PUPIBinary

A tool for binary function optimisation
inspired in the behaviour of urban pigeons

User Guide

Version 1.4

Sergio A. Rojas, PhD.

Universidad Distrital Francisco José de Caldas

Bogotá, Colombia, 2020

PUPIBinary Version 1.4 - User guide.

Copyright c 2020 Sergio A. Rojas

This document is distributed under the CC BY-NC-ND license (*Creative Commons Attribution-Noncommercial-NoDerivatives 3.0*). Any other unauthorised form of distribution, copying, duplication, reproduction, or sale (total or partial) of the content of this document, both for personal and commercial use, will constitute an infringement of copyright. This guide is an original work of its author, and therefore it is protected by the laws that regulate copyright and intellectual property. The opinions and points of view expressed in this document are personal to the author and do not compromise the policies, intentions, strategies, or official position of any other organism, company, organisation, service or person mentioned in it.

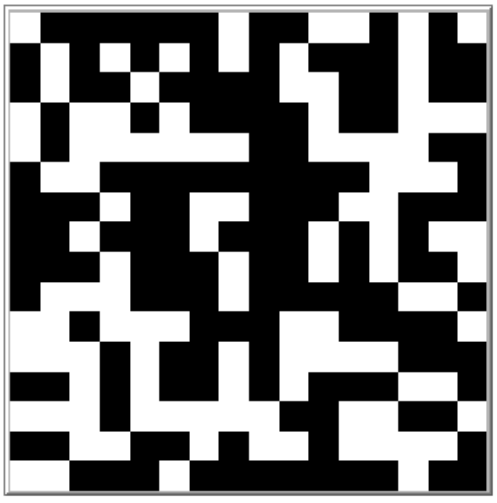
The author has made every effort to ensure that this guide is free from errors or omissions. However, the author accept no responsibility for offence, damage or loss caused to any person acting or endorsing actions using the material contained in this document.

First Edition, August 2020
Bogotá, Colombia

Overview

PUPIBinary is a software tool designed to find approximate solutions to optimisation problems whose decision variables take discrete values in the binary domain, in other words, variables taking values 0 or 1. The method used by PUPIBinary to find a solution, is derived from a recently proposed particle swarm algorithm inspired on the foraging behaviour of urban pigeons. The new variant of the algorithm is obtained by mapping the real–valued search space of the original version into a discrete binary–valued encoding. Furthermore, the extended version was adapted to the framework of an agent–based model in order to develop this software tool.

The approach to translate the original pigeon-inspired method consist of separating the search space of real–valued vector of pigeon locations from the solution space of binary-valued variables of the optimisation problem. This can be done by means of an encoding function, a technique known as genotype–to–phenotype mapping widely used in evolutionary computation algorithms. In the current version, the mapping is performed by applying a cut-off threshold to the coordinates of the pigeons' locations. In this way, the tool is able to solve high-dimensional binary problems of up to 256 variables.



PUPIBinary v1.4 has been released under GNU General Public License (GPLv3); it is available online at:

http://modelingcommons.org/browse/one_model/6400

Contents

Overview iii

1 The optimisation tool 1
- 1.1 What is PUPIBinary? . . . 1
- 1.2 How it works . . . 2
- 1.3 How to use it . . . 3
- 1.4 Other distinctive features . . . 5
- 1.5 Try it yourself . . . 5
- 1.6 Extending the tool . . . 5
- 1.7 References . . . 6

