Indonesian Journal of Electrical Engineering and Computer Science
Vol. 15, No. 1, July 2019, pp. 373~381
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v15.i1.pp373-381

Review on population-based metaheuristic search techniques for optimal power flow

Muhammad Affiq Abd Rahman, Bazilah Ismail, Kanendra Naidu, Mohd Khairil Rahmat
Electrical Engineering Section, Universiti Kuala Lumpur British Malaysian Institute, Malaysia

Article Info

Article history:

Received Dec 18, 2018
Revised Feb 19, 2019
Accepted Feb 27, 2019

Keywords:

Heuristic search
Optimal power flow
Optimization
Power system
Swarm intelligence

ABSTRACT

Optimal power flow (OPF) is a non-linear solution which is significantly important in order to analyze the power system operation. The use of optimization algorithm is essential in order to solve OPF problems. The emergence of machine learning presents further techniques which capable to solve the non-linear problem. The performance and the key aspects which enhances the effectiveness of these optimization techniques are compared within several metaheuristic search techniques. This includes the operation of particle swarm optimization (PSO) algorithm, firefly algorithm (FA), artificial bee colony (ABC) algorithm, ant colony optimization (ACO) algorithm and differential evolution (DE) algorithm. This paper reviews on the key elements that need to be considered when selecting metaheuristic techniques to solve OPF problem in power system operation.

Corresponding Author:

Bazilah Ismail,
Electrical Engineering Section,
Universiti Kuala Lumpur British Malaysian Institute,
53100 Gombak, Selangor, Malaysia.
Email: bazilahismail@unikl.edu.my

1. INTRODUCTION

In recent decades, concentrated research effort were employed to solve complex optimization problems. Several methods are integrated to ensure the techniques able to operate in optimal rate. One of it is optimal power flow (OPF) problem in power system operation. The major concern is to reduce total operation cost while satisfying various constraints in the system. Computational intelligence assists to understand the problems with various perspective by introducing several metaheuristic techniques which are inspired by biological systems [1].

Evolutionary computation is used to find optimal solutions. This includes genetic algorithm (GA) which is used for search and optimization purposes; evolutionary algorithm (EA) which is excellent on decision making; and multi-agent based algorithm which works best on sorting and usually used for data mining. Traditionally, Newton-Raphson method that is used to identify the optimal values which is started by making an initial guess. This method requires large number of iterations to achieve optimal solution for the problems. Modern metaheuristic algorithm improves the process by using randomization to improve the convergence rate. Convergence of an algorithm is crucial to achieve optimal results which might take long time to converge. While reducing the time will lead to premature converge which means the solution provided does not guarantee global optima for the problem that being considered [2].

Several population-based heuristic search techniques are used to provide high quality optimal solution in acceptable timescale. Several changes in parameter setting are made to suit the nature of the optimization technique such as particle swarm optimization (PSO) algorithm, ant colony optimization (ACO) algorithm, artificial bee colony (ABC) algorithm, firefly algorithm (FA) and differential evolution (DE) algorithm. These algorithms used the iterative process and self-organize system to search the space of solution using evolutionary operation. Since it is impossible to create a universal algorithm which able to solve variety of optimization problems, researchers have come out with several metaheuristics with similar framework that contains several changes in the parameter. Although it is good to have variety of nature inspired algorithms such as glowworm swarm optimization (GSO) algorithm, monkey algorithm (MA), lion optimization algorithm (LOA), grey wolf optimization (GWO) algorithm and etc, the main focus is the needed to achieve a balanced exploration and exploitation process to achieve complete convergence with reasonable time based on the complexity of the problems. This paper reviews on the key elements that need to be considered to solve OPF problem in power system operation based on five population-based metaheuristic search techniques which are PSO, FA, ABC, ACO and DE.

