

STATE UNIVERSITY OF MARINGÁ  
TECHNOLOGY CENTER  
PRODUCTION ENGINEERING DEPARTMENT  
POSTGRADUATE PROGRAM IN PRODUCTION ENGINEERING

CAMILA ANDRADE DE MACEDO

**Azure Jay Search (AJS)**

Maringá  
2021

CAMILA ANDRADE DE MACEDO

## **Azure Jay Search (AJS)**

Dissertation presented to the Postgraduate Program in Production Engineering of the Department of Production Engineering, Technology Center of the State University of Maringá, as a partial requirement for obtaining the title of Master's in Production Engineering.  
Area of concentration: Production Management.

Advisor: Prof. Dr. Rodrigo Clemente Thom de Souza

Co-advisor: Profa. Dra. Gislaine Camila Lapasini Leal

Maringá  
2021

Dados Internacionais de Catalogação-na-Publicação (CIP)  
(Biblioteca Central - UEM, Maringá - PR, Brasil)

M141a

Macedo, Camila Andrade de

Azure Jay Search (AJS) / Camila Andrade de Macedo. -- Maringá, PR, 2021.  
85 f.: il. color., figs., tabs.

Orientador: Prof. Dr. Rodrigo Clemente Thom de Souza.

Coorientadora: Profa. Dra. Gislaine Camila Lapasini Leal .

Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia,  
Departamento de Engenharia de Produção, Programa de Pós-Graduação em Engenharia  
de Produção, 2021.

1. Azure Jay Search . 2. Binary Azure Jay Search. 3. modified Crow Search Algorithm .  
4. Continuous optimization problems. 5. Feature Selection. I. Souza, Rodrigo Clemente  
Thom de , orient. II. Leal , Gislaine Camila Lapasini , coorient. III. Universidade Estadual de  
Maringá. Centro de Tecnologia. Departamento de Engenharia de Produção. Programa de  
Pós-Graduação em Engenharia de Produção. IV. Título.

CDD 23.ed. 629.8

# CERTIFICATE OF APPROVAL

CAMILA ANDRADE DE MACEDO

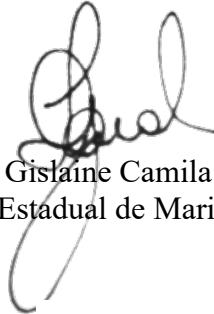
## AZURE JAY SEARCH (AJS)

Dissertation presented to the Postgraduate Program in Production Engineering of the Department of Production Engineering, Technology Center of State University of Maringá, as a partial requirement for obtaining the title of Master in Production Engineering by Examining Board composed of the members:

### EXAMINING BOARD



Prof. Dr. Rodrigo Clemente Thom de Souza  
Universidade Federal do Paraná – Jandaia do Sul/UFPR  
Universidade Estadual de Maringá – PGP/UEM



Profa. Dra. Gislaíne Camila Lapasini Leal  
Universidade Estadual de Maringá – PGP/UEM



Prof. Dr. Ademir Aparecido Constantino  
Universidade Estadual de Maringá – PGP/UEM



Prof. Dr. Leandro dos Santos Coelho  
Pontifícia Universidade Católica do Paraná – PUCPR  
Universidade Federal do Paraná – UFPR

## ACKNOWLEDGMENTS

Infinitely grateful to my parents for their unconditional love and support!

To my boyfriend Vitor, for always believing in my potential and encouraging me in the most difficult moments!

To my advisor, for the countless teachings and for all the time dedicated to me over the years. I emphasize the unconditional support provided, the understanding, patient, interested, enthusiastic and extraordinary way in which he accompanied the realization of this work and all others, from the time of Undergraduate Research. I will be eternally grateful for all the good advice, always with the intention of guiding me to the best way!

To my co-advisor, for accepting me and accepting to be part of this very important period in my life!

To my friend Karol, who with a simple gesture of attention and affection made me see the light in a moment of darkness!

To the other members of the examining board and to all the professors and friends who each contributed in their own way to my education!

I will be eternally grateful!

