PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

BINARY GLOBAL-BEST HARMONY SEARCH ALGORITHM FOR SOLVING SET-COVERING PROBLEM

JUAN AGUSTÍN SALAS FERNÁNDEZ

THESIS TO APPLY FOR THE MASTER'S DEGREE
IN INFORMATIC ENGINEERING

JULY, 2016

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

BINARY GLOBAL-BEST HARMONY SEARCH ALGORITHM FOR SOLVING SET-COVERING PROBLEM

JUAN AGUSTÍN SALAS FERNÁNDEZ

Dr. Broderick Crawford Labrín
Master Thesis Advisor

PhD. Ricardo Soto de Giorgis
Master Thesis Co-Advisor

THESIS TO APPLY FOR THE MASTER'S DEGREE
IN INFORMATIC ENGINEERING

JULY, 2016

Abstract

In this thesis, we propose a Binary Global-Best Harmony Search (BGBHS) Algorithm to solve different instances of the Set Covering Problem (SCP). The SCP is considered a classic combinatorial optimization problem, belonging to the class NP-hard problem and have many practical applications. In this paper we consider applying BGBHS and Modified BGBHS supported in adaptive adjustment of probability parameter p in the Bernoulli trials when the harmonies are created. The different results presented in this paper show that our algorithm is a good alternative at a low cost to solve the SCP.

Keywords: Binary Global-Best Harmony Search, Metaheuristic, Set Covering Problem.

Contents

1 Introduction **1**
- 1.1 Metaheuristics . . . 1
  - 1.1.1 Classical Optimization Models . . . 1
- 1.2 Main goal . . . 1
- 1.3 Specific objectives . . . 1

2 Conceptual Framework **3**
- 2.1 Metaheuristics . . . 3
  - 2.1.1 Definition . . . 3
  - 2.1.2 Clasification . . . 3
- 2.2 Nature-inspired vs. non-nature inspiration . . . 4
  - 2.2.1 Operator . . . 4
  - 2.2.2 Solution . . . 4
  - 2.2.3 Constrain . . . 4
  - 2.2.4 Benchmark . . . 4
  - 2.2.5 Objective Function . . . 4
  - 2.2.6 Fitness . . . 4
  - 2.2.7 Matrix of Costs . . . 4
  - 2.2.8 Optimal Value . . . 4
  - 2.2.9 Domain . . . 5
  - 2.2.10 Matrix A . . . 5
  - 2.2.11 RPD . . . 5
  - 2.2.12 Harmony Memory . . . 5
  - 2.2.13 Harmony Memory Size . . . 5
  - 2.2.14 Harmony Memory Consideration Rate (HMCR) . . . 5
  - 2.2.15 Pitch Adjusting Rate (PAR) . . . 5

3 Theoretical Framework **6**
- 3.1 . . . 6
- 3.2 Set Covering Problem . . . 6
  - 3.2.1 SCP formulation . . . 6
  - 3.2.2 SCP sample solution . . . 8
  - 3.2.3 Harmony Search Metaheuristic . . . 10
  - 3.2.4 HS operation in depth . . . 10
  - 3.2.5 Global-Best Harmony Search Metaheuristic . . . 12
  - 3.2.6 Binary Global-Best Harmony Search Metaheuristic . . . 12

4 Methodological Framework **14**
4.1 Binary Global-Best Harmony Search Algorithm 14

5 Results **15**

6 Conclusion **16**

7 Appendix **17**
7.1 Instance 4.1 17
7.2 Instance 4.2 18
7.3 Instance 4.3 19
7.4 Instance 4.4 20
7.5 Instance 4.5 21
7.6 Instance 4.6 22
7.7 Instance 4.7 23
7.8 Instance 4.8 24
7.9 Instance 4.9 25
7.10 Instance 4.10 26
7.11 Instance 5.1 27
7.12 Instance 5.2 28
7.13 Instance 5.3 29
7.14 Instance 5.4 30
7.15 Instance 5.5 31
7.16 Instance 5.6 32
7.17 Instance 5.7 33
7.18 Instance 5.8 34
7.19 Instance 5.9 35
7.20 Instance 5.10 36
7.21 Instance 6.1 37
7.22 Instance 6.2 38
7.23 Instance 6.3 39
7.24 Instance 6.4 40
7.25 Instance 6.5 41
7.26 Instance A.1 42
7.27 Instance A.2 43
7.28 Instance A.3 44
7.29 Instance A.4 45
7.30 Instance A.5 46
7.31 Instance B.1 47
7.32 Instance B.2 48