2. OPTIMAL POWER FLOW PROBLEM FORMULATION

The combination of power flow equation and economic dispatch equation can be simplified into non-linear function. Standard OPF problem can be formulated to minimize the objective in the system and satisfy system equality and inequality constrains as:

$$\begin{aligned} &\text{Minimize} && f(x,u) \\ &\text{Subject to} && g(x,u) \leq 0 \\ & && h(x,u) = 0 \end{aligned} \tag{1}$$

where : x : The dependent variables
u : The independent variables
f (x,u) : Objective functions
g (x,u) : Set of inequality constraints
h (x,u) : Set of equality constraints

2.1. The Dependent Variable (State Vector)

$$x^T = [P_{G1}, V_{L1} \dots V_{LNL}, Q_{G1} \dots Q_{GNG}, S_{l1} \dots S_{ln1}] \tag{2}$$

where : PG1 : Generator active power at slack bus
VL : Load-bus voltage
QG : Generator reactive power
Sl : Transmission line flow
NL : Number of load buses
NG : Number of generators
nl : Number of transmission lines
T : Transformer tap setting

2.2. The Indipendent Variable (Control Vector)

$$u^T = [P_{G2} \dots P_{GNG}, V_{G1} \dots V_{GNG}, Q_{C1} \dots Q_{CNC}, T_1 \dots T_{NT}] \tag{3}$$

where : PG : Generator active power at PV buses (except at the slack bus PG1)
VG : Generator voltage at PV buses
QC : Shunt VAR compensation
NT : Number of regulating transformers
NC : Number of VAR compensators

2.3. The Equality Constrains (Typical Load Flow Equation)

$$P_{Gi} - P_{Di} - V_i \sum_{j=1}^{NB} V_j \, [G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)] = 0 \tag{4}$$

$$Q_{Gi} - Q_{Di} - V_i \sum_{j=1}^{NB} V_j \left[G_{ij}\cos(\delta_i - \delta_j) \right. \\ \left. + B_{ij}\sin(\delta_i - \delta_j)\right] = 0 \quad (5)$$

where : NB : Number of buses
PD : Active load demands
QD : Reactive load demands
Gij : Transfer conductance
Bij : Susceptance

2.4. The Inequality Constrains (Operation Limits of Equipment) Generator constraints:

$$V_{Gi}^{min} \le V_{Gi} \le V_{Gi}^{max}, \; i = 1, \ldots, NG \quad (6)$$

$$P_{Gi}^{min} \le P_{Gi} \le P_{Gi}^{max}, \; i = 1, \ldots, NG \quad (7)$$

$$Q_{Gi}^{min} \le Q_{Gi} \le Q_{Gi}^{max}, \; i = 1, \ldots, NG \quad (8)$$

Transformer constraints:

$$T_i^{min} \le T_i \le T_i^{max}, \; i = 1, \ldots, NT \quad (9)$$

Shunt VAR constraints:

$$Q_{Ci}^{min} \le Q_{Ci} \le Q_{Ci}^{max}, \; i = 1, \ldots, NC \quad (10)$$

Security constraints:

$$V_{Li}^{min} \le V_{Li} \le V_{Li}^{max}, \; i = 1, \ldots, NL \quad (11)$$

$$S_{li} \le S_{li}^{max}, \; i = 1, \ldots, nl \quad (12)$$

The OPF problems solution are designed by considering both normal and contingency operation conditions. This is to ensure that contingencies factor will not cause any violations of limits. Since the inequality constrains are self-limiting therefore, the optimization techniques using evolutionary computation techniques can be performed to overcome the problem.

3. PERSPECTIVE OF OPTIMIZATION ALGORITHM

3.1. Exploitation (Intensification) and Exploration (Diversification)

Exploitation or intensification in an algorithm is important to generate new solutions which better than existing solutions referring to the problem of interest. This allow high convergence rates but usually trapped in local search. However, the local search is limiting the end solution since it did not compare the information globally and it depends on the starting location. Exploration or diversification on the other hand, search for information globally and able to generate the solution with enough diversity. Since it needs to search in large space, this will slow down the convergence rate [1].

The balance between exploitation and exploration will be able to converge the solution faster and the potential to achieve global optima is high. The current algorithms use different parameter setting which cause certain algorithm to achieve fast convergence rate or able to achieve global optima. Randomization also been used by certain algorithms to make sure the convergence rate is fast. Since there are so many algorithms that perform very well in very specific problem, No Free Lunch (NFL) theorems is introduced. NFL states there is no universal algorithm that efficient enough for every problem. This is because the main purpose of OPF is to obtain high quality solution within acceptable timescale. So there is no reason to obtain average solution for all problems that being considered. However, some researchers suggest that NFL did exist on some types of problems such as coevolutionary approaches [3].