## Azure Jay Search (AJS)

### *ABSTRACT*

A growing variety of Swarm Intelligence (SI) based algorithms can be found in the literature. This constant increase is due to the good performance of such methods in solving complex optimization problems. Among these problems, continuous to binary search spaces are considered. Generally, these metaheuristics are proposed primarily to deal with continuous variables, however, to deal with binary variables, some binarization technique must be employed. This dissertation has as main objective to propose a new SI based optimization metaheuristic for the Feature Selection (FS) task in a Fault Diagnosis (FD) problem. FD is the process of determining the nature or cause of a failure, analyzing a set or history of information to improve quality, reduce costs, and facilitate preventive maintenance in manufacturing processes. The research structure follows the multipaper model, composed by two papers developed to reach the main objective. The first paper proposes two versions of a new flocking-based modified Crow Search Algorithm (CSA) named Azure Jay Search (AJS) to solve continuous optimization problems. The performance of AJS is experimentally compared with the classical Particle Swarm Optimization (PSO) and some SI techniques also based on the birds behavior, in terms of average accuracy, standard deviation and computational cost when applied to 10 fixed-dimension multimodal and public domain benchmark functions. Despite a relatively higher computational cost when compared to other techniques, both versions of the AJS algorithm performed relatively well in terms of average accuracy. The second paper proposes a binary version of AJS (BAJS) for the FS problem in a wrapper-based model. The proposed BAJS is applied to the FS task precisely in a FD dataset in the Steel Industry. The BAJS performance is evaluated in terms of average training accuracy, standard deviation, computational cost, and number of selected features when compared to other binary SI metaheuristics. Despite the complexity of the dataset and the relatively high computational cost presented by BAJS when compared to other techniques, it achieved relatively good average training accuracy based on  $k$ -fold Cross Validation, and subsets with a relatively low number of features.

**Keywords:** Azure Jay Search, Binary Azure Jay Search, modified Crow Search Algorithm, Continuous optimization problems, Feature Selection.

## Azure Jay Search (AJS)

### RESUMO

Uma variedade crescente de algoritmos baseados em Inteligência de Enxames (IE) pode ser encontrada na literatura. Esse aumento constante se deve ao bom desempenho de tais métodos na resolução de problemas complexos de otimização. Entre esses problemas, pode-se considerar desde espaços de busca contínuos até binários. Geralmente, essas metaheurísticas são propostas primeiramente para lidar com variáveis contínuas. Contudo, para lidar com variáveis binárias, alguma técnica de binarização deve ser empregada. Esta dissertação tem como principal objetivo propor uma nova metaheurística de otimização baseada em IE para a tarefa de Seleção de Atributos (SA) em um conjunto de dados de Diagnóstico de Falhas (DF). DF é o processo de determinar a natureza ou causa de uma falha, analisando um conjunto ou histórico de informações para melhorar a qualidade, reduzir custos e facilitar a manutenção preventiva nos processos de produção. A estrutura da pesquisa segue o modelo *multipaper*, composta por dois artigos desenvolvidos de modo a alcançar o objetivo principal. O primeiro artigo propõe duas versões de um novo *Crow Search Algorithm (CSA)* modificado baseado em *flocking* chamado *Azure Jay Search (AJS)* para resolver problemas de otimização contínua. O desempenho do *AJS* foi comparado experimentalmente com o clássico Particle Swarm Optimization (PSO) e algumas técnicas de IE também baseadas no comportamento de aves, em termos de acurácia média, desvio padrão e custo computacional quando submetidos a 10 funções de domínio público multimodais de dimensão fixa. Apesar de um custo computacional relativamente maior quando comparado a outras técnicas, ambas as versões do algoritmo *AJS* tiveram um desempenho relativamente bom em termos de acurácia média. O segundo artigo propõe uma versão binária do *AJS (BAJS)* para o problema de SA em um modelo baseado em wrapper. O *BAJS* proposto é aplicado para a tarefa de SA justamente em um conjunto de dados de DF na Indústria Siderúrgica. O desempenho do *BAJS* é avaliado em termos de acurácia média de treinamento, desvio padrão, custo computacional e número de atributos selecionados quando comparado a outras metaheurísticas de IE também binárias. Apesar da complexidade do conjunto de dados e do *BAJS* ter apresentado um custo computacional relativamente maior do que as outras técnicas comparadas, ele alcançou uma acurácia média de treinamento relativamente boa (baseada no método de validação *k-fold Cross Validation*) e subconjuntos com um número relativamente baixo de atributos.

