2011 International Conference on Management and Artificial Intelligence
IPEDR vol.6 (2011) © (2011) IACSIT Press, Bali, Indonesia

Particle Swarm Optimization for Constrained and Multiobjective Problems: A Brief Review

Nor Azlina Ab. Aziz
Faculty of Engineering and Technology
Multimedia University
Malaysia
azlina.aziz@mmu.edu.my

Mohamad Yusoff Alias
Faculty of Engineering
Multimedia University
Malaysia
yusoff@mmu.edu.my

Ammar W. Mohemmed
Knowledge Engineering & Discovery Research Institute
Auckland University of Technology
New Zealand
ammar.mohemmed@aut.ac.nz

Kamarulzaman Ab. Aziz
Faculty of Management
Multimedia University
Malaysia
kamarulzaman.aziz@mmu.edu.my

***Abstract*— Particle swarm optimization (PSO) is an optimization method that belongs to the swarm intelligence family. It was initially introduced as continuous problem optimization tool. It has evolved to being applied to more complex multiobjective and constrained problem. This paper presents a systematic literature review of PSO for constrained and multiobjective optimization problems.**

Keywords- Constrained optimization problems; multiobjective optimization problems; particle swarm optimization

I. INTRODUCTION

One of successful optimization algorithms is particle swarm optimization (PSO). PSO main attractive feature is its simple and straightforward implementation. PSO has been applied in multiple fields such as human tremor analysis for biomedical engineering, electric power and voltage management and machine scheduling [1].

The original PSO is proposed for optimization of single objective continuous problem. However the concept of PSO has been expanded to allow it to handle other optimization problems such as; binary, discrete, combinatorial, constrained and multiobjective optimization. This paper will review some of the works conducted in constrained and multiobjective PSO in section 4 and 5. However before that the concept of PSO is discussed in the next section followed by the concept of multiobjective and constrained problems. The paper is then concluded in section 6.

II. PARTICLE SWARM OPTIMIZATION

In 1995, James Kennedy and Russell Eberhart introduced particle swarm optimization (PSO). It is a swarm based algorithm that mimics the social behaviour of organisms like birds and fishes. The success of an individual in these communities is affected not only by its own effort but also by the information shared by its surrounding neighbours. This nature of the social behaviour is imitated by PSO using swarm of agents called particles [2]. The interaction of the particles with their neighbours is the key to the effectiveness of PSO. Particles neighbourhood in PSO had been studied from two perspectives; global neighbourhood (*gBest*) and local neighbourhood (*lBest*). In *gBest* the particles are fully connected therefore the particles search is directed by the best particle of the swarm. While in *lBest* the particles are connected to their neighbours only and their search is conducted by referring to the neighbourhood best. A particle in PSO has a position ($\mathbf{X_i}$) and velocity ($\mathbf{V_i}$). The position represents a solution suggested by the particle while velocity is the rate of changes of the next position with respect to current position. Initially these two values (position and velocity) are randomly initialised. In the subsequent iterations the search process is conducted by updating these values using the following equations:

$$\mathbf{V}_i = \boldsymbol{w} \times \mathbf{V}_i + c_1 \times rand_1() \times (\mathbf{P}_i - \mathbf{X}_i) + c_2 \times rand_2() \times (\mathbf{P}_g - \mathbf{X}_i) \quad (1)$$

$$\mathbf{X}_i = \mathbf{X}_i + \mathbf{V}_i \quad (2)$$

where i is particle's number (i = 1,..,N; N: number of particles in the swarm).