3.2. Evolutionary Operators

GA and EA uses genetic operators such as recombination (crossover), mutation and selection (survival of fitness) [4]. This allows diversity from mutation and crossover process; and also from the selection and survival of fitness high quality information can be obtained. This is important to solve complex and discontinuous problems. The rate of crossover and mutation will influence the global search rate (exploration) while the selection increase the exploitation action. In general, crossover is used to mix within a subspace and allowing to reach convergence of system. Mutation is a randomization technique which makes the iterative process for local search faster [2]. The selection is used to allow the system to evolve towards desired states. This happen by allowing the current solution compared with previous best solution, if the current solution generate better solution, the previous solution will be discarded. This repetitive process is carried out until the final best solution is achieved. Taxonomy of Evolutionary Computation (Optimization Algorithm) is shown in Figure 1.

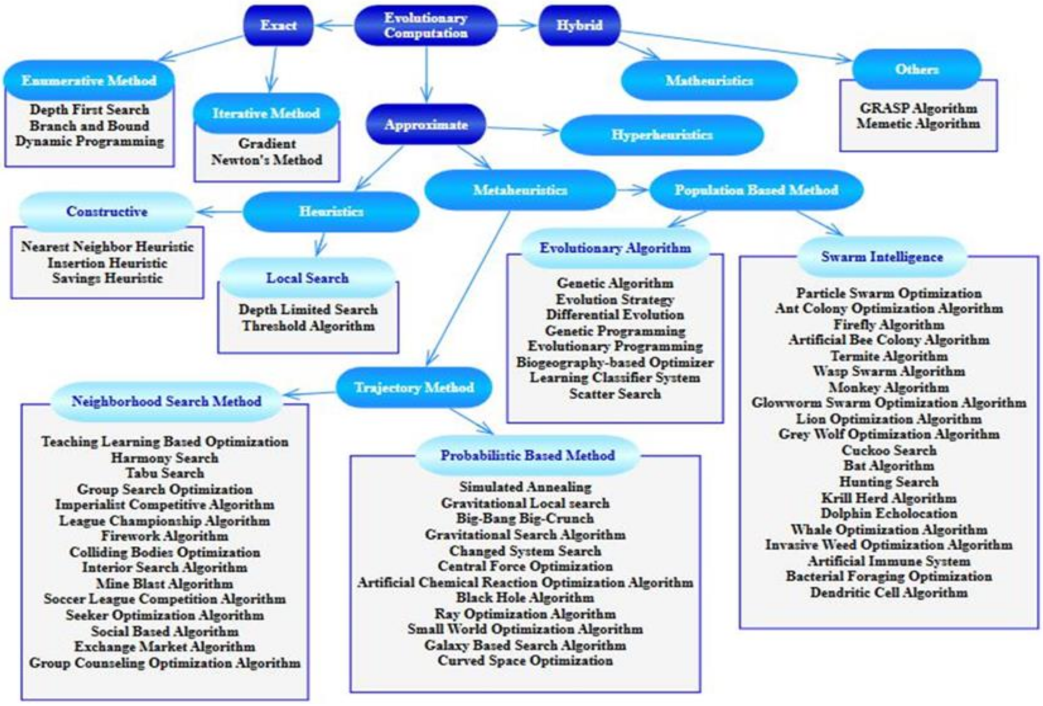
Evolutionary Computation
Exact
Hybrid
Enumerative Method
Depth First Search
Branch and Bound
Dynamic Programming
Iterative Method
Gradient
Newton's Method
Approximate
Matheuristics
Others
GRASP Algorithm
Memetic Algorithm
Hyperheuristics
Heuristics
Metaheuristics
Population Based Method
Constructive
Nearest Neighbor Heuristic
Insertion Heuristic
Savings Heuristic
Local Search
Depth Limited Search
Threshold Algorithm
Evolutionary Algorithm
Genetic Algorithm
Evolution Strategy
Differential Evolution
Genetic Programming
Evolutionary Programming
Biogeography-based Optimizer
Learning Classifier System
Scatter Search
Swarm Intelligence
Particle Swarm Optimization
Ant Colony Optimization Algorithm
Firefly Algorithm
Artificial Bee Colony Algorithm
Termite Algorithm
Wasp Swarm Algorithm
Monkey Algorithm
Glowworm Swarm Optimization Algorithm
Lion Optimization Algorithm
Grey Wolf Optimization Algorithm
Cuckoo Search
Bat Algorithm
Hunting Search
Krill Herd Algorithm
Dolphin Echolocation
Whale Optimization Algorithm
Invasive Weed Optimization Algorithm
Artificial Immune System
Bacterial Foraging Optimization
Dendritic Cell Algorithm
Trajectory Method
Neighborhood Search Method
Teaching Learning Based Optimization
Harmony Search
Tabu Search
Group Search Optimization
Imperialist Competitive Algorithm
League Championship Algorithm
Firework Algorithm
Colliding Bodies Optimization
Interior Search Algorithm
Mine Blast Algorithm
Soccer League Competition Algorithm
Seeker Optimization Algorithm
Social Based Algorithm
Exchange Market Algorithm
Group Counseling Optimization Algorithm
Probabilistic Based Method
Simulated Annealing
Gravitational Local search
Big-Bang Big-Crunch
Gravitational Search Algorithm
Changed System Search
Central Force Optimization
Artificial Chemical Reaction Optimization Algorithm
Black Hole Algorithm
Ray Optimization Algorithm
Small World Optimization Algorithm
Galaxy Based Search Algorithm
Curved Space Optimization

