Energy 77 (2014) 372–381

Contents lists available at ScienceDirect

Energy

journal homepage: www.elsevier.com/locate/energy

Solving non-convex economic dispatch problem via backtracking search algorithm

Mostafa Modiri-Delshad [a, c], Nasrudin Abd Rahim [a, b, *]

[a] UM Power Energy Dedicated Advanced Center (UMPEDAC), Level 4, Wisma R&D University of Malaya, JalanPantai Baharu, 59990 Kuala Lumpur, Malaysia
[b] Renewable Energy Research Group, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[c] Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 25 April 2014
Received in revised form
15 August 2014
Accepted 3 September 2014
Available online 5 October 2014

Keywords:
Economic dispatch
Backtracking search algorithm
Valve point effect
Transmission loss
Non-convex
Evolutionary

ABSTRACT

This paper presents BSA (backtracking search algorithm) for solving of ED (economic dispatch) problems (both convex and non-convex) with both the valve-point effects in the generator cost function and the transmission network loss considered. BSA is a new evolutionary algorithm for solving of numerical optimization problems; it uses a single control parameter and two crossover and mutation strategies for powerful exploration of the problem's search space. Four test systems (with 3, 6, 20, and 40 generators) are the case studies verifying the method's robustness and effectiveness. The results confirm that compared with existing well-known methods and especially in large-scale test systems, the proposed algorithm is the better approach to solving ED problems.

1. Introduction

The problem of ED (economic dispatch) is a basic consideration to optimizing power system operation. ED determines the power shared among the generating units of power system to meet electrical demand while minimizing cost and satisfying system constraints.

In a convex ED problem, the cost function of a generating unit is considered as a quadratic function. Practical and non-convex ED problems, however, contain non-convex cost functions that are due to the valve-point effect of the generating units. Classical methods have been adopted to solve conventional ED problems (i.e., containing convex cost functions) but instead produce non-optimal solutions because of the non-convexity/non-linearity of practical ED problems [1]. Dynamic programming, for example, has been proposed in addressing non-convex ED problems because it does not restrict the form of the cost function; the increased dimension of the problem, however, may demand higher computational efforts [2]. Classical methods include interior point [3], quadratic programming [4], linear programming [5], Lagrangian relaxation algorithm [6], dynamic programming [7], and lambda iteration [8].

Unlike classical methods, metaheuristic methods are better options because they can handle more constraints and are able to explore the search domain effectively in finding the optimum; they include ICA (imperialist competitive algorithm) [9], CS (cuckoo search) [1], DE (differential evolution) [10], ABC (artificial bee colony) [11], PSO (particle swarm optimization) [12], TLBO (Teaching–learning-based optimization) [13], SOA (seeker optimization algorithm) [14], MGSO (modified group search optimizer) [15], GA (genetic algorithm) [16], and HBMO (honey bee mating algorithm) [17]. DE is especially very effective because it does not need derivative information from the cost function; instead it sub-optimally or prematurely converges [17]. Other drawbacks associated with metaheuristics are high sensitivity to the control parameters, long computational time, and slow convergence to approximately optimum solution [18].

Recent hybrid methods overcome those drawbacks, able to handle the high complexities of practical ED problems. One method might be adopted for its high convergence, another for its provision of a suitable initial guess for the problem. The hybrid methods are combinations of either two or more metaheuristic methods or metaheuristic with classical techniques. Combinations of PSO with DE [17], GA with API [19], GA-LI [20], CPSO-SQP [21], and FCASO-SQP [22] perform better as hybrids than individually.

* Corresponding author. UM Power Energy Dedicated Advanced Center (UMPEDAC), Level 4, Wisma R&D University of Malaya, JalanPantai Baharu, 59990 Kuala Lumpur, Malaysia. Tel.: +60 3 22463246; fax: +60 3 22463257.
E-mail addresses: modiri.d@gmail.com (M. Modiri-Delshad), nasrudin@um.edu.my (N.A. Rahim).

http://dx.doi.org/10.1016/j.energy.2014.09.009
0360-5442/

BSA (Backtracking search algorithm) is a new evolutionary algorithm developed by Civicioglu [23]. It has been successfully applied to many high-dimensional multimodal optimization benchmarks. Statistical analysis of its results confirms its superiority and performance over several other widely used evolutionary methods of optimization.

