variable, which include $3+NK+1$ continuous nonnegative variables. In all, there are $12(N-1)NM+9NM+2NMK+2NK+3M$ constraints, including $KN+M$ equality constraints, $7NM+2M$ inequality non-timing constraints and the rest are inequality timing constraints. The size of the MILP model grows rapidly with the increase of problem size. So it is a big challenge to deal with more complicate and constraint problem. In this report, based on swarm intelligence theory, a computation technique, termed Particle Swarm Optimizer (PSO) is applied to the UAV assignment problem in [144]. Firstly, the problem is analyzed and the model is re-presented so as to adapt the requirement of PSO.

Assumption:

(1) The number of tasks are the same as those in [144]. In another word, three tasks (classification, attack and verification) or two tasks (without the classification tasks) will be implemented on each target. In [139], there is another task, named 'search task', which will be considered in the future research.

(2) All UAVs are homogeneous. That means all vehicles can perform all tasks on all targets, but they have different time cost.

(3) The number of targets and UAVs obey the requirement in [144], which means the number of UAVs is no less than targets.

(4) All targets have the same priority. In paper [143], the priority difference is considered, which will be considered in our future research.

Definition: The objective function can be defined as:

$$\min\{\max_{1\le i\le N,1\le j\le M} C_{N.M}(i,j)\} \tag{4-1}$$

subject to all the constraints which will be expressed in the following ***Constraints***[144]:

1) Mission completion requires that all three tasks are performed on each target exactly one time;
2) Not more than one UAV is assigned to perform a specific task on a specified target;
3) An UAV, coming from the outside, can visit the target at most once; if an AV entered target (node) to perform a classification, it can perform the 'attack' task on the same target;
4) UAV leaves a node at most once;
5) An UAV can be assigned to attack at most one target and cannot also be assigned to perform any other tasks at targets;
6) If UAV enters a target (node) for the purpose of performing 'classification' or 'verification' task, it must also exit the target;
7) If UAV is not assigned to visit node, then it cannot possibly be assigned to fly out of node;

8) All UAVs leave the source nodes. An AV leaves the source node even if this entails a direct assignment to the sink;

9) A UAV can leave a node, unless it completes the task at the node;

10) A UAV can perform a task, only if the preceding task at the node has been completed.

All variables are explained in Table 8.

Table 8 Nomenclature

N	Number of targets	K	Number of tasks for each target
M	Number of UAVs	$C_{N,M}$	Objective time Matrix
T_{ij}	Fly time matrix between node i and j.	$T_{n,m,k}$	Time matrix of task completion
S_{target}	Visiting Target Sequence Array	A_{UAV}	UAVs Assignment Matrix

4.2.1. Objective time matrix $C_{N,M}$

The objective time matrix $C_{N,M}$ (as shown in Equation 4-2) contains a cumulative time accounting of the mission time tasks. Each row in the $C_{N,M}$ matrix corresponds to a target, and each column corresponds to a UAV. When a task of the *n*-th target is completed by the *m*-th UAVs, the cumulative completion time $c_{n,m}$ is calculated by Equation 4-3.

$$C_{N,M} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,M} \\ \cdots & c_{n,m} & \cdots \\ c_{N,1} & \cdots & c_{N,M} \end{bmatrix}_{N \times M} \tag{4-2}$$

$$c_{n,m} = t_{n,m,k} + \max\{c^{\max}_{n-Row}, c^{\max}_{m-Column}\}, t_{n,m,k} \in T_{n,m,k} \tag{4-3}$$

$$c^{\max}_{n-Row} = \max\{c_{n,i} \mid i = 1, 2...M\} \tag{4-4}$$

$$c^{\max}_{m-Column} = t_{i,j} + \max\{c_{j,m} \mid j = 1, 2...N\}, t_{i,j} \in T_{i,j} \tag{4-5}$$

$c^{\max}_{m-Column}$ means after the *m*-th vehicle completes all its own preceding tasks, it will takes $t_{i,j}$ to arrive at the *n*-th target. $c^{\max}_{n-Row}$ means the new task at the *n*-th target has to wait all tasks so far have been completed. $t_{n,m,k}$ stands for the time cost of the *m*-th vehicle performs the *k*-th tasks on the *n*-th target. The largest value appearing in any cell of matrix $C_{N,M}$ is thus the maximum elapsed time for the mission, which is the maximum time taken by any of the UAVs. This time is the fitness function, and is the value which is being optimized (minimized) by the PSO algorithm. The operations of $C_{N,M}$ meet the time constraints.

4.2.2. Visiting target sequence array S_{target}

The Visiting Target Sequence Array S_{target} (VTSA, shown as), an integral row vector, indicates the order that the targets should be visited. There are $N \times K$ elements in S_{target}, and every element s_i is the target number and stands for the time precedence of targets. Since M tasks should be implemented on each target, each target number should appear M times which

correspond to the tasks in order, such that the first time that a target number appear is corresponded to the first task of this target.

$$S_{target} = [s_1, s_2 \ldots s_{N \times K}] \tag{4-6}$$

4.2.3. UAVs assignment matrix A_{UAV}

The UAV assignment matrix A_{UAV} is a K by M matrix (see Equation 4-7). Each row stands for a task and each column stands for a target (illustrated in Figure 12). $v_{k,n}$ is the k-th row and the n-th column element in A_{UAV} and stands for UAV number, which performs the *k*-th task of the *n*-th target. (Here, using the same letter *v* in [144] to stand for vehicles).

$$A_{UAV} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,M} \\ \cdots & v_{k,n} & \cdots \\ v_{K,1} & \cdots & v_{K,M} \end{bmatrix}_{K \times M} \tag{4-7}$$

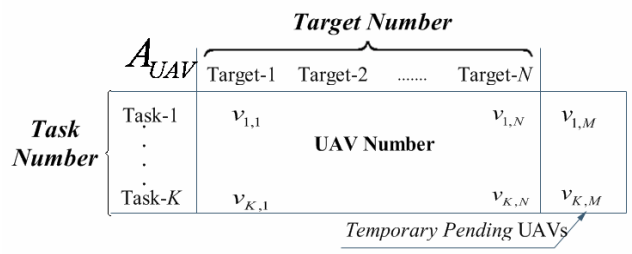

Figure 12 Illustration of UAV assignment matrix

The calculation of objective time matrix $C_{N,M}$ is based on visiting target sequence array S_{target} and UAV assignment matrix A_{UAV}. Every vehicle assignment in A_{UAV} should satisfy rigid constraints. The pseudo-code to generate assignment matrix is shown as Figure 13. The 5th constraints have the highest priority in the assignment matrix, so vehicles which will implement "ATTACK" task are assigned, firstly. According some selection methods that it can be designed through many ways, the feasible vehicle is selected from the feasible UAV array. The available vehicle list includes all UAVs which satisfy all constraints at that time and is illustrated in Figure 14.

```
Step 1:  Assign different UAVs to perform 'ATTACK' task at all target, and set them
         in the row corresponding to 'ATTACK' task of the assignment matrix A_UAV

Step 2:  do:
           { current_target_number = S_target(order) ;
             current_task_number = the times that current_target_number has
                                   been visited before the 'order' ;
             if current_task_number is not 'ATTACK'
                   Generate feasible UAV arrayy for current_target_number;
                   current_UAV_number = get a UAV from feasible UAV array ;
             end

             set   current_UAV_number as the element value in A_UAV ,
                   whose row number is current_task_number
                   and column number is current_target_number ;

             order = order+1 ;
           }
         while ( order < NK )

Step 3: All UAVs, except assigned 'ATTACK' task, fly to sink node
```