As what can be observed in Eq.1 velocity is influenced by; $\mathbf{P_i}$ – the best position found so far by the particle and $\mathbf{P_g}$ – the best position found by the neighbouring particles. $\mathbf{P_g}$ could be either the position of *gBest* or *lBest*. The $\mathbf{V_i}$ value is clamped to $\pm V_{max}$ to prevent explosion. If the value of V_{max} is too large the exploration range is too wide however if it is too small particles will favour local search [3]. c_1 and c_2 are the learning factors to control the effect of the "best" factors of particles; $\mathbf{P_i}$ and $\mathbf{P_g}$, typically both are set to 2 [1]. $rand_1()$ and $rand_2()$ are two independent random numbers in the range of [0.0,1.0]. The randomness provides energy to the particles. w is known as inertia weight, a term added to improve PSO's performance. The inertia weight controls particles momentum so that, they can avoid continuing to explore the wide search space and switch to fine tuning when a good area is found [4].

The particle position is updated using Eq.2 where the velocity is added to the previous position. This shows how a

particle next search is launched from its previous position and the new search is influenced by its pass search [5].

The quality of the solution is evaluated by a fitness function, which is a problem-dependent function. If the current solution is better than the fitness of $\mathbf{P_i}$ or $\mathbf{P_g}$ or both, the best value will be replaced by current solution accordingly. This update process continues until stopping criterion is met, usually when either maximum iteration is achieved or target solution is attained. When the stopping criterion is satisfied, the best particle found so far is taken as the optimal solution (near optimal). The PSO algorithm is presented in Fig. 1.

```
Initialize particles population;
Do{
  Calculate fitness values of each particles using fitness function;
  Update p_id if the current fitness value is better than p_id;
  Determine p_gd : choose the particle position with the best fitness
  value of all the neighbors as the p_gd;
  For each particle {
      Calculate particle velocity according to (1);
      Update particle position according to (2);
  }
} While maximum iteration or ideal fitness is not attained;
```

Fig. 1: PSO Algorithm

III. Multiobjective and Constrained Optimization Problems

Multiobjective optimization is a problem with many objectives to be fulfilled and most of the time these objectives are in conflict with each other. Whereas constrained optimization is an optimization problem with one or more constraints to be obeyed.

These types of problems are commonly faced in everyday life, for example in this situation:

Mr. ABC wants to buy a new computer. He wants a computer with the best specifications but with the least cost.

The problem faced by Mr. ABC is a multiobjective problem with two objectives; the first objective is to get a computer with the best specifications while the second objective is to spend the least amount of money. Following is a similar problem in constrained optimization problem form:

Ms. XYZ also wants to buy a computer, but she has a limited budget. She is aiming to get the best computer possible within her budget.

Ms. XYZ is facing a constrained optimization problem where she needs to buy a good computer but subject to a limited budget.

Mathematically these optimization problems can be presented as follows:

$$\begin{aligned} &\textit{optimize} && :f_i(\mathbf{X}) && i = 1,...,l \\ &\textit{subject to} && :c_j(\mathbf{X}) \le 0 && j = 1,..., p \\ & && h_k(\mathbf{X}) = 0 && k = p+1,...,q \\ &\mathbf{X} = (x_1, x_2, ..., x_n)^T && && \mathbf{X} \in [x_{\min}, x_{\max}] \end{aligned} \quad (3)$$

$f_i(\mathbf{X})$ are the l objectives to be optimized while $c_j(\mathbf{X})$ and $h_k(\mathbf{X})$ are inequality and equality constraints enforced on the problem, respectively. $\mathbf{X}$ is the search variables vector that is bounded in the search space within the $x_{\min}$ and $x_{\max}$ values. A constrained optimization could have only one objective, meanwhile multiobjective optimization could be limited by its search space only.

IV. PSO For Constrained Optimization Problems

In their work, [6] categorised the evolutionary algorithm optimization methods for constrained problem into four types:

A) Preserve feasibility of solutions

According to this approach only feasible solutions are generated to maintain feasibility. The search process is also restricted within only the feasible area.

B) Penalty functions

In the penalty functions method, the constrained optimization problem is solved using unconstrained optimization method by incorporating the constraints into the objective function thus transforming it into an unconstrained problem.

C) Differentiate the feasible and infeasible solutions

Among common methods belonging to this group are repairing infeasible solutions so that a feasible solution is produced and preferring feasible solutions over infeasible solutions.

