JSCDSS
An International Journal

http://www.jscdss.com
Vol.4 No.6 December 2017: 1-6
Article history:
Accepted 28 October 2017
Published online 28 October 2017

Journal of Soft Computing and Decision Support Systems

E-ISSN: 2289-8603

Metaheuristic Algorithms: Guidelines for Implementation

Ashkan Memari [a,*], Robiah Ahmad [a], Abd. Rahman Abdul Rahim [a]
[a] Department of Engineering, UTM Razak School of Engineering and Advanced Technology
Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100, Kuala Lumpur, Malaysia

*** Corresponding author email address**: ashkan.memari@utm.my

Abstract

This paper presents a quick review of the basic concepts and essential steps for implementing of metaheuristic algorithms. It can be therefore used as a roadmap to shed light on solving an optimization problem using a metaheuristic algorithm. We provide a brief review of the topics, including general concepts for metaheuristics, the need to design metaheuristics, the need for further improvement of metaheuristics, parameters tuning and performance assessment of metaheuristic algorithms. Finally, the paper ends with a guideline framework which aims to assist new researchers for solving optimization problems via metaheuristics.

Keywords: Metaheuristic algorithms, Optimization, Literature review, Performance assessment, Guidelines for implementation

1. Introduction

Heuristic means ‘to find’ or ‘to discover solutions by trial and error’. Heuristic methods aim at searching feasible solutions for large-scale problems (Yang, 2010). They have proven to overcome many shortcomings of traditional optimization methods, however, they are typically experience-based methods. Heuristic methods find the solution which hopes to be near optimal solution, rapidly. However, heuristic techniques are not considered effective for the complex problems. There are two main reasons for this:

i. Although heuristic techniques are usually (not always) provide a decent and acceptable answer in solving complex problems; they do not promise and guarantee the optimal solution for solving complex problems (Chinneck, 2004).

ii. Heuristic techniques are highly depending on experience and mathematical knowledge of the model developer (Coello et al., 2007).

The development of heuristic algorithms is named metaheuristic algorithms. The term “meta” means “beyond” or “higher level”. Metaheuristics often perform better compare to simple heuristics. Basically, metaheuristic algorithms use certain trade-offs of randomization and local search (Yang, 2010). Since metaheuristic algorithms conduct a more thorough search, they have rapidly become the favoured methodology for generating solutions to complex real-life problems which exact methods are unable to solve (Glover and Kochenberger, 2003). There are several available classifications for metaheuristic algorithms in the literature. Fig. 1 presents a general classification of metaheuristic algorithms.

2. The need to design a metaheuristic algorithm: complexity of problems

The main branch of the theory of computation, which is known as computational complexity theory, aims at sorting problems concerning their inherent difficulty. The complexity associated with a problem is actually equivalent to the finest algorithm tackling that problem. Furthermore, the complexity theory concerns with the decisions which they always have a ‘yes’ or ‘no’ answer. In this line, a problem is called easy or tractable if a polynomial-time algorithm can solve it. If no polynomial-time algorithm can solve a problem, it is known as a difficult or intractable problem.

Another goal in computational theory is to sort problems into specified complexity classes. For a given amount of computational resources, a complexity class corresponds to the set of all problems that can be solved. *P (*polynomial time*)* and *NP* (nondeterministic polynomial time) are two main classes of problems in complexity classes (Coello et al., 2007).

A problem is in *P* class if a deterministic algorithm can solve all the decision problems in polynomial time and a problem is *NP* if a non-deterministic algorithm can solve all decision problems in polynomial time (Talbi, 2009). In this line, one of the central open questions is whether $P = NP$ or not? The answer would have a broad impact on