Figure 13 Pseudo-code to generate assignment matrix A_{UAV}

```
sub-function: Generate feasible UAV array for current_target_number
    Input variables:
        {   implementation 'order'
            current_target_number
            current_task_number          }
    Operation:
        Initiate feasible_UAV_array = [1,2.....M]
        for  m =1 to M
            if m is assigned to perform 'ATTACK' task befor the current time i
                delete m from feasible_UAV_array ;
            end
            if current_task_number is 'CLASSIFICATION'
                if m performed 'CLASSIFICATION' at another target before 'order'
                    &&  m performed 'ATTACK' at the same target after 'order'
                        delete m from feasible_UAV_array ;
                end
            end
            if current_task_number is 'VERIFICATION'
                delete the UAV number which performs the 'CLASSIFICATION'
                at 'current_target_number', from feasible_UAV_array ;
            end
        end
    Output:
        feasible_UAV_array ;
```

Figure 14 Sub-function to generate feasible UAV array

4.2.4. Discussion and explanation

There are $K \cdot N + K \cdot M$ decision variables of the proposed model in this section, including $K \cdot N$ variables of visiting target sequence array and $K \cdot M$ UAV assignment matrix. Based on the operation of 'Step 1' in Figure 13, UAVs, which will have the 'ATTACK' mission, are assigned first and it is generated by the sorted particle vector of PSO (discussed in the 4th section), so there is no constraint here. In Figure 14, each vehicle has $2M$ constraints. There are $(K-1)N$ vehicles, performing 'CLASSIFICATION' or 'VERIFICATION'. So, in all, there are $2(K-1)NM$ constraint variables.

Explanation: A simple scenario (2-targets, 3-tasks and 3-UAVs) goes as follows to explain the aforementioned model.

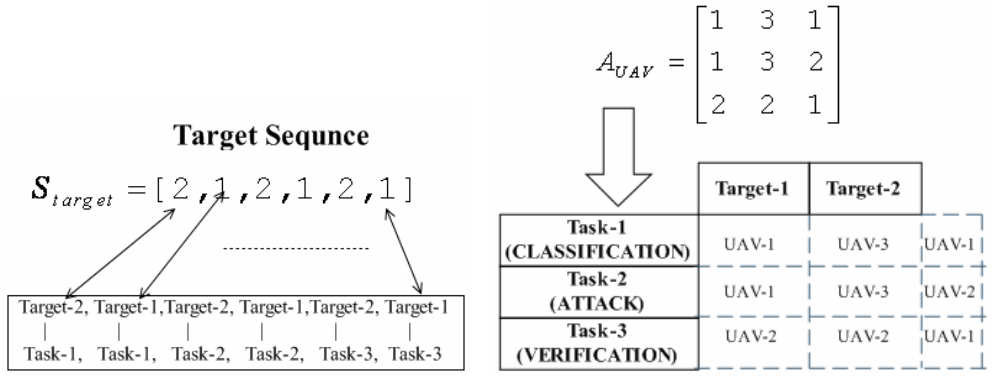

Figure 15 Illustration of the target sequence(a) and the assignment matrix (b)

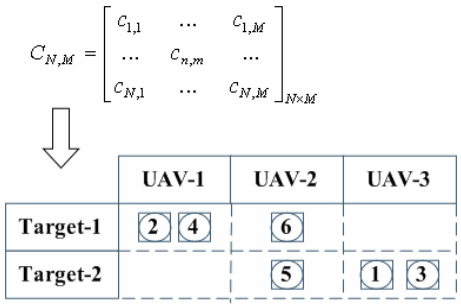

Figure 16 Calculation order of the objective time matrix

The numbers in Figure 15 (a) then are interpreted as follows: First implement the first task for the 2nd target, then the first task for the 1st target, then the second task for the 2nd target, then the second task for the 1st target, and so on.

The matrix in Figure 15 (b) is interpreted as follows: The top row corresponds to the first task ('CLASSIFICATION', in this scenario), while the middle row corresponds to the second task ('ATTACK') and the third row corresponds to the third task ('VERIFICATION'). Each column corresponds to a target. The first column corresponds to the first target, etc. (Note that there is one more column than there are targets. In this case, the last column on the right is ignored.) .

The objective time matrix $C_{N,M}$ in Figure 16 contains a cumulative time accounting of the mission time tasks. Each row in the $C_{N,M}$ matrix corresponds to a target, and each column corresponds to a UAV. The values in the second column then are for UAV 2, and will show cumulative increasing values as the UAV verifies destruction of targets 1 and 2, in that order. Cumulative time cost is calculated by Equation 4-3. The largest value appearing in any cell of matrix $C_{N,M}$ is thus the maximum elapsed time for the mission, which is the maximum time taken by any of the UAVs.

The matrix values are combined with the string of numbers of target sequence array in Figure 15 (a) to form a schedule. Thus, in this case, the first task on the 2nd target is implemented with UAV number 3 and the time cost ① is filled in Figure 16. Then the first task on the 1st target is

implemented with UAV number 1 and ② is calculated, followed by implementing the second task on the 2nd target with UAV number 3 and ③ is accumulated, followed by implementing the second task on the 1st target with UAV number 1, and so on. The last task in the schedule is to implement the third task on the 1st target with UAV number 2.

4.3. Constraints treatment

Generally, the constraints of UAV assignment problem are derived and represented as equality and inequality equations, so as to in accordance with the requirement of the models and optimization methods, such as Mixed-Integer Linear Programming whose advantages are the problem and constraints are organized as standard mathematic equation and the optimal can be achieved. On the other hand, it will leads to variables size explosion with the increase of problem complexity. Aforementioned, according to the model definition in Equation 4-1, constraints and data structure in the 2nd section, constraints can be classified into two categories: rigid constraints and soft constraints (see Table 9). To deal with the rigid constraints, it is necessary to design some techniques which will check and adjust potential solutions to meet the constraint's requirements at each iteration. Contrarily, the soft constraints will always be satisfied, since some operations have been applied to guarantee the constraints in the process of model configuration.

Table 9 Constraint classification

	Constraints in the report	Constraints in [144]
Rigid constraints:	the 3rd, 4th, 5th, 6th, 9th, 10th;	the 1st, 2nd, 3rd, 5th, 7.3th, 7.6th, 9th;
Soft constraints:	the 1st, 2nd, 7th, 8th;	the 4th, 6th, 7.1th, 7.2th, 7.4th, 7.5th, 8th,10th;
Constraints for further research:	---	the 11th, 12th, 13th;

4.3.1. Soft constraints treatment

The 1st constraint: In Equation 4-7 and Figure 12, there are only K rows of A_{UAV}, which is equal to the task number and stands for K tasks of a target separately.

The 2nd constraint: In Equation 4-7 and Figure 12, each element of A_{UAV} correspond to a UAV number. There is exactly one UAV assigned to perform a task at a target,

The 7th constraint: In Equation 4-2 and 4-4, the time-of-flight is accumulated only based on the preceding visited node. It means vehicles can not fly out of a node, unless it executes a task at the node.

The 8th constraint: In Equation 4-5, it means a vehicle either fly to implement another task at a node, or just fly to sink node directly, which also could be considered as a mission of a vehicle.