D) Hybrid methods

A hybrid method is a combination of at least two of the three methods described above.

Among the early research works on PSO for constrained optimization problem is by [7] where the simple idea of feasible solution preservation is used. There are two points that differentiate this work from the basic PSO. The first one is that the particles are initialised with only feasible solutions. This step speeds up the search process. The second point is the update of the best solutions (*gBest* and *pBest*). Only feasible solutions will be selected as the best values. This approach is simple and able to handle a wide variety of constrained problems. However, for problems with a small feasible region the initialisation process tends to consume a lot of time thus delaying the search [8].

Penalty function technique is usually used in evolutionary algorithm for constrained optimization problems [9]. The penalty function combines the constraints with the objective function by adding penalty value to infeasible solutions (in minimization problems). In [10] the penalty functions methods are grouped into three classes:

(a) Static penalty functions

This is the simplest approach for penalty functions. It gives fixed penalty value once a constraint is violated no matter how large or small the violations are. A better method is to penalise the constraints violation based on how far the solution is from the feasible region (distance based penalty). This can be done by multiplying the fixed penalty

coefficients with their constraint functions or by dividing the violation into several levels with each level having its own penalty coefficient.

(b) Dynamic penalty functions

The main problem of the static penalty method is the difficulty in determining suitable penalty coefficients values. Another issue is that although feasibility is a must but exploration in infeasible region is not always bad, especially during the initial stage. The static penalty is too strict in ensuring solution is feasible. Hence, if the penalty coefficients are large, exploration in infeasible regions will be severely penalised throughout the search process. To overcome this, a dynamically changing distance based penalty where the penalty value increases with time is usually used. The severity of the penalty needs to be chosen properly, as too lenient will result in an infeasible solution, but when it is too strict the search ability will be lowered thus leading to a non-optimal feasible solution.

(c) Adaptive penalty functions

Adaptive penalty function incorporates the quality of solutions (i.e., whether the previous best solutions are feasible or not), in determining the penalty. Exploration in feasible region is encouraged and a search in infeasible region is discouraged but not as rigid as static approach.

Since the penalty method transforms a constrained optimization problem into an unconstrained problem, constrained problems that adopt penalty function methods can be optimized using basic PSO, without the need to change the structure of the PSO algorithm. [11] used the penalty functions in their work, where a penalty is given based on the search stage and the distance of the solution from the feasible region. [8], compared this penalty function based PSO [11] with feasibility preservation based PSO [7], and they concluded that both methods are highly competitive. However the penalty function PSO converges faster and does not need to deal with the initialisation problem faced by the feasibility preservation PSO. Penalty function method performance degrades when dealing with highly constrained problems [12].

A distance based fuzzy penalty function method where the penalty coefficients are given based on which infeasible region the solution in, is proposed by [13]. The method is to be used with evolutionary algorithms. The results of ten cases studied show that the fuzzy penalty is a good method in determining the penalty coefficients.

A constrained PSO algorithm with turbulence operator is proposed in [12]. The turbulence operator is incorporated into the particle velocity to encourage exploration in the initial stage of the search process. The constraints are handled during the leader selection phase. When comparing between two particles, if both particles are feasible, the particle with the best fitness wins. However, if one is feasible but the other is infeasible, the feasible particle wins regardless of the fitness values. Finally, if both are infeasible, the particle with the lowest constraints violation wins.

Another interesting constraints handling approach using PSO is introduced in [14]. Their work proposes to convert the constrained optimization problem into min-max problem by transforming the primal problem into its dual problem, e.g., for a minimization primal problem, the dual problem is maximization. Lagrange multipliers are introduced in the dual problem. In order to solve the min-max problem, two coevolving swarms of particles are used. The first swarm optimizes the solution vector of the minimization problem while the Lagrange multipliers are frozen. In the second swarm the solution vector values are fixed, the particles look for optimal value of the Lagrange multipliers for the maximization problem. The frozen values of Lagrange multipliers in the first swarm is determined by the search conducted in the second swarm, while the value of the solution vector in the second swarm is chosen based on the best value found by the first swarm.