This paper proposes BSA as an approach to solving both convex and non-convex ED problems. The transmission network loss is modeled in consideration of the network topology whereas the valve-point effect is considered for accurate modeling of the generator cost. BSA's performance in solving ED problems is compared with other popular methods in terms of solution quality. The paper is organized next as follows: Section 2 presents the proposed method's algorithm, Section 3 provides the mathematical model of the ED problem considering the transmission loss, Section 4 presents the method's application to four test systems (the case studies), and Section 5 presents the results analysis.

2. BSA (Backtracking search optimization algorithm)

Evolutionary algorithms use the techniques inspired by natural or biological evolutions such as mutation, crossover, and selection to generate solutions for optimization problems. BSA (Backtracking search algorithm) is also an evolutionary algorithm for solving of constrained optimization problems. It was developed to overcome some of the drawbacks of evolutionary methods; e.g., high sensitivity to the control parameters, time-consuming computation, and premature convergence. Its structure is simple and it has only one control parameter to solve even multimodal optimization problems. Also, it is not too sensitive to the initial parameter value, and performs effectively and fast. It is able to explore the search domain effectively to find the optimum, through crossover and mutation operators that are different from the ones defined in past evolutionary methods. It also has the advantage of a memory that stores the population of past generations and directs the search. Its evolutionary feature is similar to that of the return of a social group of creatures to a past life situation advantageous to getting food.

BSA comprises five processes: initialization, selection-I, mutation, crossover, and selection-II. Its general structure is as shown in Fig. 1. Its major steps are described next.

2.1. Initialization

This process is the generation of an initial population through Eq. (1). The population size and number of optimization variables are respectively represented by nPop and nVar.

$$P_{ij} \sim U\left(\mathrm{low}_j, \mathrm{up}_j\right) \tag{1}$$

where:

i: stands for individual number. $i = \{1,2, \ldots, \mathrm{nPop}\}$
j: stands for variable number. $j = \{1,2,3, \ldots, \mathrm{nVar}\}$

```
1. Initialization
repeat
    2. Selection-I
    Generation of Trial-Population
        3. Mutation
        4. Crossover
    end
    5. Selection-II
until stopping conditions are met;
```

Fig. 1. General structure of BSA [23].

U: uniform distribution function
P_i: the target individual which is a member of population

2.2. Selection-I

In this stage, BSA generates the historical population old$P$, used as the search direction. The historical population is also initialized by Eq. (2).

$$\mathrm{old}P_{ij} \sim U\left(\mathrm{low}_j, \mathrm{up}_j\right) \tag{2}$$

During each iteration, BSA has an option to redefine old$P$ by comparing between two randomly generated numbers a and b through the ‘if-then’ rule and according to Eq. (3):

$$\text{if } a < b\Big|_{a,b \sim U(0,1)} \rightarrow \mathrm{old}P = P \tag{3}$$

BSA memory remembers old$P$ as the historical population in each iteration until it is changed. After determining the historical population based on the aforementioned equation, BSA changes the order of the individuals in old$P$ randomly through Eq. (4).

$$\mathrm{old}P = \mathrm{permuting}(\mathrm{old}P) \tag{4}$$

A random shuffling function is used as the permuting function in Eq. (4).

2.3. Mutation

The mutation process of BSA generates Mutant as the initial form of the trial population through Eq. (5).

$$\mathrm{Mutant} = P + F.(\mathrm{old}P - P) \tag{5}$$

where F is a function that controls the amplitude of the search-direction matrix (the difference between the population and the historical population); a normal distribution function can be considered for it. The value of $F = 3.\mathrm{randn}$ is used in this paper where randn ~ $N(0,1)$ (N is the standard normal distribution).

2.4. Crossover

The crossover process is performed to finalize the trial population ‘Mutant’ as set in the preceding stage. In this process, the initial trial population ‘Mutant’ is changed to the final trial population ‘T’ through a crossover operator.

The value of ‘T’ is first set to ‘Mutant’, then a binary matrix (map) with nPop rows and nVar columns is generated to select the individuals that have to be manipulated. If $\mathrm{map}_{i,j} = 1$, where $i = \{1,2, \ldots, \mathrm{nPop}\}$ and $j = \{1,2,3, \ldots, \mathrm{nVar}\}$, the individual $T_{i,j}$ is updated with P_{ij} which means $T_{ij} = P_{ij}$.

A single parameter with the name ‘mixrate’ controls the number of elements of individuals to be engaged in the crossover process. Some individuals of T may violate the boundary condition of the optimization problem, so they are regenerated at the end of the crossover process, as Eq. (1) demonstrates.

2.5. Selection-II

In Selection-II stage, the individuals of population T with better fitness values than the corresponding particles of population P are used to update P. The global minimum among all the individuals is also updated according to the fitness values for T and P.