4.3.2. Rigid constraints treatment

The 3rd constraints: in Figure 14, it limits that the UAV m can not perform a 'CLASSIFICATION' at another target before the current implementation order in S_{target}, and also assigned to perform 'ATTACK' at the same target after the current implementation order in S_{target}

The 4th constraints: in Figure 14, the vehicle which has visited a target for 'CLASSIFICATION' or 'ATTACK' is not allowed to allocate to 'VERIFICATION'.

The 5th constraints: in Figure 14, any vehicles, performed 'ATTACK' before the current implementation order in S_{target}, are not feasible to other tasks or targets.

The 6th constraints: in Figure 13, the 'Step 3' operation enforces all vehicles, except those allocated to 'ATTACK', has to fly to sink node.

The 9th and10th constraint are guaranteed by Equation 4-4 and 4-5, separately.

4.4. PSO application

UAV assignment problem is a typical scheduling problem with constraints. It is a challenge to solve it efficiently in subject to large number of constraints, local minimum, difficulties for construction feasible solutions, etc. Several version PSO algorithms have been applied on the test problems, according to their own features. Experiments results demonstrate all PSO algorithms exhibit a good performance and very effective, moreover the classic PSO with 2-Neighborhood information topology has the best performance and good global search ability. Potential solutions are separately optimized by two swarms by Bi-PSO and Bi-AFPSO and combined together to calculate the fitness. This is a kind potential method to reduce the dimension of solution space which is divided into two parts and searched separately.

Table 10 PSO algorithms

Algorithms	Information Topology	Swarm Quantities	Encoding
Classic PSO	Global version	1	Real value
Classic PSO	2-Neighborhood	1	Real value
AFPSO	Global version	1	Real value
Bi-PSO	Global version	2	Real value
Bi-AFPSO	Global version	2	Real value

4.4.1. Fitness function

The cumulative time cost is stored in the objective time matrix $C_{N,M}$ (Equation 4-2). and the largest value appearing in any cell of matrix is thus the maximum elapsed time for the mission, which is the maximum time taken by any of the UAVs. This time is the fitness function, and is the value which is being optimized (minimized) by the PSO algorithm.

4.4.2. Encoding

The position vector X (Equation 4-8) of a particle in PSO is a 1 by $(K \cdot N + K \cdot M)$ real value vector, which is consisted of two parts: the first $K \cdot N$ items of X (as Equation 4-9. shown) corresponding to the target sequence array (Equation 4-6) and the last $K \cdot M$ variables (as Equation 4-10 shown) standing for assignment matrix (Equation 4-7).

$$X = [x_1, ..., x_n, ..., x_{K \cdot N}, x_{K \cdot N+1}, ..., x_v, ..., x_{K \cdot N+K \cdot M}] = [X_S, X_A] \quad (4\text{-}8)$$

$$X_S = [x_1, ..., x_n, ..., x_{K \cdot N}]_{1 \times K \cdot N} \quad (4\text{-}9)$$

$$X_A = [x_{K \cdot N+1}, ..., x_v, ..., x_{K \cdot N+K \cdot M}]_{1 \times KM} \quad (4\text{-}10)$$

X_s will be sorted firstly and then the modules of these sorted serial numbers will be calculated to acquire the target sequence array. An example is given as bellow to explain it.

$$X_S \xrightarrow{\text{sort}} X_S^{'} \xrightarrow{\text{module}} S_{target}$$

Exp: 4-targets, 5-UVAs, 2-Tasks

X_S=[-0.2339 42.7231 101.5801 -13.8674
96.4728 1.3992 -12.5743 -19.9650]

$X_S^{'} = sort(X_S) = [8,4,7,1,6,2,5,3]$

$S_{target} = \text{mod}(X_S^{'}) + 1 = [1,1,4,2,3,3,2,4]$

Figure 17 Example of target sequence array encoding

X_A can be firstly organized as a K by M matrix $X_A^{'}$ (see Equation: 4-11), $X_A \rightarrow X_A^{'} \rightarrow A_{UAV}$.

$$X_A^{'} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,M} \\ \cdots & x_{k,m} & \cdots \\ u_{(K-1)\cdot M+1,1} & \cdots & u_{K\cdot M} \end{bmatrix}_{K\times M} \tag{4-11}$$

In Equation 4-11, the row vector, corresponds to the attack task, is sorted. (In Figure 18, it is the first row, since there are only two tasks in the paper). This is the 'Step 1' in Figure 13. In the sorted list, the order of the first N elements is preserved, according to the N targets. See Figure 18. Moreover, it guarantees the 5th constraints that a vehicle can be assigned to attack at most one target. In the feasible UAV array (described in Figure 14) the vehicle whose real value in X_A is the largest, , will be selected in the assignment process of other tasks.

$X_A \rightarrow X_A^{'} \rightarrow A_{UAV}$

Exp: 4-targets, 5-UVAs, 2-Tasks

$$X_A^{'} = \begin{bmatrix} 91.5188 & -63.1606 & 101.7629 & 97.6368 & -12.932 \\ -9.3305 & -7.5982 & -66.4578 & 76.6219 & 94.1288 \end{bmatrix}$$

$sort(1^{st}\ row\ of\ X_A^{'}) = [2,5,1,4,3]$,

so the relationship is as follows

	Target-1	Target-2	Target-3	Target-4
'Attack'	UAV-2	UAV-5	UAV-1	UAV-4

Figure 18 Example of assignment matrix encoding

There are two advantage of the proposed encoding method in the report. First, real value number is utilized so that it is very easy to realize and calculate. Second, since it is based on the sort of particle vector to transfer real value vector to integral sequence array and assignment matrix, there is no any value limitation and initial requirement of the particle position vector.

4.5. Experiments results

The classic PSO with 2-Neighborhood information topology, in term local version PSO, exhibits the best performance in all the three scenarios. Various scenarios with different target and UAV size, task requirement, are studied in the section. An assumption is made in advance that the time of attack delay and flight time to the sink node are not taken into account. The assumption makes

the fitness (Equation 4-1) of the scenarios with 3 tasks requirement is the same as those with 2 tasks requirement, so as to verify the optimization results for both of them.

4.5.1. Experiment 1

In the experiment 3, only two tasks, that is 'ATTACK' and 'VERIFICATION', are considered. So the dimension of each particle vector of PSO is $2(N+M)$. 100 particles are adopted in the PSO. Bothe acceleration factors of PSO are set to 2 and the inertia weight is 0.7. All experiments in Table 11 are run 100,000 times. The format of 'Scenario' in Table 11, is [Target, Task, UAV] whose elements stands for the number of variables.

Table 11 PSO experiment on two tasks scenarios

Scenario	Fitness	Min-Time	Avg-Time	Min-Iteration	Max-Iteration	Avg-Iteration
[1,2,3]	7.0711	0ms	0.675 ms	1	1	1
[2,2,3]	10.831	0ms	1.563 ms	1	1	1
[3,2,4]	14.3162	0ms	1.845 ms	1	18	1.536
[4,2,5]	13.831	0ms	14.828 ms	1	252	13.785
[5,2,6]	16.099	0ms	122.106 ms	1	98.0681	2622

4.5.2. Experiment 2

Some more complicated scenarios are studied in this part. Vehicles should complete three tasks at all targets. The dimension of each particle vector of PSO is $3(N+M)$. The population size and parameters setting are the same with those of the experiment 3.