7.33 Instance B.3 49
7.34 Instance B.4 50
7.35 Instance B.5 51
7.36 Instance C.1 52
7.37 Instance C.2 53
7.38 Instance C.3 54
7.39 Instance C.4 55
7.40 Instance C.5 56
7.41 Instance D.1 57
7.42 Instance D.2 58
7.43 Instance D.3 59
7.44 Instance D.4 60
7.45 Instance D.5 61

List of Figures

1 Guidelines for solving SCP. 7
2 Set Covering Problem example. 8
3 Set Covering Problem solution. 9

List of Tables

1 HS - Musical Notes . . . 11
2 HS components - Musician . . . 11
3 HS components - Pitch range . . . 12
4 HS components - Solution . . . 12
5 HS components - Aesthetics audience . . . 12
6 Instance 4.1 . . . 17
7 Instance 4.2 . . . 18
8 Instance 4.3 . . . 19
9 Instance 4.4 . . . 20
10 Instance 4.5 . . . 21
11 Instance 4.6 . . . 22
12 Instance 4.7 . . . 23
13 Instance 4.8 . . . 24
14 Instance 4.9 . . . 25
15 Instance 4.10 . . . 26
16 Instance 5.1 . . . 27
17 Instance 5.2 . . . 28
18 Instance 5.3 . . . 29
19 Instance 5.4 . . . 30
20 Instance 5.5 . . . 31
21 Instance 5.6 . . . 32
22 Instance 5.7 . . . 33
23 Instance 5.8 . . . 34
24 Instance 5.9 . . . 35
25 Instance 5.10 . . . 36
26 Instance 6.1 . . . 37
27 Instance 6.2 . . . 38
28 Instance 6.3 . . . 39
29 Instance 6.4 . . . 40
30 Instance 6.5 . . . 41
31 Instance A.1 . . . 42
32 Instance A.2 . . . 43
33 Instance A.3 . . . 44
34 Instance A.3 . . . 45
35 Instance A.5 . . . 46
36 Instance B.1 . . . 47
37 Instance B.2 . . . 48
38 Instance B.3 . . . 49

39 Instance B.4 50
40 Instance B.5 51
41 Instance C.1 52
42 Instance C.2 53
43 Instance C.3 54
44 Instance C.4 55
45 Instance C.5 56
46 Instance D.1 57
47 Instance D.2 58
48 Instance D.3 59
49 Instance D.4 60
50 Instance D.5 61

1 Introduction

1.1 Metaheuristics

Every process has a potential to be optimized. the vast majority of companies is involved in solving optimization problems. Indeed, many challenging applications in science and industry can be formulated as optimization problems. Metaheuristics are a branch of optimization in computer science and applied mathematics that are related to algorithms and computational complexity theory. metaheuristics are raising a large interest in diverse technologies, industries, and services since they proved to be efficient algorithms in solving a wide range of complex real-life optimization problems in different domains.

1.1.1 Classical Optimization Models

Optimization problems are encountered in many domains: BigData, engineering, logistics, and business. An optimization problem may be defined by the couple (S, f), where S represents the set of feasible solutions, and $f : S \implies \mathbb{R}$ the objective function to optimize. The objective function assigns to every solution $s \in S$ of the search space a real number indicating its worth. The objective function f allows to define a total order relation between any pair of solutions in the search space.

The global optimum is defined as a solution $s^* \in S$ and it has a better objective function than all solutions of the search space, that is, $\forall s \in S$, $f(s^*) \leq f(s)$.

The main goal in solving an optimization problem is to find a global optimal solution s^*.

1.2 Main goal

Solving the SCP using the Binary Global-Best HS metaheuristic algorithm.

1.3 Specific objectives

- Solve the SCP with the BGBHS metaheuristic and compare the results with benchmark of Beasley [4]
- Introduce new operators that make the results more close to the global optimum.

- Search nearest optimal solutions to problems in the SCP.
- Compare and analyze the results using different resolution strategies.

2 Conceptual Framework

2.1 Metaheuristics

2.1.1 Definition

The *meta* and *heuristic* are Greek words, *meta* it means *higherlevel* and *heuristics* means *to find*, *to know* or *to discover*. Metaheuristics are a set of intelligent strategies to enhance the efficiency of heuristic procedures.

A metaheuristic will be successful on a given optimizaction problem if it can provide a balance between exploration and exploitation. Exploration means to generate diverse solutions so as to explore the search space on the global scale, while exploitation means search in a local region by exploiting the information that a current good solution is found in this region. exploration, often by use of randomization, which enables an algorithm to have the ability to jump out of any local optimum so as to explore the search globally.