**Palavras-chave:** *Azure Jay Search, Binary Azure Jay Search, Crow Search Algorithm* modificado, Problemas de otimização contínua, Seleção de Atributos.



## LIST OF FIGURES

Figure 1 – Types of Research.....	12
Figure 2 – Typical purplish-blue bird (a) and greenish-blue morph.....	20
Figure 3 – AJS Flowchart.....	25
Figure 4 – AJS pseudocode .....	26
Figure 5 – Begging and Compassion.....	27
Figure 6 – Random element added to version 02 (pseudo-code) .....	28
Figure 7 – Begging and compassion (version 02).....	29
Figure 8 – Benchmark functions search space .....	30
Figure 9 – Average training accuracy (30 independent runs) obtained by each technique.....	38
Figure 10 – AJS reduced flowchart .....	45
Figure 11 – 10-fold Cross Validation.....	48
Figure 12 – Point cloud of dataset analyzed after PCA application.....	51
Figure 13 – Selected features.....	53

## LIST OF TABLES

Table 1 – Summary of papers.....	13
Table 2 – Details of fixed-dimension multimodal benchmark functions .....	30
Table 3 – AJS <sub>1</sub> parameter tuning result.....	31
Table 4 – AJS <sub>2</sub> parameter tuning result .....	32
Table 5 – AJS <sub>1</sub> parameter values .....	32
Table 6 – AJS <sub>2</sub> parameter values .....	32
Table 7 – Parameter settings of optimization methods for comparison of the AJS .....	33
Table 8 – Global parameters.....	34
Table 9 – Results in terms of average accuracy, standard deviation, and statistical significance of the fixed-dimension multimodal benchmark functions (AJS <sub>1</sub> ).....	35
Table 10 – Results in terms of average accuracy, standard deviation, and statistical significance of the fixed-dimension multimodal benchmark functions (AJS <sub>2</sub> ) .....	36
Table 11 – Computational cost.....	37
Table 12 – Number of instances for each class .....	48
Table 13 – Parameter settings of optimization methods for comparison of the BAJS.....	49
Table 14 – Global parameters.....	50
Table 15 – Results in terms of average training accuracy, standard deviation, and statistical significance.....	52
Table 16 – Computational Cost (expressed in seconds).....	53

## SUMMARY

<b>CHAPTER 1 – INTRODUCTION</b>	09
1.1 RESEARCH CONTEXTUALIZATION	09
1.2 OBJECTIVES	10
1.3 RESEARCH ORGANIZATION	11
<b>CHAPTER 2 – METHODOLOGY</b>	12
2.1 RESEARCH CHARACTERIZATION	12
2.2 RESEARCH STRUCTURE	13
<b>CHAPTER 3 – PAPER 1</b>	15
1 INTRODUCTION	16
2 THE PROPOSED AJS ALGORITHM	19
2.1 CSA OVERVIEW	19
2.2 AJS INSPIRATION	19
2.3 AJS IMPLEMENTATION FOR CONTINUOUS PROBLEMS	20
2.3.1 Step 1: Adjustable parameters	21
2.3.2 Step 2: Initial population	21
2.3.3 Step 3: Fitness, Initial Dominant Leader and Global Solution	22
2.3.4 Step 4: Flocking	22
2.3.5 Step 5: Contempt	23
2.4 AJS VERSION TWO (AJS <sub>2</sub> )	27
3 EXPERIMENTAL SETUP	29
3.1 BENCHMARK FUNCTIONS	29
3.2 AJS PARAMETER TUNING	30
3.3 PARAMETER SETTINGS	33
4 RESULTS AND DISCUSSIONS	34
5 CONCLUSIONS AND FUTURE WORKS	39
<b>CHAPTER 4 – PAPER 2</b>	40
1 INTRODUCTION	41