Table 12 PSO experiment on three tasks scenarios

Scenario	Fitness	Min-Time	Avg-Time	Min-Iteration	Max-Iteration	Avg-Iteration
[1,3,3]	7.0711	0 ms	1.384 ms	1	1	1
[2,3,3]	10.831	0 ms	2.467 ms	1	1	1
[3,3,4]	14.3162	0 ms	20.989 ms	1	192	9.882
[4,3,5]	13.831	15.625 ms	5.769 sec	2	1301	223.764
[5,3,6]*	16.099	156.246 ms	17.934 sec	20	58314	2545.286

* In the scenario, 200 particles of PSO are adopted and algorithms parameters are adjusted ($w=0.7, c_1=c_2=2.1,$) so as to deal with the complexity of the problem and lots of local optimums in the solution space.

4.5.3. Comparison and discussion

(1) Two tasks comparison between PSO and GLPK

The UAV assignment problem is also modeled as a Mixed-Integer Linear Programming problem and the GLPK (GNU Linear Programming Kit)[155] is applied on all scenarios. The average computation time cost by the two methods is compared in

Table 13 and Figure 19 . Computation time rapidly increases with the increase of the number of target and vehicles. Moreover, shown in Figure 19 , search process of GLPK will lead to unacceptable computation time.

Table 13 Two tasks comparison between PSO and GLPK

	Scenarios	Avg-Time of PSO	Avg-Time of GLPK

Scenario-1	[1,2,3]	0.675 ms	47 ms
Scenario-2	[2,2,3]	1.563 ms	157 ms
Scenario-3	[3,2,4]	1.845 ms	2.86 sec
Scenario-4	[4,2,5]	14.828 ms	194.094 sec
Scenario-5	[5,2,6]	122.106 ms	More than 6 hours

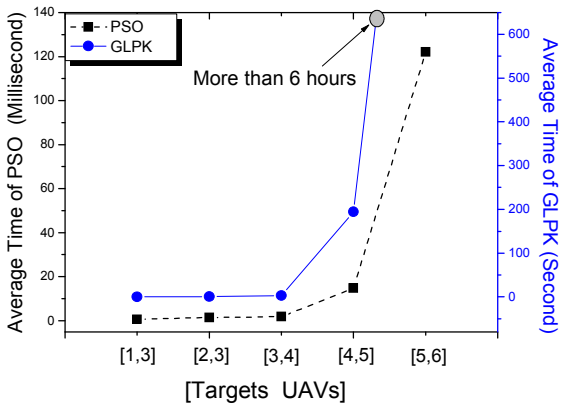

Figure 19 Comparison of two tasks between PSO and GLPK

(2) Three tasks comparison between PSO and GLPK

When a new task is added, the computation cost of both PSO and GLPK increase obviously, compared to the two task scenarios. However, GLPK can not achieve the optimization within an acceptable time.

Table 14 Three tasks comparison between PSO and GLPK

	Scenarios	Avg-Time of PSO	Avg-Time of GLPK
Scenario-5	[1,3,3]	1.384 ms	62 ms
Scenario-6	[2,3,3]	2.467 ms	425 ms
Scenario-7	[3,3,4]	20.989 ms	467.282 sec
Scenario-8	[4,3,5]	5.769 sec	More than 6 hours
Scenario-9	[5,3,6]	17.934 sec	More than one day

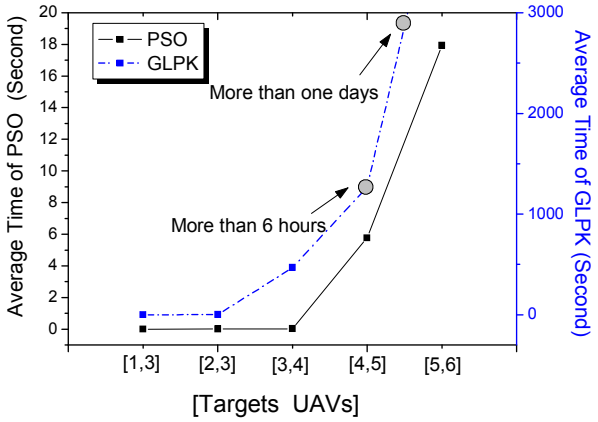

Figure 20 Comparison of three tasks between PSO and GLPK

(3) Impact of the number of tasks on computation time cost

With the increase of number of tasks, it not only leads to a large quantity of variables, but also brings more constraints. So the consideration of computation complexity and time cost becomes more important. In Figure 21 , the computation time cost of PSO is compared between two tasks and three tasks scenarios. The unit of the left vertical coordinates, corresponding to the two tasks scenarios, is millisecond and the unit of the right vertical coordinates is second. An interesting result in the figure is that the computation time cost of PSO increases almost 1000 times when there is a new task added.

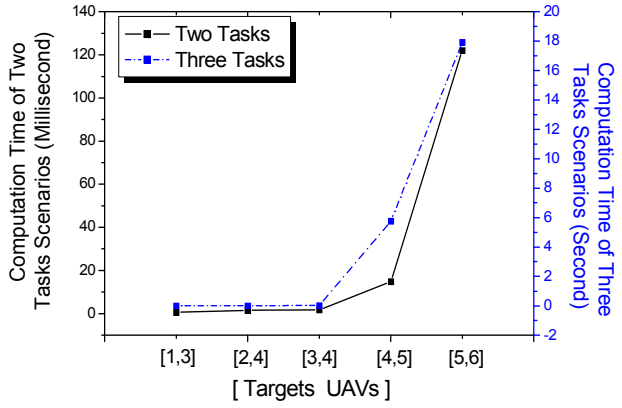

Figure 21 Comparison of PSO computation time between two tasks and three tasks

4.6. Discussion and plan for further research

Flexibility: Current finished work adopted the solution construction method. It will construct feasible solutions at each iteration based on the constraints requirement, which obviously is a very efficient search way in the solution space to solve those assignment problems not too complex. Meanwhile, it lacks flexibility in two aspects. The first one is a new construction techniques will be designed when a new constraint is introduced, and another one is it will difficult to construct feasible solutions, when constraints are complex and have a large mount. So, the plan for next step research is to utilize the techniques of constraint and multi-objective optimization. Solutions will be examined whether it satisfies constraints and it will allow those feasible and unfeasible solutions exits at the same time. Fitness function and selection methods will base on the Pareto rules.

Reliability and Efficiency: Reliability is also the most important point should be considered. However it is a dilemma subject to the efficiency sometimes. So, besides the design of the problems, the improvement and configuration of PSO algorithms are also necessary. The statistic performance will be applied to examine the reliability.

Rescheduling: In a more real-operational environment, online rescheduling will be realized in the feature work. Several non-stationary situations will be occurred, such as the dimension of solution space, quality of constraints or objective changing. The dimension of solution space will changed, when a target or vehicle is added or moved. The quality of constraints is often caused by the change of tasks quality, or some new constraints are introduced, which maybe leads to some feasible solutions become infeasible. If the fitness metrics change, the assignment problem

will be transferred to a dynamic multi-objective optimization problem. So various procedures and heuristics operations, widely adopted in the constraint, dynamic and multi-objective optimization, should be utilized or developed to assign vehicles to targets and schedule those vehicles in the non-stationary environment.