2.1.2 Clasification

The different metaheuristic approaches can be characterized by different aspects concerning the search path they follow or how memory is exploited [7].

2.1.2.1 Trajectory methods vs. discontinuous methods

The different between this metaheuristics is whether they follow one single search trajectory corresponding to a closed walk on the neighborhood graph or whether larger jumps in the neighborhood graph are allowed. Examples of this clasificaction are: tabu search and simulated annealing. These methods usually allow moves to worse solutions to be able to escape from local minima.

The generation of starting solutions corresponds to jumps in the search space; these algorithms, in general, follow a discontinuous walk with respect to the neighborhood graph used in the local search.

2.1.2.2 Population-based vs. single-point search

2.1.2.3 Memory usage vs. memoryless methods

2.1.2.4 One vs. various neighborhood structures

2.1.2.5 Dynamic vs. static objective function

2.2 Nature-inspired vs. non-nature inspiration

2.2.1 Operator

Unitary procedure for transforming information or implement the behavior of the algrithm.

2.2.2 Solution

Array of n columns containing a solution for a given problem. In the binary case, the posible values are 0 and 1.

2.2.3 Constrain

Conditions to be met to find a viable solution.

2.2.4 Benchmark

Optimal set of known problem instances to validate the propose algorithm.

2.2.5 Objective Function

Implements the mathematical expression representing the problem to solve. The guiding objective function is related to the goal to achieve.

2.2.6 Fitness

Value resulting by applying the objective function to solution.

2.2.7 Matrix of Costs

Consist of n columns vector containing the cost associated with each problem variable.

2.2.8 Optimal Value

Solution with the best fitness.

2.2.9 Domain

Set of possible values for the variable.

2.2.10 Matrix A

Matrix containing the restrictions for the given problem.

2.2.11 RPD

Relative Percentage Deviation.

2.2.12 Harmony Memory

Memory space which includes the population of the solution vectors.

2.2.13 Harmony Memory Size

Defines the amount of harmonies that can be stored in HM.

2.2.14 Harmony Memory Consideration Rate (HMCR)

In memory consideration, the value of decision variable x'_1 is randomly selected from the historical values, other decision variables, $(x'_2, x'_3, \ldots, x'_N)$ are sequentially selected in the same manner with probability where HMCR $\in$ (0,1).

2.2.15 Pitch Adjusting Rate (PAR)

Each decision variable x'_i assigned a value by memory considerations is pitch adjusted with the probability of PAR where PAR $\in$ (0,1).

3 Theoretical Framework

3.1

3.2 Set Covering Problem

3.2.1 SCP formulation

The Set Covering Problem (SCP) is a well-known mathematical problem, which tries to cover a set of needs at the lowest possible cost. The SCP was included in the list of 21 $\mathcal{NP}$-*complet* problems of Karp [19]. There are many practical uses for this problem, such as: crew scheduling [1,3], location of emergency facilities [23, 27], production planning in industry [24, 25, 26], vehicle routing [2, 13], ship scheduling [5, 12], network attack or defense [6], assembly line balancing [14,22], traffic assignment in satellite communication systems [9,21], simplifying boolean expressions [8], the calculation of bounds in integer programs [10], information retrieval [11], political districting [15], crew scheduling problems in airlines [18], among others. The SCP can be formulated as follows:

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

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 $j \in S$.

In the following section, we present a simple way to understand the SCP, through an example:

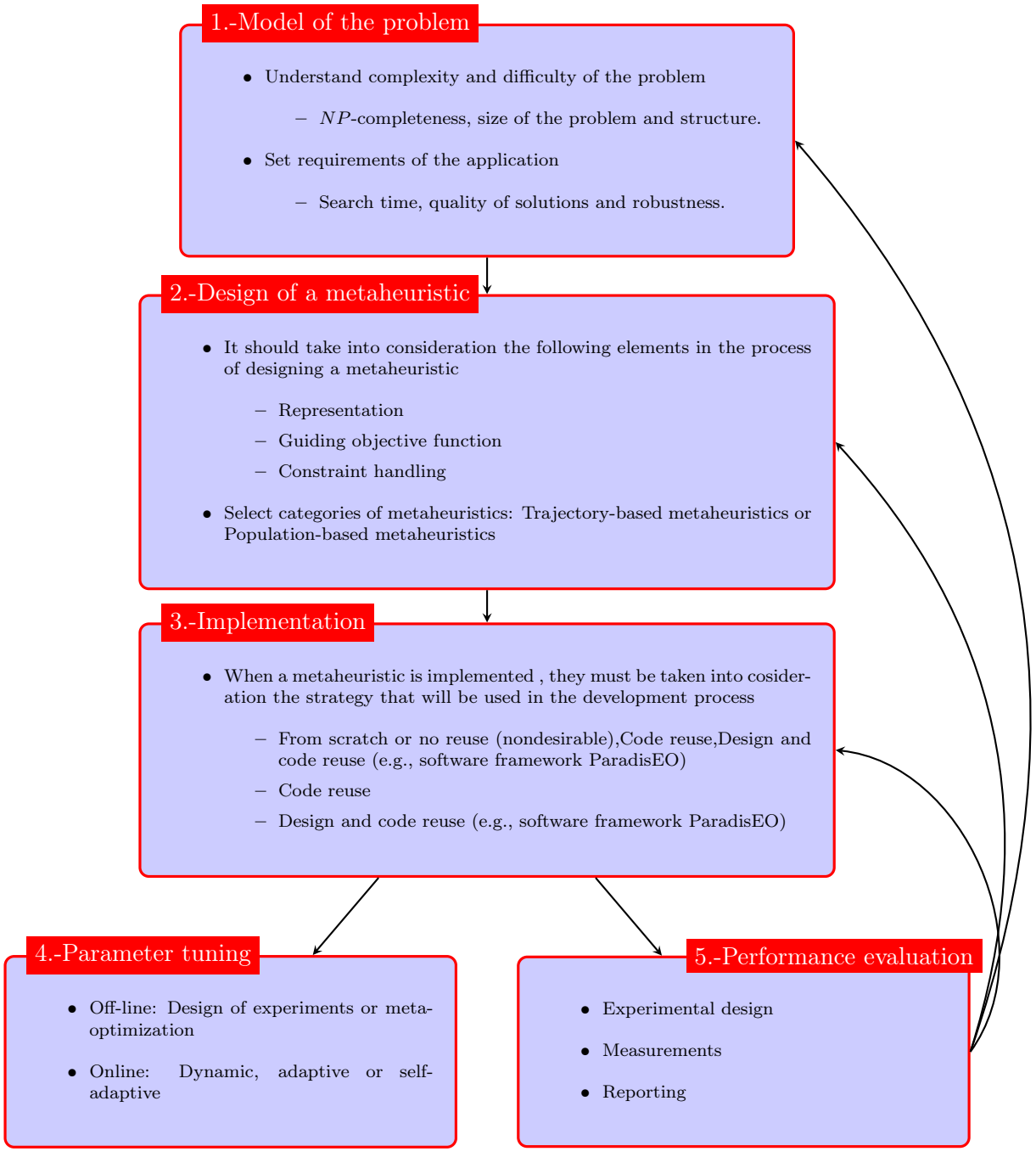

Figure 1: Guidelines for solving SCP.

3.2.2 SCP sample solution

Imagine that an ambulance station can meet the needs of an geographic zone. Similarly the ambulance station can cover all the needs of the nearby areas. For example, if a station is built in Zone 1 (Figure 2) ambulance station can meet the needs of neighboring areas, that is, it could also cover: Zone 1, Zone 2, Zone 3 and Zone 4. This can be appreciated in equation (5).

In this example, we must fulfill the need to cover the geographical areas defined in accordance with the restrictions.

The restriction of this case is that all areas must be covered by at least one ambulance station and the goal is to minimize the number of stations built, the cost of building a station is the same for all areas. The x_j variable represents the area j which is 1 if the ambulance station is built, and will be 0 if not. As above, it can be formulated as follows:

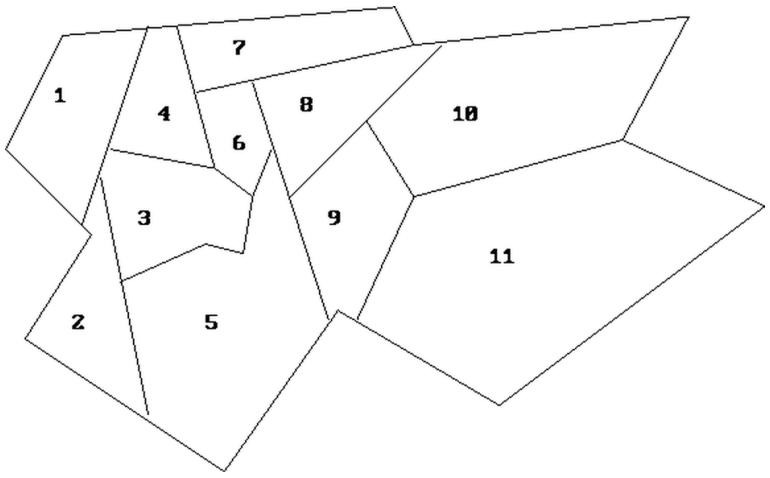

Figure 2: Set Covering Problem example.