2	AZURE JAY SEARCH (AJS)	43
2.1	AJS FOR CONTINUOUS OPTIMIZATION PROBLEMS	43
2.2	NEW BAJIS APPROACH FOR FS	45
3	DESIGN OF EXPERIMENTS	46
3.1	BENCHMARK DATA SET	46
3.2	VALIDATION, EVALUATION AND STATISTICAL TEST	48
3.3	PARAMETER SETTINGS	49
4	RESULTS AND DISCUSSION	50
5	CONCLUSIONS AND FUTURE WORKS	53
	<b>CHAPTER 5 – CONCLUSIONS</b>	<b>55</b>
	<b>REFERENCES</b>	<b>57</b>

# INTRODUCTION

---

This first chapter contextualizes the research, presents the general objectives to be achieved, as well as a synthesis of the remaining chapters that composes this dissertation.

## 1.1 RESEARCH CONTEXTUALIZATION

Over the past two decades, a growing variety of nature- and bio-inspired algorithms can be found in the literature. This constant increase is due to the fact that these metaheuristics find quality solutions to many complex and NP-hard problems (DHIMAN et al., 2021; YANG, 2021; EZUGWU et al., 2021). According to Molina et al. (2020), nature-inspired algorithms can be divided into Breeding-based Evolution, Swarm Intelligence (SI), Physics and Chemistry, Social Human Behavior and Plants Based. The SI algorithms, in particular, are based on agents (natural or artificial), which generally respond to environmental stimuli both individually and collectively, from a behavior considered intelligent when they interact with each other in a given environment (HORNISCHER et al., 2019; MOLINA et al., 2020). Over the years, the behavior patterns of different agents have served as inspiration for the construction of SI metaheuristics, such as aquatic animals, terrestrial animals, flying animals, microorganisms, among others (MOLINA et al., 2020). Among the algorithms inspired by flying animals, the bees (KARABOGA & BASTURK, 2007; MUÑOZ et al., 2009; WANG et al., 2021), the bats (ALSALIBI et al., 2021; YANG, 2010), and the crows (ASKARZADEH, 2016; TORABI & SAFI-ESFAHANI, 2018) stand out, among many others.

The tradeoff between exploration and exploitation is one of the key challenges in SI metaheuristics (AL-RIFAIE, 2021). In the exploitation phase, also known as diversification, it is expected that solutions with the greatest possible diversity are generated to ensure that the search is not limited to a small number of regions within the search space, that is, the objective of this phase is to explore the search space globally. On the other hand, the exploration phase, also known as intensification, aims to explore the search space locally. In this phase, algorithms use local information to thoroughly explore promising regions of the search space to find better solutions. One of the disadvantages is that the algorithm can easily get stuck in “local optimums”, as the final solution usually depends on the starting point (MORALES-CASTAÑEDA et al., 2020; YANG, 2014).

There are now hundreds of different SI algorithms that can be used to solve a wide range of problems (MORALES-CASTAÑEDA et al., 2020). According to Wolpert and Macready (1997), the No Free Lunch (NFL) theorem, in general, states that any algorithm may not performance well for a given type of problem, while the same algorithm may performance well for other problems. This means that there is no universally good algorithm for the optimization task, but an infinity of algorithms, each suitable for certain type(s) of problem(s). Furthermore, considering this growing emergence of new metaheuristics for solving complex problems and the various factors that can directly influence the resolution of these problems, choosing the algorithm that best fits the problem you want to solve or the data to be analyzed is of vital importance and it is usually the user's responsibility to choose the appropriate algorithm for a specific problem (NOCEDAL & WRIGHT, 2006).

Thus, the comparative study of metaheuristics brings relevant information about them, regarding their solving capacity and computational cost in face of problems of different areas and complexities. This information is very useful in making a decision regarding which metaheuristic to use considering the characteristics of the problem.