Algorithm Analysis and Improvement: With the increase of the problem size and the number of constraints, the performance of the algorithm (PSO) will not be as good as before. In Figure 22 and Figure 23 , the scenario, which includes 8 targets, 10 UAVs and 2 tasks, is studied. The experiments are run 5 times separately and the fitness curve is illustrated in Figure 22 . The optimum is 10.3031, the best fitness we found so far. Only one experiment, corresponding to the red solid line, achieves the optimum and others stagnate into some local optimum. In Figure 23 , those local optimums of all 5 experiments are summarized and those iterations that the algorithm stagnates into those local optimums are counted. Apparently, the most terrible local optimums of the scenario include 12.6620, 13.0623, 13.6620, 14.2334, 14.4424, 14.4403, etc and they will take algorithm a long time to jump out, or even leads to premature convergence. So, it is necessary to make an improvement to the algorithm to handle those more difficult and complex problems.

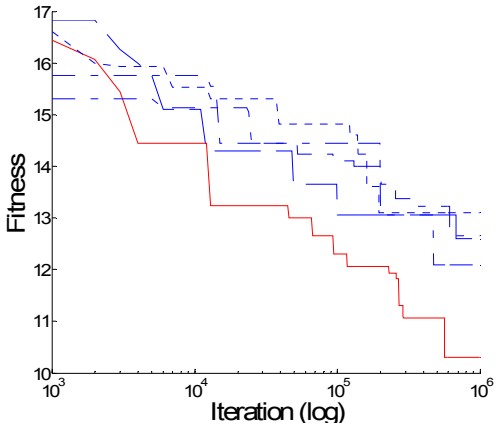

Figure 22 The fitness curve of 8-targets 10-UAVs and 2-tasks

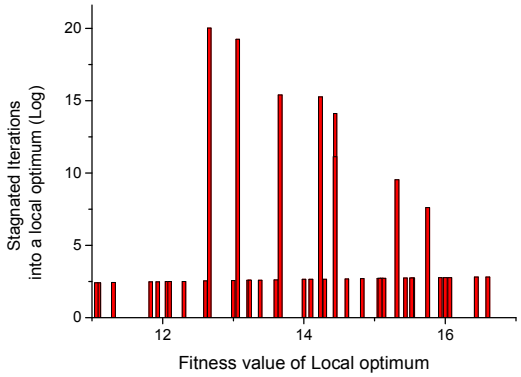

Figure 23 Stagnated iteration into local optimums

5. References

[1] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm intelligence* San Francisco: Morgan Kaufmann Publishers, 2001.

[2] Eberhart, R. C. and Kennedy, J., "A new optimizer using particle swarm theory", Nagoya, Japan. pp. 39-43, 1995

[3] Kennedy, J. and Eberhart, R. C., "Particle swarm optimization", Piscataway, NJ. pp. 1942-1948, 1995

[4] Fogel, L. J., Owens, A. J., and Walsh, M. J., *Artificial intelligence through simulated evolution* 1966.

[5] Carlisle, A. and Dozier, G., "Adapting particle swarm optimization to dynamic environments", Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA. pp. 429-434, 2000

[6] Eberhart, R. C. and Shi, Y., "Tracking and optimizing dynamic systems with particle swarms", Proceedings of IEEE Congress on Evolutionary Computation, pp. 94-100, 2001

[7] Carlisle, Anthony, "Applying the particle swarm optimizer to non-stationary environments." Auburn University, Auburn Alabama, 2002.

[8] Carlisle, A. and Dozier, G., "Tracking changing extrema with adaptive particle swarm optimizer", Proceedings of World Automation Congress, pp. 265-270, 2002

[9] Veeramachaneni, K. and Osadciw, L. A., "Dynamic particle swarm optimizer for information fusion in non stationary sensor networks", Proceedings of IEEE Swarm Intelligence Symposium, Indianapolis, Indiana. 2006

[10] Blackwell, T. M. and Bentley, P. J., "Dynamic search with charged swarms", Proceedings of Genetic and Evolutionary Computation Conference (GECCO), New York, NY. USA. pp. 19-26, 2002

[11] Blackwell, T. M. and Bentley, P. J., "Don't push me! Collision-avoiding swarms", Proceedings of IEEE Congress on Evolutionary Computation, pp. 1691-1696, 2002

[12] Jatmiko, W., Sekiyama, K., and Fukuda, T., "A PSO-based mobile sensor network for odor source localization in dynamic environment: theory, simulation and measurement", Proceedings of IEEE Congress on Evolutionary Computation, 2006

[13] Branke, J., Kaußler, T., Schmidt, C., and Schmeck, H., "A multi-population approach to dynamic optimization problems", Adaptive Computing in Design and Manufacture, pp. 299-308, 2000

[14] Blackwell, T. M. and Branke, J., "Multi-swarm optimization in dynamic environments", Lecture Notes in Computer Science, Coimbra, portugal. pp. 489-500, 2004

[15] Blackwell, T. M. and Branke, J., "multiswarms, exclusion, and anti-convergence in dynamic environments", *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459-472, 2006.

[16] Parrott, D. and Li, X., "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation", Proceedings of IEEE Congress on Evolutionary Computation, pp. 98-103, 2004

[17] Parrott, D. and Li, X., "Locating and tracking multiple dynamic optima by a particle swarm model using speciation", *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440-458, 2006.

[18] Li, X., Branke, J., and Blackwell, T. M., "Particle swarm with speciation and adaptation in a dynamic environment", Proceedings of the annual conference on genetic and evolutionary computation, Seattle, Washington, USA. pp. 51-58, 2006

[19] Lung, R. I. and Dumitrescu, D., "A new collaborative evolutionary-swarm optimization technique", Proceedings of the annual conference on genetic and evolutionary computation, London, United Kingdom. pp. 2817-2820, 2007

[20] Mullen, P. B., Monson, C. K., Seppi, K. D., and Warnick, S. C., "Particle swarm optimization in dynamic pricing", Proceedings of IEEE Congress on Evolutionary Computation, pp. 1232-1239, 2006

[21] Cui, X., Hardin, C. T., Ragade, R. K., Potok, T. E., and Elmaghraby, A. S., "Tracking non-stationary optimal solution by particle swarm optimizer", Proceedings of International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 133-138, 2005

[22] Hu, X. and Eberhart, R. C., "Adaptive particle swarm optimization: detection and response to dynamic systems", Proceedings of IEEE Congress on Evolutionary Computation, pp. 1666-1670, 2002

[23] Bird, S. and Li, X., "Informative performance metrics for dynamic optimisation problems", Proceedings of the annual conference on genetic and evolutionary computation, London, England. pp. 18-25, 2007

[24] Zaharie, D. and Zamfirache, F., "Diversity enhancing mechanisms for evolutionary optimization in static and dynamic environments", Proceedings of of 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, pp. 460-471, 2006

[25] Li, X. and Khanh, H. D., "Comparing particle swarms for tracking extrema in dynamic environments", Proceedings of IEEE Congress on Evolutionary Computation, pp. 1772-1779, 2003

[26] Morrison, R. W., "Performance measurement in dynamic environments", Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 1210-1221, 2003

[27] Moser, I. Review all currently known publications on approaches which solve the moving peaks problem. 7-18-2007. Swinburne University of Technology.

[28] Moser, I. Review all currently known publications on approaches which solve the moving peaks problem. 7-18-2007. Swinburne University of Technology.

[29] Branke, J., *Evolutionary optimization in dynamic environment* Kluwer Academic Publishers, 2002.