$$\text{Min} \quad c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + c_{11}x_{11} \qquad (4)$$

Subject to:

$$
\begin{array}{llllllllllll}
x_1+ & x_2+ & x_3+ & x_4 & & & & & & & \geq 1 & (5)\\
x_1+ & x_2+ & x_3 & + & x_5 & & & & & & \geq 1 & (6)\\
x_1+ & x_2+ & x_3+ & x_4+ & x_5+ & x_6 & & & & & \geq 1 & (7)\\
x_1 & + & x_3+ & x_4 & + & x_6+ & x_7 & & & & \geq 1 & (8)\\
 & x_2+ & x_3 & + & x_5+ & x_6 & + & x_8+ & x_9 & & \geq 1 & (9)\\
 & & x_3+ & x_4+ & x_5+ & x_6+ & x_7+ & x_8 & & & \geq 1 & (10)\\
 & & & x_4 & + & x_6+ & x_7+ & x_8 & & & \geq 1 & (11)\\
 & & & & x_5+ & x_6+ & x_7+ & x_8+ & x_9+ & x_{10} & \geq 1 & (12)\\
 & & & & x_5 & & + & x_8+ & x_9+ & x_{10}+ & x_{11} \geq 1 & (13)\\
 & & & & & & & x_8+ & x_9+ & x_{10}+ & x_{11} \geq 1 & (14)\\
 & & & & & & & & x_9+ & x_{10}+ & x_{11} \geq 1 & (15)
\end{array}
$$

The first constraint (5) indicates that to cover zone 1, it is possible to locate a station in the same area or in the border. The following restriction is for zone 2 and so on. One possible optimal solution for this problem is to locate ambulance stations in zones 3, 8 and 9. That is, $x_3 = x_8 = x_9 = 1$ y $x_1 = x_2 = x_4 = x_5 = x_6 = x_7 = x_{10} = x_{11} = 0$. As shown in (Figure 3).

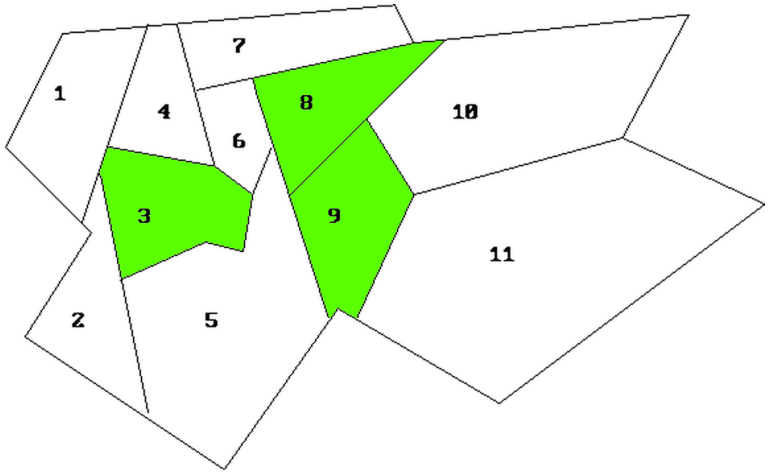

Figure 3: Set Covering Problem solution.

We propose solve the SCP, with a variation metaheuristic Harmony Search (HS) called Binary Global-Best Harmony Search to obtain satisfactory solutions within a reasonable time. HS mimics the process of musical improvisation, where musicians make adjustments in tone to achieve aesthetic harmony.

3.2.3 Harmony Search Metaheuristic

Harmony Search (HS) is a population-based metaheuristic algorithm inspired from the musical process of searching for a perfect state of Harmony or Aesthetic Quality of a Harmony (AQH). The HS was proposed by Z. W. Geem et al. [17].

In other words, the main idea of the HS metaheuristic is to mimic the process performed by musicians when they try to play a beautiful harmony.

For the purpose of properly understand what is looking like a good solution in this metaheuristic, we must know the meaning of AQH, this concept and others are defined below.

