A Comparison of Three Recent Nature-Inspired Metaheuristics for the Set Covering Problem

Broderick Crawford[1,2,3(✉)], Ricardo Soto[1,4,5], Cristian Peña[1], Marco Riquelme-Leiva[1], Claudio Torres-Rojas[1], Sanjay Misra[6], Franklin Johnson[1,7], and Fernando Paredes[8]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{broderick.crawford,ricardo.soto}@ucv.cl,
{cristian.pena.v,claudio.torres.r}@mail.pucv.cl,
marcoriquelmeleiva@gmail.com, franklin.johnson@upla.cl
[2] Universidad San Sebastián, Santiago, Chile
[3] Universidad Central de Chile, Santiago, Chile
[4] Universidad Autónoma de Chile, Santiago, Chile
[5] Universidad Cientifica del Sur, Lima, Perú
[6] Covenant University, Ota, Nigeria
sanjay.misra@covenantuniversity.edu.ng
[7] Universidad de Playa Ancha, Valparaíso, Chile
[8] Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. The Set Covering Problem (SCP) is a classic problem in combinatorial optimization. SCP has many applications in engineering, including problems involving routing, scheduling, stock cutting, electoral redistricting and others important real life situations. Because of its importance, SCP has attracted attention of many researchers. However, SCP instances are known as complex and generally NP-hard problems. Due to the combinatorial nature of this problem, during the last decades, several metaheuristics have been applied to obtain efficient solutions. This paper presents a metaheuristics comparison for the SCP. Three recent nature-inspired metaheuristics are considered: Shuffled Frog Leaping, Firefly and Fruit Fly algorithms. The results show that they can obtainn optimal or close to optimal solutions at low computational cost.

Keywords: Set Covering Problem · Metaheuristics · Shuffled Frog Leaping Algorithm · Firefly algorithm · Fruit fly algorithm

1 Introduction

The Set Covering Problem (SCP) is defined as follows, let $A = (a_{ij})$ be an m-row, n-column, zero-one matrix. We say that a column j covers a row i if $a_{ij} = 1$. Each column j is associated with a non negative real cost c_j. Let $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ be the row set and column set, respectively. The SCP calls

O. Gervasi et al. (Eds.): ICCSA 2015, Part IV, LNCS 9158, pp. 431–443, 2015.
DOI: 10.1007/978-3-319-21410-8_34

for a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$. A mathematical model for the SCP is:

$$Minimize \quad f(x) = \sum_{j=1}^{n} c_j x_j \tag{1}$$

subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad \forall i \in I \tag{2}$$

$$x_j \in \{0, 1\}, \qquad \forall j \in J \tag{3}$$

The SCP has many practical applications like location of emergency facilities [5], airline and bus crew scheduling [4,15], steel production [11], logical analysis of numerical data [3], ship scheduling [12], vehicle routing [1]. The SCP has been solved using complete techniques and different metaheuristics [6,7,17].

This work proposes to solve the SCP with three recent Nature-inspired metaheuristics: Shuffled Frog Leaping Algorithm (SFLA), Modified Binary FireFly Algorithm (MBFF) and Binary Fruit-Fly Algorithm (bFFOA).

SFLA is based on the observation, the imitation and the modeling of the behavior of a group of frogs searching a location that has the maximum available quantity of food [10].

Firefly algorithm was presented by Yang [19] and its operation, is based on the social behaviour of fireflies.

The Fruit Fly Optimization Algorithm (FFOA) was created by Pan [16] and it is inspired by the knowledge from the foraging behavior of fruit flies in finding food.

2 Shuffled Frog Leaping Algorithm

In SFLA a set of frogs are generated randomly. Then, the population is divided in frog subgroups named *memeplexes*. For each subgroup, a local search is realized to improve the position of the worst frog, which in turn can be influenced by other frogs since each frog has ideas affecting others. This process is named *evolution memetica*, which can repeat up to a certain number of iterations. The ideas generated by every memeplexe are transmitted to other memeplexes in a process of redistribution [14]. The local search, the evolution memetica and the redistribution they continue until the criterion of stop is reached [9].

The initial population of frogs P is created at random. This means that for a problem of n variables, a frog i is represented as a vector $X_i = (x_i^1, x_i^2, \ldots, x_i^n)$. Then, the fitness is calculated for each frog and they are arranged in descending order according to the obtained fitness. m memeplexes are generated from the division of the population, each subgroup contains f frogs (i.e. $P = m \times f$). In this process, the first frog goes to the first memeplexe, the second frog goes to the second memeplexe, frog f goes to the memeplexe m, and the frog $f + 1$ goes back to the first memeplexe, ...