[30] Kadrovach, B. A. and Lamont, G. B., "A particle swarm model for swarm-based networked sensor systems", Proceedings of ACM Symposium on Applied Computing, Madrid, Spain. pp. 918-924, 2002

[31] Branke, J., "Memory enhanced evolutionary algorithms for changing optimization problems", Proceedings of IEEE Congress on Evolutionary Computation, 1999

[32] Morrison, R. W. and De Jong, K. A., "A test problem generator for non-stationary environments", pp. 2053, 1999

[33] Coello, C. A., Veldhuizen, D. A. V., and Lamont, G. B., "Evolutionary Algorithms for Solving Multi-Objective Problems", *Kluwer Academic Publishers*, 2002.

[34] Moore, J. and Chapman, R., "Application of Particle Swarm to Multiobjective Optimization", *Department of Computer Science and Software Engineering, Auburn University*, 1999.

[35] Reyes-Sierra, M. and Coello Coello, C. A., "Multi-objective particle swarm optimizers: A survey of the state-of-the--art", *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287, 2006.

[36] Hu, X. and Eberhart, R. C., "Multiobjective optimization using dynamic neighborhood particle swarm optimization", pp. 1677-1681, 2002

[37] Li, X. and Andries, P. E. Particle swarm optimization: an introduction and its recent developments. Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. 3391-3414. 2007. London, United Kingdom, ACM.

[38] Hu, X., Shi, Y., and Eberhart, R. C., "Recent advances in particle swarm", pp. 90-97, 2004

[39] Moore, J., Chapman, R., and Dozier, G. Multiobjective particle swarm optimization. Proceedings of the 38th annual on Southeast regional conference. 56-57. 2000. Clemson, South Carolina, ACM.

[40] Hu, X., Eberhart, R. C., and Shi, Y., "Particle swarm with extended memory for multiobjective optimization", Indianapolis, Indiana, USA. pp. 193-197, 2003

[41] Ho, S. L., Shiyou, Y., Guangzheng, N., Lo, E. W. C., and Wong, H. C., "A particle swarm optimization-based method for multiobjective design optimizations", *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1756-1759, 2005.

[42] Yapicioglu, H., Dozier, G., and Smith, A. E., "Neural Network Enhancement of Multiobjective Evolutionary Search", pp. 1909-1915, 2006

[43] Cagnina, L., Esquivel, S., and Coello, C. A. C., "A particle swarm optimizer for multi-objective optimization", *Journal of Computer Science & Technology*, vol. 5, no. 4, pp. 204-210, 2005.

[44] Mostaghim, S. and Teich, J., "Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)", pp. 26-33, 2003

[45] Hsieh, S. T., Sun, T. Y., Chiu, S. Y., Liu, C. C., and Lin, C. W. Cluster based solution exploration strategy for multiobjective particle swarm optimization. Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications. 295-300. 2007. Innsbruck, Austria, ACTA Press.

[46] Mostaghim, S. and Teich, J., "Covering Pareto-optimal fronts by subswarms in multi-objective particle swarm optimization", pp. 1404-1411, 2004

[47] Abido, M. A. Two-level of nondominated solutions approach to multiobjective particle swarm optimization. Proceedings of the 9th annual conference on Genetic and evolutionary computation. 726-733. 2007. London, England, ACM.

[48] Zhang, X. h., Meng, H. y., and Jiao, L. c., "Intelligent particle swarm optimization in multiobjective optimization", pp. 714-719, 2005

[49] Praveen, K., Sanjoy, D., and Stephen, M. W., "Multi-objective hybrid PSO using Âµ-fuzzy dominance", 2007.

[50] Li, X., "A non-dominated sorting particle swarm optimizer for multiobjective optimization", Chicago, IL, USA. pp. 37-48, 2003

[51] Carlo, R. R. and Prospero, C. N. An effective use of crowding distance in multiobjective particle swarm optimization. Proceedings of the 2005 conference on Genetic and evolutionary computation. 257-264. 2005. Washington DC, USA, ACM.

[52] Fieldsend, J. and Singh, S., "A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence", *In The 00 U.K.Workshop on Computational Intelligence*, pp. 34-44, 2002.

[53] Coello, C. A. and Lechuga, M. S., "MOPSO: a proposal for multiple objective particle swarm optimization", pp. 1051-1056, 2002

[54] ZHAO, B. and CAO, Y. j., "Multiple objective particle swarm optimization technique for economic load dispatch", *Zhejiang Univ SCI*, vol. 6A, no. 5, pp. 420-427, 2005.

[55] Leong, W. F. and Yen, G. G., "Dynamic Population Size in PSO-based Multiobjective Optimization", *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 1718-1725, 2006.

[56] Bartz-Beielstein, T., Limbourg, P., Mehnen, J., Schmitt, K., Parsopoulos, K. E., and Vrahatis, M. N., "Particle swarm optimizers for Pareto optimization with enhanced archiving techniques", *Evolutionary Computation, 2003.CEC apos;03.The 2003 Congress on*, vol. 3, no. 8-12, pp. 1780-1787, 2003.

[57] Ray, T. and Liew, K. M., "A swarm metaphor for multiobjective design optimization", *Engineering Optimization*, vol. 34, no. 2, pp. 141-153, 2002.

[58] Salazar-Lechuga, M. and Rowe, J. E., "Particle swarm optimization and fitness sharing to solve multi-objective optimization problems", pp. 1204-1211, 2005

[59] Pulido, G. T. and Coello Coello, C. A., "Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer", Seattle, WA, USA. pp. 225-237, 2004

[60] ZHAO, B. and CAO, Y. j., "Multiple objective particle swarm optimization technique for economic load dispatch", *Zhejiang Univ SCI*, vol. 6A, no. 5, pp. 420-427, 2005.

[61] Reyes-Sierra, M. and Coello, C. A. C., "Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and e-Dominance," *Evolutionary Multi-Criterion Optimization* 2005, pp. 505-519.

[62] Mich and le, R. L. Multi-objective PSO for interplanetary trajectory design. Proceedings of the 9th annual conference on Genetic and evolutionary computation. 175. 2007. London, England, ACM.

[63] Liu, D. S., Tan, K. C., Goh, C. K., and Ho, W. K., "On Solving Multiobjective Bin Packing Problems Using Particle Swarm Optimization", pp. 2095-2102, 2006

[64] ZHAO, B. and CAO, Y. j., "Multiple objective particle swarm optimization technique for economic load dispatch", *Zhejiang Univ SCI*, vol. 6A, no. 5, pp. 420-427, 2005.

[65] Konstantinos, E. P. and Michael, N. V., "Particle swarm optimization method in multiobjective problems", pp. 603-607, 2002

[66] Ho, S. L., Yang, S., Ni, G., Lo, E. W. C., and Wong, H. C., "A particle swarm optimization-based method for multiobjective design optimizations", *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1756-1759, 2005.

[67] XIA, W. and WU, Z., "Hybrid particle swarm optimization approach formulti-objective flexible job- shop scheduling problems", *Control and Decision*, vol. 20, no. 2, pp. 137-141, 2005.

[68] ZHANG, X. h. and ZHOU, L. x., "Multi-object Optimization Design of PID Controllers Based on Particle Swarm Algorithms", *JOURNAL OF APPLIED SCIENCES*, vol. 25, no. 4, pp. 392-396, 2007.