The AQH in an instrument it is essentially determined by its pitch (or frequency), sound quality, and amplitude (or loudness). The sound quality is mainly determined by the harmonic content that is in turn determined by the waveforms or modulations of the sound signal. However, the harmonics that it can generate will largely depend on the pitch or frequency range of the particular instrument [16].

Different notes have different frequencies. For example, the note A has a fundamental frequency of $f_0 = 440Hz$. The fundamental frequency of each note can be seen in (Table 1).

Given the above, it is established that a good harmony has good AQH.

The pitch of each musical instrument determines the AQH, just as the fitness function values determines the quality of the decision variables.

In the music improvisation process, all musicians sound pitches within possible range together to make one harmony.

If all pitches make a good harmony, each musician stores in his memory that experience and possibility of making a good harmony in increased next time. The same thing in optimization, the initial solution is generated randomly from decision variables within the possible range.

If the objetive function values of these decision variables is good to make promising solution, then the possibility to make a good solution is increased next time.

In this document, we focus on studying the classical SCP problem (equation 1), where we want minimize the cost.

3.2.4 HS operation in depth

In general, the procedure for HS metaheuristic, consists of the following four steps.

Musical Note	**Frequency**
C	261,625565 Hz
D	293,664768 Hz
E	329,627557 Hz
F	349,228231 Hz
G	391,995436 Hz
A	440,000000 Hz
B	493,883301 Hz

Tabla 1: HS - Musical Notes

	Each musician represents a decision variable, according to the example shown, there would be 11 musicians, since there are 11 decision variables $x_1 \dots x_{11}$.

Tabla 2: HS components - Musician

Step 1: Initialization parameters
The parameters required to solve the optimization problem are speci?ed in this step, an initial HM is filled with a population of Harmony Memory Size (HMS), harmonies are generated randomly.
In addition, the parameters of HS, that is, Harmony Memory Consideration Rate (HMCR) which determines the rate of selecting the value from the memory and Pitch Adjusting Rate (PAR) determines the probability of local improvement and number of improvisations (NI) are given when the metaheuristic begins.

Step 2: New Harmony
Improvise a new harmony from the current HM.

Step 3: Replace worst Harmony in HM
If the new generated harmony is better than the worst one in HM, then replace the worst harmony with the new one; otherwise, go to the next step.

Step 4: Check the stop criteria
If a stopping criterion is not satisfied, go to Step 2.

	The pitch range of the instrument represents the range of values that can take a decision variable. Given the nature of the problem, the possible values are $\{0, 1\}$

Tabla 3: HS components - Pitch range

	Musical harmony at a certain time, corresponds to a solution at a certain iteration.

Tabla 4: HS components - Solution

3.2.5 Global-Best Harmony Search Metaheuristic

To further improve the convergence performance of HS and overcome some shortcomings of HS, a new variant of HS, called Global-Best Harmony Search (GHS), was proposed by Omran and Mahdavi [20]. First, the GHS dynamically updates parameter PAR according to equation (16):

$$PAR(t) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI} t \tag{16}$$

where $PAR(t)$ represents the pitch adjusting rate at generation t, PAR_{min} and PAR_{max} are the minimum and maximum adjusting rate, respectively. The parameter t is the iterative variable, and parameter NI is the number of improvisations.

3.2.6 Binary Global-Best Harmony Search Metaheuristic

The HS is good at identifying the high performance regions of the solution space in a reasonable time, but poor at performing local search [28]. Namely, there is imbalance between the exploration and the exploitation of HS. Furthermore, HS designed for continuous space cannot be directly used to solve

	Aesthetics audience, judges whether harmony is good or not. In the problem it refers to the objective function.

Tabla 5: HS components - Aesthetics audience

discrete combinatorial optimization problems.

In order to overcome the drawbacks of HS, a novel binary global-best harmony search (BGHS) is designed for binary optimization problems.

Owing to better performance of GHS, some modifications to GHS are introduced to further enhance the convergence performance of GHS. Then a novel binary coded GHS, a two-phase repair operator, and a greedy selection mechanism are integrated into the BGHS. And they are described in detail as follows.

. . . .