A dynamic multi-swarm PSO tailored for constrained problems is suggested in [15]. The swarm is divided into subswarms periodically with the particles assigned in each subswarms are randomly chosen. Each of the subswarms is searching for a better solution considering the set of constraints assigned to it.

In [16], a relax dominance concept which is borrowed from the multiobjective optimization field is used. The constrained optimization problem is transformed into a multiobjective problem by viewing the constraints as an additional objectives to be optimized. Therefore, the objectives are now:

1. Optimized objective function
2. Search for feasibility.

The second objective has a higher priority (i.e., feasibility is more important).

V. PSO FOR MULTIOBJECTIVE OPTIMIZATION PROBLEMS

Classically, optimization methods for multiobjective optimization problems search for a single optimal solution [17]. This is done by converting the problem into a single objective problem through weighted aggregation method or by handling one objective at a time. On the other hand, modern approaches try to generate a set of solutions for a multiobjective optimization problem using Pareto based or non-Pareto based approaches. Pareto based approach looks for non-dominated solutions which are solutions that cannot be further improved with respect to an objective without decreasing its performance for other objectives [18]. Since modern approaches generate a set of solutions, at the end of the optimization process a decision maker is needed to decide which of these solutions is the most suitable for the problem.

There are quite a number of works conducted to enable PSO to solve multiobjective problems. [19] in their survey classified the approaches taken into six groups:

A) Aggregating approaches

In these approaches the objectives are transformed into a single objective by summing them together using weighted aggregation method. Each of the objectives is assigned a weight based on their importance level. Hence, prior knowledge on the problem is needed.

B) Lexicographic ordering

Lexicographic ordering ranks the objectives based on their importance level. Lexicographic ordering method starts by optimizing the most important objective [20]. Using the solutions for this objective, the next important objective is optimized. This process is continued until all objectives are processed. Lexicographic ordering solution depends on the objectives rank. This method is only suitable for problems with a few objectives to be optimized.

C) Sub-population approaches

A sub-population approach divides the swarm of particles into subswarms where they work on their own to search for solution/s and exchange the information they have with other subswarms.

D) Pareto-based approaches

Pareto-based approaches find set of solutions known as non-dominated solutions. The Pareto based method is one of the most popular methods studied for multiobjective PSO.

E) Combined approaches

The methods in this category combine multiple approaches as discussed above in a single optimization algorithm.

F) Other approaches

All other approaches that do not fall under the previous categories are grouped in this category.

Multiobjective PSO using weighted aggregation approach is suggested in [21]. Three variations of weighted aggregation approach are used in the paper;

1. linear aggregation function – fixed weights are assigned to the objectives throughout the search process,
2. bang-bang weighted aggregation function – weights keep changing during the search, and
3. dynamic weighted aggregation function – weight changing during the search but the change is smaller than the bang-bang method.

The PSO weighted aggregation method is applied by [22] to solve machine loading problems in flexible manufacturing systems.

Another multiobjective PSO algorithm is proposed in [21]. This second algorithm is known as vector evaluated PSO (VEPSO). In VEPSO, multiple swarms are used where the number of swarms is based on the number of objectives. Each of the swarms has its own objective to optimize and the velocity is updated using information from other swarms. This algorithm produces a set of Pareto front solutions.

Information sharing in traditional PSO is a one-way process, where the information is passed by *gBest* particle in global neighbourhood or *lBest* particles in local neighbourhood, while other particles are not able to share their experience with the swarms. Due to this, the swarm is not able to identify non-dominated/ Pareto front solutions. To overcome this problem, a dynamic neighbourhood PSO (DNPSO) is proposed by [23]. Taking example of a multiobjective problem with two objectives to be optimized, a particle neighbourhood in DNPSO is determined using the distance of the particle fitness of the first objective with the rest of the swarm. Based on the distance, *m* of the nearest particles are selected as neighbours and the best particle in this neighbourhood is chosen using their fitness of the second objective. The Pareto front solutions are identified during particle best selection, where if and only if the new solution dominates current particle best, the particle best is updated. However, this means that the number of Pareto front solutions found by the proposed DNPSO is limited by the number of particles, resulting some potential good solutions to be lost. Hence in [24] a DNPSO with extended memory is introduced, so that the Pareto front candidates are not just the particle bests but also the extended memory.