Figure 1. Taxonomy of Evolutionary Computation (Optimization Algorithm)

3.3. Heuristic Techniques

Heuristic technique is an approach to solve problem or learning using practical method to obtain satisfactory solution within reasonable time. This occurs by using similar rules from data set which has been trained to solve similar problems. Based on Figure 1, heuristic techniques can be classified into 4 categories which are heuristics, metaheuristics, matheuristics and hyperheuristics.

Matheuristics is a hybrid techniques that combines exact and approximate method to solve mathematical problems of interest using their exploitation features. These techniques uses improved mathematic programming techniques based on metaheuristics and vice versa [5]. This technique also shows significant improvement in local searching and local branching during neighborhood exploration [6], [7].

The approximate techniques which involves heuristics, metaheuristics and hyperheuristics applies exploitation and exploration. However, for heuristics method, it focus more on the exploitation which allows the problem able to obtain good solution in reasonable time. But they are usually trapped in local optima due to minimal exploration.

Metaheuristics techniques are suitable to optimize large and complex data. This is possible due to the behavior which accepts a temporary deterioration of the solution. Active exploration and exploitation allows it to randomly explore to obtain global optimum solution. However, the true balance of exploration and exploitation in any algorithms or problems still does not have a clear line and is supported with NFL theorem.

Hyperheuristics focus more on exploration compared to exploitation. The set space also not as usual space set of solution. This is because hyperheuristics is used as heuristics to search for optimal heuristics to be used. It is used to handle several classes of problem rather than solving just one problem [8]. The idea is to compensate the strength and weakness of known heuristics to select suitable method to be used for the assigned problems.

3.4. Classification of Metaheuristics

There are four main criteria that are used to classify various technique of metaheuristics. Firstly, the method used for every iteration is classified between trajectory and discontinuous methods. Usually trajectory methods is used for single point search in which the process just closed walk on the neighborhood graph. Discontinuous methods commonly used by population based techniques which allows it to make large jumps in the neighborhood graph for high exploration in the search space.

Second criteria is the involvement of memory in the algorithm. Since most of the techniques which is based upon either short term memory or long term memory, hence there is no clear group for this criteria. The purpose of having short term memory is to avoid cycling in similar solution by forbid the revisiting to the recent solution. Long term memory on the other hands, allows good diversification and intensification.

Third criteria is the way the algorithm interact with neighborhood structures. It is either kick-moves operation which interact with single neighbor or using operation that able to interact with various neighborhood such as mutation and crossover. Final criteria that been consider is the objective function during operation of the technique. Some algorithm uses static objective function to achieve optimal solution.

However, there is also algorithms which uses reward or penalty to modify the objective function. This dynamic objective function allows the search process to be guided in the correct direction to obtain global optimal solution. All four criteria are used to determine the main trait for each algorithm then classified into the group as shown in Figure 1.

4. POPULATION-BASED METAHEURISTIC SEARCH TECHNIQUES

The basic operation and theory to obtain OPF have been discussed. This section will focus on the exploration and exploitation. Techniques on current popular algorithms such as Particle Swarm Optimization (PSO), Firefly Algorithm (FA), Artificial Bee Colony Algorithm (ABC), Ant Colony Optimization (ACO) and Differential Evolution Algorithm (DE). Those five algorithms are compared due to the versatility in optimization process in various problem and the development in various study and hybridization techniques.