2 Installation and execution 7
- 2.1 Online version . . . 7
- 2.2 Desktop version . . . 8

3 Source code 11

4 Software license 17

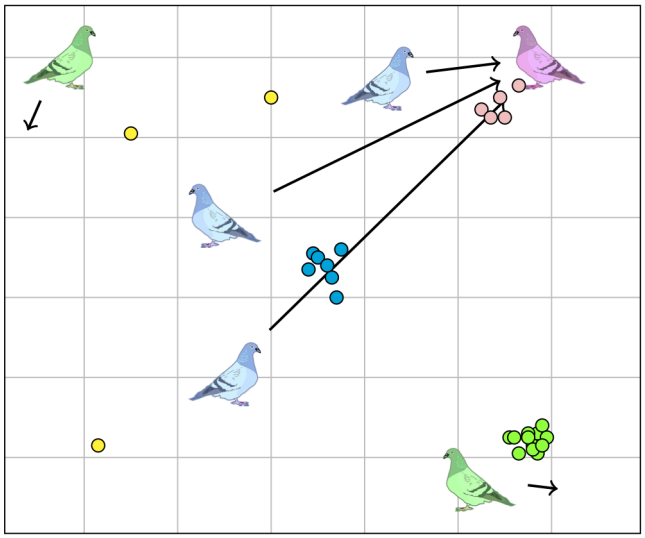
Chapter 1

The optimisation tool

1.1 What is PUPIBinary?

PUPIBinary is a software tool developed as an extension of an agent-based model for numeric real-valued unconstraint optimisation to binary domains, inspired in the foraging behaviour of urban pigeons (see [1] for details). Optimisation problems with binary domains contemplate variables taking values 0 or 1, representing decisions associated to conditions being false or true. Hence, these kind of domains are of interest in many applications in engineering and management.

The tool maintains an internal representation of the pigeons locations which is transformed to binary values via a threshold function. The movement rules of the internal representation are identical to the continuous-valued version, and the inspiration of the original method regarding foraging strategies of this species, is sustained: a leader pigeon (coloured fuchsia) located at a promising source of food is pursued by a flock of follower pigeons (coloured blue), while simultaneously other pigeons are walking around (coloured green), exploring the space for new sources of food too. Food sources in this context refers to maximal values of the binary cost function that is being optimised.



1.2 How it works

A benchmark of binary problems is included. Each PROBLEM consists of finding a bitstring that maximises a cost function. In the current version, the tool features three functions: "oneMax", "powSum" and "squareWave", with the following definitions:.

Problem	Definition	Description
oneMax	$f(\mathbf{b}) = \sum_{i=1}^{d} b_i$	Counts the total number of ones in the vector **b**. The optimum is: $f(\underbrace{1,1,\ldots,1}_{d \text{ times }!}) = d$
squareWave	$f(\mathbf{b}) = \sum_{i=1}^{d} \left(1 \quad \texttt{abs}\left(b_i \quad \left(2\left\lfloor \frac{i}{T} \right\rfloor \quad \left\lfloor \frac{2i}{T} \right\rfloor\right)\right)\right)$	Computes the similarity of vector **b** to a binarised version of a sinusoidal wave with period $T = \overline{p}\overline{d}$. The optimum is: $f(\underbrace{0,0,0,1,1,1,\ldots}_{\frac{d}{T} \text{ bits } !},\underbrace{0,0,0,1,1,1}_{\frac{d}{T} \text{ bits } !}) = d$
powSum	$f(\mathbf{b}) = \sum_{i=1}^{d} b_i \quad \log_2 2^{i \ 1}$	Computes the sum of the powers corresponding to positions set to one in the binary vector **b**. The optimum is, likewise oneMax, the all-ones vector: $f(\underbrace{1,1,\ldots,1}_{d \text{ times }!}) = \sum_{i=0}^{d} i = \frac{d(d+1)}{2}$

Pigeons are characterised by an internal array of continuous-valued coordinates of size DIM. These coordinates are mapped to binary coordinates via a cut-off threshold called TAU. This is a mechanism similar to the genotype-to-phenotype mapping widely used in evolutionary algorithms (implemented in the tool as the GPM code block). Two types of pigeon breeds were defined, namely followers and walkers. The initial population is created with an amount of pigeons given by the parameter POP-SIZE, with the subset of walkers assigned randomly in proportion to the parameter WALKERS-RATE; the remainder pigeons are assigned to the subset of followers.

At each step of the simulation, the tool performs four simple actions: find the leader, move the followers, move the walkers and update the best solution found so far. These actions correspond to the following code routines implemented in the software:

UPDATE-LEADER: chooses as leader pigeon the one having the best fitness and updates the best fitness ever if necessary. Fitness is computed with the aforementioned cost functions depending on the chosen problem.

FOLLOW-MOVE: moves each follower towards the leader in one randomly chosen dimension, with a step-size ALPHA plus a random shift in its orientation due to wind or collisions.