[25] in their work suggest a global repository system to be incorporated to PSO to store the search experiences of the particles and hold non-dominated solutions. The particles update takes best value from the repository for the social term of the velocity. This algorithm is further extended to handle constraints, and a mutation operator is introduced to avoid PSO from converging towards false Pareto fronts and encourage exploration [26]. A parallel processing; divided range multiobjective PSO (DRMPSO) divides the swarm into subpopulations and uses the multiobjective PSO as proposed by [25] to handle the mutiobjective in each subpopulations [27]. The work also proposed symbiosis mechanism to handle the constraints.

The works based on limited memory/repository to store non-dominated solutions provide better performance than [23]. However, the limitation on the number of non-dominated solutions makes them inefficient [28]. Therefore, [28] suggest an unconstraint archive to store the non-dominated solutions. The archive used the dominated tree data structure. They also used velocity with turbulence operator to enhance the search.

In [29] the aim is to look for non-dominated solutions, it proposes a multiobjective PSO that determines the non-dominated solutions in two levels – local and global set update. Line search was also proposed to be used with multiobjetive PSO algorithm to generate uniform distributed Pareto optimal solutions [30].

VI. CONCLUSION

PSO is a very useful optimization algorithm. The original PSO is proposed for optimization of single objective continuous problem. However the concept of PSO has been expanded to allow it to handle other optimization problems such as; binary, discrete, combinatorial, constrained and multiobjective optimization. This paper reviewed some of the works conducted in constrained and multiobjective optimization problems. This body of work reviewed suggest the significance of PSO as optimization strategy and highlights its evolution from being used for simple single objective continuous problems to more complex multiobjective and constrained problem.

REFERENCES

[1] Kennedy, J., Eberhart, R. and Shi, Y. (2001). *Swarm Intelligence*, Morgan Kaufmann, US.

[2] Kennedy, J., and Eberhart, R. C., (1995). *Particle swarm optimization,* In Proceedings of IEEE International Conference on Neural Networks. pp.:1942-1948.

[3] Shi, Y., and Eberhart, R. C. (1998b). *Parameter selection in particle swarm optimization,* In Proceedings of the 7th International Conference on Evolutionary Programming pp.: 591 – 600

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

[5] Kennedy, J. (2005) *Why Does It Need Velocity?* In: Proceedings of Swarm Intelligence Symposium. pp.: 38 – 44

[6] Michalewicz, Z. and Schoenauer, M. (1996) *Evolutionary Algorithms for Constrained Parameter Optimization Problems*, Evolutionary Computation, 4(1). pp.: 1-32

[7] Hu, X. and Eberhart, R. (2002b) *Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization,* In Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics

[8] Coath, G. and Halgamuge, S.K. (2003) *A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constraint Nonlinear Optimization Problems,* In Congress on Evolutionary Computation, Vol.4. pp.: 2419 - 2425

[9] Michalewicz, Z. (1995) *Genetic Algorithms, Numerical Optimization, and Constraints,* In Proceedings of the 6th International Conference on Genetic Algorithms. pp.: 151-158.

[10] Smith, A.E. and Coit, D.W. (1996). *Constraint Handling Techniques - Penalty Functions,* In: Baeck, T., Fogel, D. and Michalewicz, Z. (Eds.) *Handbook of Evolutionary Computation.* Oxford University Press and Institute of Physics Publishing, Bristol, U.K. Chapter C5.2.