4.1. Particle Swarm Optimization Algorithm (PSO)

PSO was developed by Kennedy and Eberhart in 1995 which is based on the behavior of bird and fish schooling in nature [9], [10]. PSO is widely used due to the details explanation in algorithm operation in their book ‘Swarm Intelligent’ [11]. This allows PSO to be easily hybridized with other algorithms to compensate with the PSO limitation when dealing with several problems. The movement of particles depend on stochastic component and deterministic component. This two main components did used to determine the current global best location and current best location. Since the movement is random, active comparison occurred and allowing global best solution can be obtained over time. The vector can be represented as:

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^* - x_i^t] \tag{13}$$

$$v_i^{t+1} = x_i^t + v_i^{t+1} \tag{14}$$

where : x_i : Position of a particle
v_i : Velocity of a particle
ϵ_1, ϵ_2 : Random vectors that carry values between 0 and 1
α, β : Learning parameters or acceleration constants, typically $\alpha \approx \beta \approx 2$
g^* : Current global best solution
x_i^* : Individual best

Based on (13) and (14), PSO consists of mutation and selection as whole. Crossover is not present in PSO algorithm. This allows a high exploration rate in order for fast convergence rate. Since no crossover happens during the process, it is high possible that the current global best solution has prematurely converged.

However, there is possibility to achieve global optimal solution. The randomization for initial location allowing for the process take different route. Multiple random location will possibility lead to global optimal solution.

4.2. Firefly Algorithm (FA)

FA was developed by Xing-She Yang in 2008 which based on the flashing patterns and behavior of tropical fireflies [12]-[14]. The brightness of the flashing will be used for better exploration. The vector can be represented as:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{i/j}^2}\left(x_j^t - x_i^t\right) + \alpha \epsilon_i^t \tag{15}$$

where :	i	: Movement of a firefly
	j	: More attractive (brighter) firefly
	β_0	: Attractiveness at r = 0
	α	: Randomization parameter
	ϵ_i^t	: Vector of random numbers drawn

Based on (15), FA is easier to be implemented and flexible. The mutation that used from random numbers drawn from the Gaussian distribution and Levy flight allowing local and global search. The rate of searching can be controlled with randomization parameter. When compared with PSO, FA did not have global best solution which means FA did not undergo selection process.

However, FA uses ranking system which will be updated during the loops. The attraction feature is unique for FA which allows it to automatically classifies the solution into several groups [15]. This due to fact that the local attraction is stronger than the attraction from another location. The classification allows FA suitable to be used as multi objective problems and can be applied in various applications which are highly non-linear problems.

4.3. Artificial Bee Colony Algorithm (ABC)

ABC algorithm was developed by Karaboga in 2005 which is based on social behavior of bees [16]. ABC algorithm is divide into three group which are forager bees (employed), observer bees (onlooker) and scout bees. ABC did used the concentration of solution as their indicator which known as the amount of nectar intake from different food sources. Scouts and employed is used for mutation process to generate randomization then the solution will be selected by the onlooker. Employed bees will forage every food sources available with ratio of one bee per food source. When the food sources area been discard, forager bee will become scout to search for new food sources area by referring to the information given by onlooker bees. When compared to honey bee algorithm, ABC has advantage when the forager bees converted to scout directly. The vector for forager bees phase, observer bees phase and scout bee phase can be represented respectively as:

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \tag{16}$$

$$p_i = \frac{fit_i}{\sum_{j=1}^{S_N} fit_j} \tag{17}$$

$$x_{ij} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) rand(0,1) \tag{18}$$

where :	v_{ij}	: Source position
	S_N	: Size of employed or onlooker bees
	x_i	: *D*-dimensional vector (i =1, 2,..., S_N)
	j	: Number of optimization parameters ($j \in$ 1, 2,..., D)
	x_{ij}	: Comparison between two food positions
	k	: Random chosen indexes different from i ($k \in$ 1, 2,..., S_N)
	$\emptyset_{ij}$	: Random number between -1 and 1
	x_j^{min}	: lower bounds of parameter j
	x_j^{max}	: upper bounds of parameter j

Based on (16)–(18), there is no crossover to maximize their randomization. This allowing the algorithm to adapt with tough optimization since the operation is simplify with the specific task which already been assigned for the particular roles [17], [18]. ABC in general have the ability to globally search which cause the convergence rate relatively slow and the exploitation ability is limited due to the absence of crossover process.