[69] Liu, W. and Liang, M., "A Particle Swarm Optimization Approach to A Multi-objective Reconfigurable Machine Tool Design Problem", pp. 2222-2229, 2006

[70] Marandi, A., Afshinmanesh, F., Shahabadi, M., and Bahrami, F., "Boolean Particle Swarm Optimization and Its Application to the Design of a Dual-Band Dual-Polarized Planar Antenna", *IEEE Congress on Evolutionary Computation, 2006.CEC 2006.*, pp. 3212-3218, 2006.

[71] Baumgartner, U., Magele, C., and Renhart, W., "Pareto optimality and particle swarm optimization", *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1172-1175, 2004.

[72] Reyes-Sierra, M. and Coello, C. A. C., "A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization", pp. 65-72, 2005

[73] Reyes-Sierra, M. and Coello Coello, C. A., "A Study of Techniques to Improve the Efficiency of a Multi-Objective Particle Swarm Optimizer," *Evolutionary Computation in Dynamic and Uncertain Environments* 2007, pp. 269-296.

[74] Reyes-Sierra, M. and Coello Coello, C. A., "Fitness Inheritance in Multi-Objective Particle Swarm Optimization", Pasadena, California. pp. 116-123, 2005

[75] Reyes-Sierra, M. and Coello, C. A. C. Dynamic fitness inheritance proportion for multi-objective particle swarm optimization. Proceedings of the 8th annual conference on Genetic and evolutionary computation. 89-90. 2006. Seattle, Washington, USA, ACM.

[76] Li, X., "Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximum Fitness Function", *Lecture Notes in Computer Science*, vol. 3102, no. 117-128, 2004.

[77] Zitzler, E., Deb, K., and Thiele, L., "Comparison of multiobjective evolutionary algorithms: empirical results", *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

[78] Emma, L. B. Optimising the flow of experiments to a robot scientist with multi-objective evolutionary algorithms. Proceedings of the 2007 GECCO conference companion on

Genetic and evolutionary computation. 2429-2436. 2007. London, United Kingdom, ACM.

[79] Coello, C. A., Pulido, G. T., and Salazar-Lechuga, M., "Handling multiobjectives with particle swarm optimization", *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 256-279, 2004.

[80] "http://www.cs.cinvestav.mx/~EVOCINV/software.html".

[81] "http://www.pserc.cornell.edu/matpower/#optionalpackages".

[82] http://www.tik.ee.ethz.ch/~zitzler/testdata.html.

[83] Naitali, A. and Giri, F., "Hammerstein and Wiener nonlinear models identification using a multimodal particle swarm optimizer", pp. 6,

[84] Deb, K., "Multi-objective genetic algorithms: Problem difficulties and construction of test problems", *Evolutionary Computation*, vol. 7, no. 3, pp. 205-230, 1999.

[85] Halter, W. and Mostaghim, S., "Bilevel Optimization of Multi-Component Chemical Systems Using Particle Swarm Optimization", pp. 1240-1247, 2006

[86] Eric, O. and Babak, F. A Particle Swarm Algorithm for Multiobjective Design Optimization. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence. 765-772. 2006. IEEE Computer Society.

[87] Stefan, J., Daniel, M., and Martin, M., "Molecular docking with multi-objective Particle Swarm Optimization", *Appl.Soft Computing*, vol. 8, no. 1, pp. 666-675, 2008.

[88] Michalewicz, Z., "A survey of constraint handling techniques in evolutionary computation methods", Proceedings of Annual Conference on Evolutionary Programming, Cambridge, MA. 1995

[89] Coello, C. A. C. A Survey of Constraint Handling Techniques used with Evolutionary Algorithms. 1999. Xalapa, Veracruz, MÃ©xico, Laboratorio Nacional de InformÃ¡tica Avanzada. Technical Report Lania-RI-99-04.

[90] Coello, C. A. C., "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art", *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11, pp. 1245-1287, 2002.

[91] Mezura-Montes, E. and Coello Coello, C. A., "A Survey of Constraint-Handling Techniques Based on Evolutionary Multiobjective Optimization", PPSN workshop on Multiobjective Problem Solving from Nature, Reykjavik, Iceland. 2006

[92] Coello, C., "Constraint-handling techniques used with evolutionary algorithms", 2007.

[93] Apt, K. R. and Wallace, M. G., *Constraint Logic Programming using ECLiPSe* Cambridge University Press, 2006.

[94] Liang, J. J. and Suganthan, P. N., "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism", pp. 9-16, 2006

[95] Zhu, Q., Qian, L., Li, Y., and Zhu, S., "An Improved Particle Swarm Optimization Algorithm for Vehicle Routing Problem with Time Windows", pp. 1386-1390, 2006

[96] Parsopoulos, K. E. and Vrahatis, M. N., "Particle swarm optimization method for constrained optimization problems", 2002

[97] Takahama, T. and Sakai, S., "Solving Constrained Optimization Problems by the Îµ Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control", pp. 1-7, 2006

[98] Xu, F., Chen, W., and Yang, L., "Improved Particle Swarm Optimization for Realistic Portfolio Selection", 2007.

[99] Runarsson, T. P. and Yao, X., "Stochastic Ranking for Constrained Evolutionary Optimization", *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284-294, 2000.

[100] He, Q. and Wang, L., "An effective co-evolutionary particle swarm optimization for constrained engineering design problems", *Eng.Appl.Artif.Intell.*, vol. 20, no. 1, pp. 89-99, 2007.

[101] Ismael, V., Lu, and Vicente, N., "A particle swarm pattern search method for bound constrained global optimization", *J.of Global Optimization*, vol. 39, no. 2, pp. 197-219, 2007.

[102] Amin, F. F., Mehdi, K., Sied Mehdi, F., and Saeed, S., "HW/SW partitioning using discrete particle swarm", 2007.

[103] Hu, X. and Eberhart, R. C., "Solving constrained nonlinear optimization problems with particle swarm optimization", Orlando, USA. 2002

[104] Hu, X., Eberhart, R. C., and Shi, Y., "Engineering optimization with particle swarm", Proceedings of IEEE Swarm Intelligence Symposium, pp. 53-57, 2003

[105] Takahama, T. and Sakai, S., "Solving Constrained Optimization Problems by the Îµ Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control", pp. 1-7, 2006

[106] Talal, M. A. and Mohamed, A. A., "Simulation-based optimization for repairable systems using particle swarm algorithm", 2005.

[107] Goldbarg, E. F. G., de Souza, G. R., and Goldbarg, M. C., "Particle Swarm Optimization for the Bi-objective Degree constrained Minimum Spanning Tree", pp. 420-427, 2006

[108] Coath, G. and Halgamuge, S. K., "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems", pp. 2419-2425, 2003

[109] Pulido, G. T. and Coello, C. A. C., "A constraint-handling mechanism for particle swarm optimization", pp. 1396-1403, 2004

[110] Zhang, W. J. and Xie, X. F., "DEPSO: hybrid particle swarm with differential evolution operator", pp. 3816-3821, 2003

[111] Angel, E. M., oz, Z., Arturo, H., ndez, A., and Enrique, R. V. D., "Constrained optimization via particle evolutionary swarm optimization algorithm (PESO)", 2005.

[112] Zielinski, K. and Laur, R., "Constrained Single-Objective Optimization Using Particle Swarm Optimization", pp. 443-450, 2006

[113] Takahama, T. and Sakai, S., "Constrained optimization by the á constrained particle swarm optimizer", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 9, no. 3, pp. 282-289, 2005.