[11] Parsopoulos, K.E., and Vrahatis, M.N., (2002b) *Particle Swarm Optimization Method for Constrained Optimization Problems,* In Proceedings of the Euro-International Symposium on Computational Intelligence

[12] Pulido, G.T., and Coello Coello, C.A., (2004) *A Constraint-Handling Mechanism for Particle Swarm Optimization,* In Proceedings of the Congress on Evolutionary Computation

[13] Wu, B., Yu, X., and Liu, L. (2001) *Fuzzy Penalty Function Approach for Constrained Function Optimization with Evolutionary Algorithms,* In Proceedings of the 8th International Conference on Neural Information Processing, pp.: 299-304

[14] Shi, Y. and Krohling, R.A. (2002) *Co-evolutionary Particle Swarm Optimization to Solve min-max Problems,* In Proceedings of the Congress on Evolutionary Computation Vol. 02. pp.: 1682-1687

[15] Liang, J.J. and Suganthan, P.N. (2006) *Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism,* In IEEE Congress on Evolutionary Computation

[16] Vaz A. I. F., and Fernandes E. M. G. P. (2006) *Optimization of Nonlinear Constrained Particle Swarm* In Technological and economic development of economy Baltic Journal on Sustainability. Vilnius: Technika, 12(1) , pp.: 30-36

[17] Ngatchou, P., Zarei, A., and El-Sharkawi, M.A (2005) *Pareto Multi Objective Optimization*, In Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems. pp.: 84 – 91

[18] Baumgartner, U., Magele, Ch. and Renhart, W. (2004) *Pareto Optimality and Particle Swarm Optimization,* In IEEE Transactions on Magnetics, 40(2) pp.: 1172-1175

[19] Reyes-Sierra, M. and Coello Coello, C.A (2006) *Multi-Objective Particle Swarm Optimizers A Survey of the State-of-the-Art,* In International Journal of Computational Intelligence Research, 2(3) pp.:287-308

[20] Cvetkovic, D., Parmee, I., and Webb, E. (1998) *Multi-objective Optimisation and Preliminary Airframe Design,* In ACDM'98. pp.:255-267

[21] Parsopoulos, K.E. and Vrahatis, M.N (2002a) *Particle Swam Optimization Method in Multiobjective Problems,* In Proceedings of the ACM Symposium on Applied Computing. pp.: 603 - 607

[22] Ponnambalam, S.G. and Low, S.K. (2008) *Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization* In World Academy of Science, Engineering and Technology 39. pp.:14-19

[23] Hu, X. and Eberhart, R. (2002a) *Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization,* In Proceedings of the IEEE Congress on Evolutionary Computation pp.: 1677-1681

[24] Hu, X., Eberhart, R.C. and Shi, Y. (2003a) *Particle Swarm with Extended Memory for Multiobejctive Optimization,* In Proceedings of the IEEE Swarm Intelligence Symposium. pp.: 193-197

[25] Coello Coello, C.A. and Lechuga, M.S. (2002) *MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization,* In Proceedings of the Congress on Evolutionary Computation Vol. 02. pp.: 1051-1056

[26] Coello Coello, C.A., Pulido, G.T. and Lechuga, M.S. (2004) *Handling Multiple Objectives with Particle Swarm Optimization,* In IEEE Transactions on Evolutionary Computation, 8(3). pp.:256-279

[27] Ji, C. (2004) *A Revised Particle Swarm Optimization Approach for Multi-objective and Multi-constraint Optimization*, In Proceedings of the Genetic and Evolutionary Computation Conference

[28] Fieldsend, J.E. and Singh, S. (2002) *A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence* In UK Workshop on Computational Intelligence

[29] Abido, M.A (2007) *Two-Level of Nondominated Solutions Approach to Multiobjective Particle Swarm Optimization,* In Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. pp.: 726 – 733

[30] Grosan, C. and Abraham, A. (2008) *Generating Uniformly Distributed Pareto Optimal Points for Constrained Multicriteria Optimization,* In INFOS 2008 pp. MM-73- MM-77