4.4. Ant Colony Optimization Algortihm (ACO)

ACO algorithm was developed by Dorigo in 1999 which based on social behavior of ant [19-21]. When compared to FA which using attraction, ACO used similar techniques as ABC which uses the concentration of route (quality of solutions). This concentration shows the quality of the solution which is generated for discrete combinatorial problems. The vector can be represented as:

$$p_{ij} = \frac{\emptyset_{ij}^{\alpha}\, d_{ij}^{\beta}}{\sum_{i,j=1}^{n} \emptyset_{ij}^{\alpha}\, d_{ij}^{\beta}} \tag{19}$$

where :	i	: Probability in network at particular node (before)
	j	: Probability in network at particular node (after)
	$\emptyset_{ij}$	: Concentration on the route between i and j
	d_{ij}	: Desirability of the same route

Based on (19), ACO did used evaporation over time features to avoid premature convergence. The main technique by ACO is the probability of choosing a route and the evaporation rate [22]. The best solution will display with high concentration of route due to the number of route used is higher due to traveling time is short, thus less amount of evaporated within that short period. This short term memory allowing the best route remain higher in concentration without causing any redundant from another local maximum which been discovered previously.

4.5. Differential Evolution Algorithm (DE)

DE algorithm developed by Rainer and Kenneth in 1997 which based on creating new off-spring from parent chromosomes instead of classical crossover or mutation [23]–[25]. DE did used crossover, mutation and selection. The unique of DE is the self-adaptive mechanism and selection process cause by their mutation scheme. DE used greedy selection process that provide slightly advantages of converging performance for the new and parent solution that win the selection [24], [26], [27]. DE can be stated into four stages which are initialization, mutation, crossover and selection.

During the initialization, the component can be represented as:

$$x_{i,j}(0) = x_j^L + rand(0,1)\left(x_j^u - x_j^L\right) \tag{20}$$

where :	$rand(0,1)$	: Uniformly distributed random number
	x_j^u	: parameter of upper bound
	x_j^L	: parameter of lower bound
	i	: Population members
	j	: Component of population

While the mutation phase, donor vector is created from the changes of population member which can be represent as:

$$v_{i,j}(t+1) = x_{r\,1,j}(t) + F(x_{r\,2,j}(t) - x_{r\,3,j}(t)) \tag{21}$$

where : $x_{i(t)}$: Changes of population member
$v_{i(t)}$: Donor vector
x_{r1}, x_{r2}, x_{r3} : three distinct points taken randomly from the current population and not coinciding with the current x_i
F : Scalar number which between 0.4 and 1.0

The crossover process did have two crossover schemes that commonly used which are exponential and binomial. This allowing diversity of population. Next, the process that ensures the population size remains constant is known as selection stage. The selection process is determined by the concept of survival of fitness which gives advantage for the new solution and their parents. This process can be represented as:

$$\begin{aligned} x_i(t+1) &= u_i(t) \quad \text{if} \quad (f\,(u_i(t)) \le f\,(x_i(t)) \\ x_i(t+1) &= x_i(t) \quad \text{if} \quad (f\,(x_i(t)) < f\,(u_i(t)) \end{aligned} \tag{22}$$

where : $u_{i,j}(t)$: The crossover information
$f\,()$: Function to be minimized

5. CONCLUSION

The main key for evolutionary operators to ensure optimal result are crossover, mutation and selection. Crossover is efficient in exploitation. While with the extensive usage of mutation and selection enables the ability of exploration. The current global best helps to speed up the convergence but there is probability for premature convergence. Therefore, the parameter setting for the algorithm is crucial to ensure the balance between exploitation and exploration which will improve the overall performance of the algorithm. The complexity of the algorithm also needs to be considered as a form of user friendliness. The research on the parameter setting for various problem is still insufficient. This can be done by understanding the NFL theorem on problems that it is applied on. The hybridization of several algorithms is useful to enhance the rate and result of the convergence.

ACKNOWLEDGMENT

The authors would like to thank the Ministry of Higher Education of Malaysia and Universiti Kuala Lumpur (UniKL) for providing financial support under the research grant No. FRGS/1/2017/TK04/UNIKL/03/3 and str17071.

REFERENCES

[1] Chakraborty and A. K. Kar, “Swarm Intelligence : A Review of Algorithms Swarm Intelligence : A Review of Algorithms,” no. March, 2017.