[114] Takahama, T. and Sakai, S., "Constrained optimization by å constrained particle swarm optimizer with å-level control", Proccedings of IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST), Muroran, Japan. 2005

[115] Takahama, T. and Sakai, S., "Solving Constrained Optimization Problems by the Îμ Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control", pp. 1-7, 2006

[116] Pan, F., Chen, J., Gan, M., Cai, T., and Tu, X., "Model analysis of particle swarm optimizer", *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 32, no. 3, pp. 368-377, 2006.

[117] Ray, T. and Liew, K. M., "A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem", Seoul, Korea. pp. 75-80, 2001

[118] Wang, L. and Singh, C., "Adequacy-based Design of A Hybrid Generating System Including Intermittent Sources Using Constrained Particle Swarm Optimization", pp. 1-7, 2007

[119] Ji, C., "A Revised Particle Swarm Optimization Approach for Multi-objective and Multi-constraint Optimization", 2004.

[120] Takahama, T. and Sakai, S., "Solving Constrained Optimization Problems by the Îµ Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control", pp. 1-7, 2006

[121] Paquet, U. and Engelbrecht, A. P., "A new particle swarm optimiser for linearly constrained optimisation", pp. 227-233, 2003

[122] Ulrich, P. and Andries, P. E., "Particle Swarms for Linearly Constrained Optimisation", *Fundam.Inf.*, vol. 76, no. 1-2, pp. 147-170, 2007.

[123] Halter, W. and Mostaghim, S., "Bilevel Optimization of Multi-Component Chemical Systems Using Particle Swarm Optimization", pp. 1240-1247, 2006

[124] Schoofs, L. and Naudts, B., "Swarm intelligence on the binary constraint satisfaction problem", 2002.

[125] Lin, I. Ling, "Particle swarm optimization for solving constraint satisfaction problems." MASTER OF SCIENC Simon Fraser University, School of Interactive Arts and Technology, 2005.

[126] Yang, Q., Sun, J., and Zhang, J., "Improvements of Particle Swarm in Binary CSPs with Maximal Degree Varibles Ordering", *Journal of Computer Research and Development*, vol. 43, no. 3, pp. 436-441, 2006.

[127] Alberto, M. and Julian, T., "Geometric particle swarm optimization for the sudoku puzzle", 2007.

[128] Eberhart, R. C. and Kennedy, J., "A new optimizer using particle swarm theory", Proceedings of International Symposium on Micro Machine and Human Science, pp. 39-43, 1995

[129] Kennedy, J. and Eberhart, R. C., "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995

[130] Eberhart, R. C., Simpson, P. K., and Dobbins, R. W., *Computational Intelligence PC tools*, 1st ed. ed. Boston, MA: Academic press professional, 1996.

[131] Eberhart, R. C. and Shi, Y., *Computational Intelligence: Concepts to Implementations* San Francisco: Morgan Kaufmann Publishers, 2007.

[132] Shi, Y. and Eberhart, R. C., "A modified particle swarm optimizer", Proceedings of IEEE Congress on Evolutionary Computation, pp. 69-73, 1998

[133] Shi, Y. and Eberhart, R. C., "Parameter selection in particle swarm optimization", Proceedings of Annual Conference on Evolutionary Programming, New York. pp. 591-600, 1998

[134] Shi, Y., Eberhart, R., and Chen, Y., "Implementation of evolutionary fuzzy system", *IEEE Transactions on Fuzzy Systems*, 1999.

[135] Matsui, K., "New selection method to improve the population diversity in genetic algorithms", Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 625-630, 1999

[136] Wang, K., "A new fuzzy genetic algorithm based on population diversity", Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 108-112, 2001

[137] Wen, J., Wang, S., Cheng, S., Wu, Q., and Shimmin, D., "Measurement based power system load modeling using a population diversity genetic algorithm", Proceedings of International Conference on Power System Technology, pp. 771-775, 1998

[138] Wen, J. Y., Wen, J. Y., Wu, Q. H., Shimmin, D. W., Turner, D. R., and Cheng, S. J., "Population diversity based genetic algorithm for fuzzy control of synchronous generators", Proceedings of IEEE International Symposium on Computer Aided Control System Design, pp. 504-509, 1999

[139] Chandler, P. R., Pachter, M., Swaroop, D., and Fowler, J., "Complexity in UAV cooperative control", Proceedings of the American Control Conference, pp. 1831-1836, 2002

[140] Alighanbari, M., Alighanbari, M., Kuwata, Y., and How, J. P., "Coordination and control of multiple UAVs with timing constraints and loitering", pp. 5311-5316, 2003

[141] Tao, Long, "Research on Distributed Task Allocation and Coordination for Multiple UCAVs Cooperative Mission Control." PHD PHD thesis, National University of Defense Technology,, Control Science and Engineering, 2006.

[142] Secrest, Barry R., "Traveling salesman problem for surveillance mission using particle swarm optimization." Master's thesis Air University, School of Engineering and Management of the Air Force Institue of Technology, 2001.

[143] Vijay, K. S., Moises, S., and Rakesh, N., "Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles," Elsevier Science Ltd., 2008, pp. 1813-1828.

[144] Schumacher, C., Chandler, P. R., Pachter, M., and Pachter, L. S. Optimization of Air Vehicle Operations Using Mixed-Integer Linear Programming. 2006. Air Force Research Lab (AFRL/VACA) Wright-Patterson AFB OH Control Theory Optimization Branch.

[145] Schumacher, C., Chandler, P. R., Pachter, M., and Pachter, L. S., "UAV task assignment with timing constraints via mixed-integer linear programming", AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit, 2004

[146] Nygard, K. E., Chandler, P. R., and Pachter, M., "Dynamic network flow optimization models for air vehicle resource allocation", Proceedings of the American Control Conference, Arlington, Texas. pp. 1853-1858, 2001

[147] Kuhn, H. W., "Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistic Quaterly 1955;2:83-97", *Naval Research Logistic Quaterly*, vol. 2 pp. 83-97, 1955.

[148] Darryl, K. A., Arnold, H. B., and John, R., "Assignment scheduling capability for unmanned aerial vehicles: a discrete event simulation with optimization in the loop approach to solving a scheduling problem", 2006.

[149] Rasmussen, S. J. and Shima, T., "Tree search algorithm for assigning cooperating UAVs to multiple tasks", *International Journal of Robust and Nonlinear Control*, vol. 18, no. 2, pp. 135, 2007.

[150] Arulselvan, A., Commander, C. W., and Pardalos, P. M. A hybrid genetic algorithm for the target visitation problem. 2008. Naval Research Logistics.

[151] Chen, G., Jose, J., and Cruz, B., "Genetic algorithm for task allocation in UAV cooperative control", 2003.

[152] Tal, S., Steven, J. R., Andrew, G. S., and Kevin, M. P., "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms," Elsevier Science Ltd., 2006, pp. 3252-3269.

[153] Duan, H. Methods of multi-UAVs' mission assignments based on basic ant colony intelligence. 2007. P.R.China.

[154] O'Rourke, K. P., Bailey, T. G., Hill, R., and Carlton, W. B., "Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search", *Military Operations Research Journal*, vol. 6 2000.

[155] GNU Linear Programming Kit package. http://gnuwin32.sourceforge.net/packages/glpk.htm . 2008.