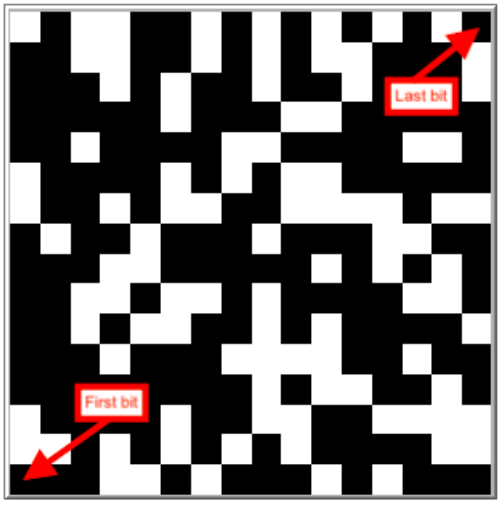
WALK-MOVE: moves each walker around randomly with a step-size SIGMA, again in one randomly chosen dimension). The last two movement rules correspond to the exploration/exploitation mechanisms of the optimisation algorithm.

The simulation is terminated either after a maximum number of steps, MAX-STEPS, or when the truth optimal solution is found prematurely. Besides, the software was implemented using the special–purpose ABM developing platform NetLogo 6.1.0 (see [2]).

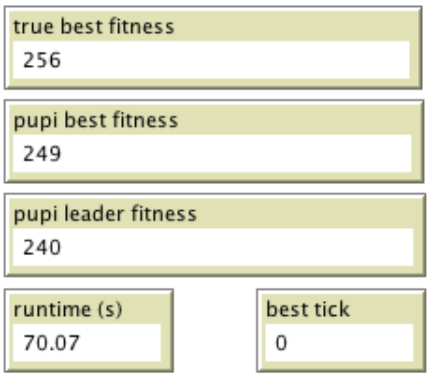
1.3 How to use it

Firstly choose an optimisation PROBLEM to be solved, from the pull-down list. For any of these problems, then define a particular dimension DIM. Additionally, choose the algorithm parameters POP-SIZE, WALKERS-RATE, TAU, ALPHA and SIGMA. You can also set the termination criterium MAX-TICKS. Then press SETUP, then GO.

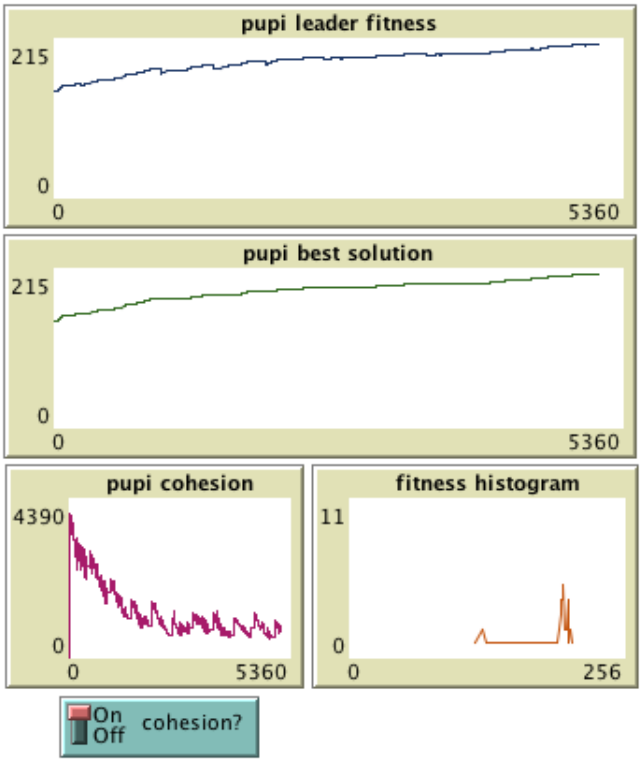
The initial genotypes of the population of pigeons will be assigned randomly and their phenotypes would be obtained. Afterwards, at each time step pigeons move according to its role, the population fitness is updated, and if needed, the leader is re-assigned. The simulation view area will display the phenotype, or bitstring, of the best pigeon found so far. This bitstring is reshaped and displayed as a 2D grid of binary cells (black cells correspond to bits set to 1 in the bitsring, whereas white cells to bits set to 0). The first bit in the bitstring is located in the bottom-left corner of the grid, and the last bit in the upper-right corner, as it is illustrated below:



The output monitors show the fitness (cost) of the true solution for the problem, the best fitness ever found by the algorithm during the simulation, and the fitness associated to the current leader. If the algorithm is able to find the true solution, the simulation stops and the BEST-TICK and RUNTIME monitors will display a "!!!" sign inserted behind their actual values. Otherwise the simulation finishes when MAX-TICKS are elapsed.



