A Binary Fruit Fly Optimization Algorithm to Solve the Set Covering Problem

Broderick Crawford[1,2,3](✉), Ricardo Soto[1,4,5], Claudio Torres-Rojas[1], Cristian Peña[1], Marco Riquelme-Leiva[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,
{claudio.torres.r,cristian.pena.v}@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 well known $\mathcal{NP}$*-hard* problem with many practical applications. In this work binary fruit fly optimization algorithms (bFFOA) were used to solve this problem using different binarization methods.

The bFFOA is based on the food finding behavior of the fruit flies using osphresis and vision. The experimental results show the effectiveness of our algorithms producing competitive results when solve the benchmarks of SCP from the OR-Library.

Keywords: Set Covering Problem · Fruit Fly Optimization Algorithm · Metaheuristics · Combinatorial optimization problem

1 Introduction

The Set Covering Problems (SCP) can be formulated as follows: [11]

$$\text{minimize} \quad Z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

Subject to:

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

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

Let $A = (a_{ij})$ be a $m \times n$ 0-1 matrix with $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ be the row and column sets respectively. We say that column j can be cover a row i if $a_{ij} = 1$. Where c_j is a nonnegative value that represents the cost of selecting the column j and x_j is a decision variable, it can be 1 if column j is selected or 0 otherwise. The objective is to find a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$. The SCP was also successfully solved with meta-heuristics such as taboo search [7], simulated annealing [6], artificial bee colony [8], genetic algorithm [12,13,15], ant colony optimization [1,18], swarm optimization particles [9,19] and firefly algorithms [10].

2 Binary Fruit Fly

The Fruit Fly Optimization Algorithm (FFOA) was created by Pan [17] and it is based on the knowledge from the foraging behavior of fruit flies or vinegar flies in finding food represented in figure 1. The traditional FFOA consists of 4 phases. These are initialization, osphresis foraging search, population evaluation, and vision foraging search. In the initialization phase, the fruit flies are created randomly and they have very sensitive osphresis and vision organs which are superior to other species. In osphresis foraging phase, flies use their osphresis organ to feel all kinds of smells in the air and fly towards the corresponding locations. Flies are evaluated to find the best concentration of smell. When they are near food, in the last phase, flying toward it using its vision organ.

The FFOA is used to solve continuous problems such as: the financial distress [17], web auction logistics service [14], power load forecasting [21] and multidimensional knapsack problem [20].The last, was solved with a new FFOA-based algorithm, which was created by Wang [20], the Binary Fruit Fly Optimization Algorithm (bFFOA).

This algorithm, in contrast from traditional FFOA, the author used: a discrete binary string to represent a solution, a probability vector to generates the population; they adopted change zero to one (or vice versa) to exploit the neighborhood in the smell-based search process; and made a global vision-based search method to improve the exploration ability. The bFFOA was divided in four phases: Initialization, and three search methods: Smell-based, Local-Vision-based and Global-Vision-based. In this paper, the bFFOA was adapted to improve it with other transfer functions, and discrete methods that be will explain in this paper.

The problems that we solve with the algorithm can be downloaded from Beasley OR-Library [1], this files was test in [4,5]. The binary FFOA was divided in the following steps:

[1] http://people.brunel.ac.uk/~mastjjb/jeb/info.html