[2] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 189–213, 2003.

[3] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67– 82, 1997.

[4] J. H. Holland, Adaptation in Natural and Artificial Systems. MA: MIT Press, 1992.

[5] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Röhler, “Matheuristics: Optimization, Simulation and Control,” Hybrid metaheuristics, pp. 171–177, 2009.

[6] I. Dumitrescu and T. Stützle, “Usage of exact algorithms to enhance stochastic local search algorithms,” *Matheuristics*, pp. 103–134, 2010.

[7] M. Fischetti and A. Lodi, “Local branching,” *Math. Program.*, vol. 98, no. 1–3, pp. 23–47, 2003.

[8] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, “Hyper-Heuristics: An Emerging Direction in Modern Search Technology,” *Handb. Metaheuristics*, pp. 457– 474.

[9] J. Kennedy and R. Eberhart, “Particle swarm optimization,” Neural Networks, 1995. Proceedings., IEEE Int. Conf., vol. 4, pp. 1942–1948 vol.4, 1995.

[10] M. N. Alam, “Particle Swarm Optimization : Algorithm and its Codes in MATLAB Particle Swarm Optimization : Algorithm and its Codes in MATLAB,” no. March, 2016.

[11] K. Eberhart, Swarm Intelligence. 2008.

[12] X.-S. Yang, “Firefly Algorithm, Stochastic Test Functions and Design Optimisation,” vol. 2, no. 2, p. 2010, 2010.

[13] X. S. Yang, “Firefly algorithms for multimodal optimization,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5792 LNCS, pp. 169– 178, 2009.

[14] X.-S. Yang, “Metaheuristic Optimization,” Cambridge Univ. UK, vol. 6, no. 2011, p. 15, 2011.

[15] T. Govindaraj and V. Tamildurai, “Firefly Algorithm for Optimal Power Flow Considering Control Variables,” vol. 2, no. 2, pp. 1116–1121, 2014.

[16] D. Karaboga, “An idea based on Honey Bee Swarm for Numerical Optimization,” Tech. Rep. TR06, Erciyes Univ., no. TR06, p. 10, 2005.

[17] M. Ding, H. Chen, N. Lin, S. Jing, F. Liu, and X. Liang, “Dynamic population artificial bee colony algorithm for multiobjective optimal power flow,” Saudi J. Biol. Sci., vol. 24, no. 3, pp. 703–710, 2017.

[18] L. Le Dinh, D. V. Ngoc, and P. Vasant, “Artificial Bee Colony Algorithm for Solving Optimal Power Flow Problem,” vol. 2013, 2013.

[19] M. Dorigo, G. Di Caro, and L. M. Gambardella, “Ant algorithms for distributed discrete optimization,” *Artifical Life*, vol. 5, pp. 137–172, 1999.

[20] J. Soares et al., “Ant Colony Search Algorithm for the Optimal Power Flow Problem,” pp. 1–8, 2011.

[21] K. Lenin, B. R. Reddy, and M. S. Kalavathi, “Optimal Power Flow using Ant Colony Search Algorithm to Evaluate Load Curtailment Incorporating Voltage Stability Margin Criterion,” vol. 3, no. 5, 2013.

[22] F. M. Tuaimah, “Ant Colony Optimization based Optimal Power Flow Analysis for the Iraqi Super High Voltage Grid,” vol. 67, no. 11, pp. 13–18, 2013.

[23] R. Storn and K. Price, “Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[24] A. A. A. El and E. M. A. Abido, “Optimal power flow using differential evolution algorithm,” pp. 69–78, 2009.

[25] P. P. Biswas, P. N. Suganthan, R. Mallipeddi, and G. A. J. Amaratunga, “Engineering Applications of Artificial Intelligence Optimal power flow solutions using differential evolution algorithm integrated with effective constraint handling techniques,” *Eng. Appl. Artif. Intell.*, vol. 68, no. October 2017, pp. 81–100, 2018.

[26] A. Shrivastava and H. M. Siddiqui, “A Simulation Analysis of Optimal Power Flow using Differential Evolution Algorithm for IEEE-30 Bus System,” vol. 2, no. 3, pp. 50–57, 2014.

[27] R. K. Verma, H. Singh, and L. Srivastava, “Optimal Power Flow Using Differential Evolution Algorithm With Conventional Weighted Sum Method,” vol. 2, no. 3, pp. 681–685, 2012.