Lastly, the tool also outputs the plot of the leader fitness vs time, the plot of fitness of the best solution found vs time, the histogram of fitness of the population and the plot of flock cohesion vs time if the COHESION? switch is enabled. The latter implies an additional cost to the running time, as the software needs to compute distances between all the pigeons in the follower's flock.



1.4 Other distinctive features

Although the actual locations of the pigeons can not be visualised (because the problems have high-dimensional search spaces), one can notice that the leader and the flock move out from one local minima to another. This occurs because every certain number of ticks, the entire population become walkers wandering about other regions of the search space. This phenomenon is explained by the fitness variation of the leader pigeon during the simulation timeline, as it can be seen in the corresponding plot. Nonetheless, the best found ever solution always has an increasing fitness as it can be verified in its respective plot. Besides, the flock formation behaviour is suggested by the periodic patterns that appear in the cohesion plot.

Lastly, we remark that the pattern of solutions showed in the view area resemble a blacked display for the "oneMax" and "powSum" problems, and a half-and-half black and white display for the "squareWave" problem, as it was expected from their mathematical definitions. Notice that for the "powSum" problem, the last bits to be set are those in the bottom of the view (the least significant loci of the bitstring), as they contribute the least to its cost function.

1.5 Try it yourself

Experiment with different DIM sizes for each problem and compare if a different configuration of parameters is needed. For starters, we suggest the following settings:

DIM = 100

POP-SIZE = 40,

MAX-TICKS = 40000,

TAU = 0.5

WALKERS-RATE = 0.2,

ALPHA = 0.9,

SIGMA = 0.1,

1.6 Extending the tool

An interesting question arising is if the convergence speed of the algorithm can be improved without compromising its simplicity for practical purposes, for example using dynamic updates of the step sizes of pigeon movements. Another related idea worth of exploring is if convergence speedup an be achieved by hybridising the model with local search techniques.

Finally, it would be appealing to include additional binary problems and try to solve them with this tool. For this purpose, users only need to add a code block that computes the cost function of the new problem, based on the bitstring phenotype of the pigeons, in addition to the TRUE-BEST-FITNESS value in the SETUP-PROBLEM procedure. We suggest for example, studying combinatorial problems such as the N-queens or the Knapsack problem.

1.7 References

[1] Rojas-Galeano, S. (2019). Binary optimisation with an urban pigeon-inspired swarm algorithm. In: Workshop on Engineering Applications. Springer CCIS Series. https://link.springer.com/chapter/10.1007/978-3-030-31019-6_17.

[2] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Chapter 2

Installation and execution

2.1 Online version

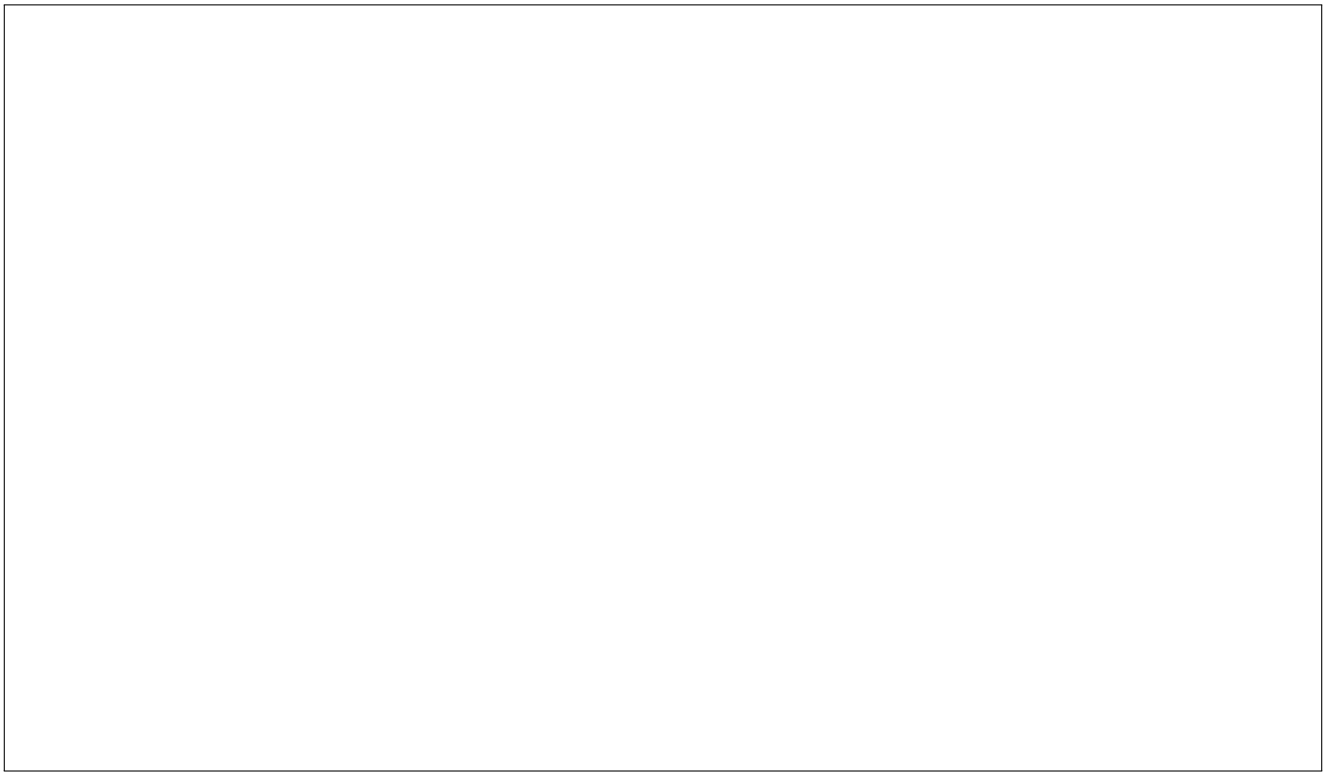
The easiest way of experimenting with UPIBinary is by using its online version. The software is available at the ModellingCommons website. So, you just need to follow these steps:

1. Open your favourite Internet browser and point it to the following URL: http://modelingcommons.org/browse/one_model/6400
2. The following web page should appear:

3. From the toolbar, choose the “Run in Netlogo Web” tab:

4. A grey area in the middle of the screen is shown. Do “Click to Run Model”:

5. The model main screen will show up:



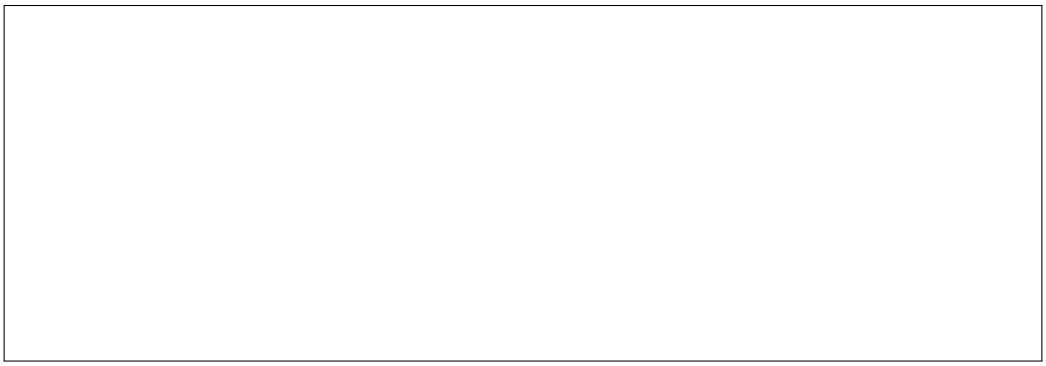
6. That's all! Choose the running parameters in the control panel, click SETUP and then GO! The ock of pigeons will start the search in the genotype space in order to solve the chosen problem, while the view area will display the phenotype of the best solution found ever and the monitors and plots will show the performance indicators of the adaptation process to the corresponding binary cost function.

2.2 Desktop version

The desktop version is recommended if you want to try heavy experimentation, such as parameter tuning, average behaviour of multiple runs or simulations with large populations. For this purpose, PUPI Binary runs over the NetLogo desktop simulation platform [2]. In this case, you need to go through the following steps:

1. Download and install the NetLogo desktop software. For this purpose, go to http://ccl.northwestern.edu/netlogo/ , click in "Download NetLogo" and follow the instructions:

2. Download the PUPI Binary model file from the tool webpage, using the "Export" button:



A file called Binary Optimisation with Urban Pigeon-inspired Model.nlogo would be downloaded to your local disk.

3. Run NetLogo on your computer. Choose the menu option File! Open

Locate the PUPI Binary .nlogo file that you downloaded previously and open it.