4 Methodological Framework

4.1 Binary Global-Best Harmony Search Algorithm

5 Results

Resultados

6 Conclusion

Conclusión

7 Appendix

7.1 Instance 4.1

Method	Optimum	Min	Max	Avg	RPD
BGBHS	429	534			24.475524475524477
BGBHS	429	553			28.904428904428904
BGBHS	429	533			24.242424242424242
BGBHS	429	600			39.86013986013986
BGBHS	429	438			2.0979020979020979
BGBHS	429	460			7.2261072261072261
BGBHS	429	455			6.0606060606060606
BGBHS	429	470			9.5571095571095572
BGBHS	429	533			24.242424242424242
BGBHS	429	532			24.009324009324008
BGBHS IMPROVED	429	451			5.1282051282051286
BGBHS IMPROVED	429	440			2.5641025641025643
BGBHS IMPROVED	429	460			7.2261072261072261
BGBHS IMPROVED	429	445			3.7296037296037294
BGBHS IMPROVED	429	470			9.5571095571095572
BGBHS IMPROVED	429	430			0.23310023310023309
BGBHS IMPROVED	429	430			0.23310023310023309
BGBHS IMPROVED	429	432			0.69930069930069927
BGBHS IMPROVED	429	435			1.3986013986013985
BGBHS IMPROVED	429	430			0.23310023310023309

Tabla 6: Instance 4.1

7.2 Instance 4.2

Method	Optimum	Min	Max	Avg	RPD
BGBHS	512	632			23.4375
BGBHS	512	662			29.296875
BGBHS	512	660			28.90625
BGBHS	512	602			17.578125
BGBHS	512	641			25.1953125
BGBHS	512	644			25.78125
BGBHS	512	642			25.390625
BGBHS	512	647			26.3671875
BGBHS	512	641			25.1953125
BGBHS	512	623			21.6796875
BGBHS IMPROVED	512	550			7.421875
BGBHS IMPROVED	512	560			9.375
BGBHS IMPROVED	512	563			9.9609375
BGBHS IMPROVED	512	520			1.5625
BGBHS IMPROVED	512	523			2.1484375
BGBHS IMPROVED	512	555			8.3984375
BGBHS IMPROVED	512	524			2.34375
BGBHS IMPROVED	512	518			1.171875
BGBHS IMPROVED	512	519			1.3671875
BGBHS IMPROVED	512	520			1.5625

Tabla 7: Instance 4.2

7.3 Instance 4.3

Method	Optimum	Min	Max	Avg	RPD
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS	516				-100
BGBHS IMPROVED	516	613			18.7984496124031
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100
BGBHS IMPROVED	516				-100

Tabla 8: Instance 4.3

7.4 Instance 4.4

Method	Optimum	Min	Max	Avg	RPD
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100
BGBHS IMPROVED	494				-100

Tabla 9: Instance 4.4

7.5 Instance 4.5

Method	Optimum	Min	Max	Avg	RPD
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100
BGBHS IMPROVED	512				-100

Tabla 10: Instance 4.5

7.6 Instance 4.6

Method	Optimum	Min	Max	Avg	RPD
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100
BGBHS IMPROVED	560				-100

Tabla 11: Instance 4.6

7.7 Instance 4.7

Method	Optimum	Min	Max	Avg	RPD
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100
BGBHS IMPROVED	430				-100

Tabla 12: Instance 4.7

7.8 Instance 4.8

Method	Optimum	Min	Max	Avg	RPD
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100
BGBHS IMPROVED	492				-100

Tabla 13: Instance 4.8

7.9 Instance 4.9

Method	Optimum	Min	Max	Avg	RPD
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100
BGBHS IMPROVED	641				-100

Tabla 14: Instance 4.9

7.10 Instance 4.10

Tabla 15: Instance 4.10

7.11 Instance 5.1

Tabla 16: Instance 5.1

7.12 Instance 5.2

Tabla 17: Instance 5.2

7.13 Instance 5.3

Tabla 18: Instance 5.3

7.14 Instance 5.4

Tabla 19: Instance 5.4

7.15 Instance 5.5

Tabla 20: Instance 5.5

7.16 Instance 5.6

Tabla 21: Instance 5.6

7.17 Instance 5.7

Tabla 22: Instance 5.7

7.18 Instance 5.8

Tabla 23: Instance 5.8

7.19 Instance 5.9

Tabla 24: Instance 5.9

7.20 Instance 5.10

Tabla 25: Instance 5.10

7.21 Instance 6.1

Tabla 26: Instance 6.1

7.22 Instance 6.2

Tabla 27: Instance 6.2

7.23 Instance 6.3

Tabla 28: Instance 6.3

7.24 Instance 6.4

Tabla 29: Instance 6.4

7.25 Instance 6.5

Tabla 30: Instance 6.5

7.26 Instance A.1

Tabla 31: Instance A.1

7.27 Instance A.2

Tabla 32: Instance A.2

7.28 Instance A.3

Tabla 33: Instance A.3

7.29 Instance A.4

Tabla 34: Instance A.3

7.30 Instance A.5

Tabla 35: Instance A.5

7.31 Instance B.1

Tabla 36: Instance B.1

7.32 Instance B.2

Tabla 37: Instance B.2

7.33 Instance B.3

Tabla 38: Instance B.3

7.34 Instance B.4

Tabla 39: Instance B.4

7.35 Instance B.5

Tabla 40: Instance B.5

7.36 Instance C.1

Tabla 41: Instance C.1

7.37 Instance C.2

Tabla 42: Instance C.2

7.38 Instance C.3

Tabla 43: Instance C.3

7.39 Instance C.4

Tabla 44: Instance C.4

7.40 Instance C.5

Tabla 45: Instance C.5

7.41 Instance D.1

Tabla 46: Instance D.1

7.42 Instance D.2

Tabla 47: Instance D.2

7.43 Instance D.3

Tabla 48: Instance D.3

7.44 Instance D.4

Tabla 49: Instance D.4

7.45 Instance D.5

Tabla 50: Instance D.5

References

[1] A. I. Ali and H. Thiagarajan. A network relaxation based enumeration algorithm for set partitioning. *European Journal of Operational Research*, 38(1):76–85, 1989.

[2] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[3] J. J. Bartholdi. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operations Research*, 29(3):501–510, 1981.

[4] J. E. Beasley. OR-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[5] M. Bellmore, H. J. Geenberg, and J. J. Jarvis. Multi-commodity disconnecting sets. *Management Science*, 16(6):B427–B433, 1970.

[6] M. Bellmore and H. D. Ratliff. Optimal defense of multi-commodity networks. *Management Science*, 18(4-part-i):B174–B185, 1971.

[7] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp. Classification of Metaheuristics and Design of Experiments for the Analysis of Components.

[8] M. A. Breuer. Simplification of the covering problem with application to boolean expressions. *Journal of the Association for Computing Machinery*, 17(1):166–181, Jan. 1970.

[9] S. Ceria, P. Nobili, and A. Sassano. A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81(2):215–228, 1998.

[10] N. Christofides. Zero-one programming using non-binary tree-search. *Computer Journal*, 14(4):418–421, 1971.

[11] R. H. Day. Letter to the editor—on optimal extracting from a multiple file data storage system: An application of integer programming. *Operations Research*, 13(3):482–494, 1965.

[12] M. L. Fisher and M. B. Rosenwein. An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistics*, 36(1):27–42, 1989.

[13] B. A. Foster and D. M. Ryan. An integer programming approach to the vehicle scheduling problem. *Operations Research*, 27:367–384, 1976.

[14] B. A. Freeman and J. V. Jucker. The line balancing problem. *Journal of Industrial Engineering*, 18:361–364, 1967.

[15] R. S. Garfinkel and G. L. Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B495–B508, 1970.

[16] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[17] Z. W. Geem, J. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.

[18] E. Housos and T. Elmroth. Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5):68–77, 1997.

[19] R. M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 219–241. 2010.

[20] M. G. H. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[21] C. C. Ribeiro, M. Minoux, and M. C. Penna. An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment. *European Journal of Operational Research*, 41(2):232–239, 1989.

[22] M. E. Salveson. The assembly line balancing problem. *Journal of Industrial Engineering*, 6:18–25, 1955.

[23] F. J. Vasko and G. R. Wilson. Using a facility location algorithm to solve large set covering problems. *Operations Research Letters*, 3(2):85–90, 1984.

[24] F. J. Vasko, F. E. Wolf, and K. L. Stott. Optimal selection of ingot sizes via set covering. *Operations Research*, 35(3):346–353, June 1987.

[25] F. J. Vasko, F. E. Wolf, and K. L. Stott. A set covering approach to metallurgical grade assignment. *European Journal of Operational Research*, 38(1):27–34, 1989.

[26] F. J. Vasko, F. E. Wolf, K. L. Stott, and J. W. Scheirer. Selecting optimal ingot sizes for bethlehem steel. *Interfaces*, 19(1):68–84, 1989.

[27] W. Walker. Using the set-covering problem to assign fire companies to fire houses. *Operations Research*, 22:275–277, 1974.

[28] W. Xiang, M. An, Y. Li, R. He, and J. Zhang. An improved global-best harmony search algorithm for faster optimization. *Expert Syst. Appl.*, 41(13):5788–5803, 2014.