## 1.2 OBJECTIVES

The aim of this study is to propose a new SI metaheuristic and evaluate its performance compared to other state-of-the-art metaheuristics, in terms of average accuracy, standard deviation, dimensionality reduction and computational cost, applied to the FS problem in a set of fault diagnosis data in the steel industry. To achieve this general objective, as already said, this dissertation was structured using the multipaper model. A summary of the two papers is provided in subsection 2.2.

### 1.3 RESEARCH ORGANIZATION

The remainder of this dissertation is organized as follows. Section 2 describes the methodology used in the research, specifically its characterization and structure. Section 3 presents the first paper that makes up this study. This first paper aims to propose two versions of an SI-based metaheuristic to deal with continuous problems. Section 4 presents the second paper, this paper aims to propose a binary version of the algorithm proposed in the previous section to deal specifically with the Feature Selection task. Finally, section 5 presents the final considerations of the research.

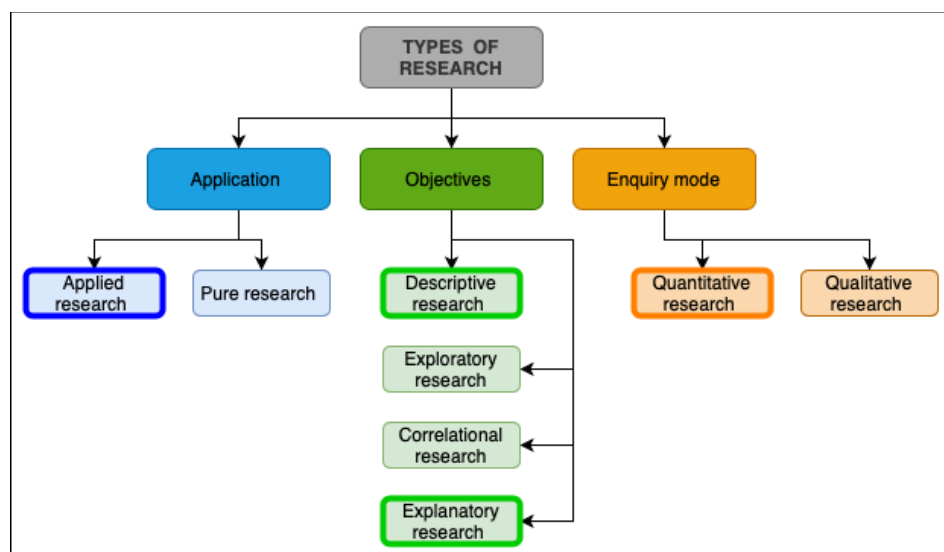
## METHODOLOGY

This chapter presents the methodology used in the research, specifically its characterization and structure considering the multipaper model.

### 2.1 RESEARCH CHARACTERIZATION

According to Kumar (2011), a scientific research can be discussed from 3 different perspectives: application, objectives and mode of enquiry (as shown in Figure 1).

Figure 1 – Types of Research



Source: (KUMAR, 2011).



Regarding the application, this research is presented as applied research as “the research techniques, procedures and methods that form the body of research methodology are applied to the collection of information about various aspects of a situation, issue, problem or phenomenon so that the information gathered can be used in other ways – such as for policy formulation, administration and the enhancement of understanding of a phenomenon” (KUMAR, 2011). Regarding the objectives, this research is characterized as explanatory and descriptive. According to Kumar (2011), descriptive research aims to “to describe systematically a situation, problem or phenomenon” and explanatory research aims to “clarify why and how there is a relationship between two aspects of a situation or phenomenon”. Finally, regarding the enquiry mode, this research is considered quantitative. As Moresi (2003) defines, the quantitative research aims to “translate opinions and information into numbers to classify and analyze them”, and for that, “require the use of statistical resources and techniques”.

## 2.2 RESEARCH STRUCTURE

Table 1 below details the two articles that make up the proposed multipaper model.

Table 1 – Summary of papers

	<b>Paper 1</b>	<b>Paper 2</b>
<b>Title</b>	Azure Jay Search (AJS): A Flocking-based modified Crow Search Algorithm (CSA)	A novel Binary Azure Jay Search (BAJS) for Feature Selection applied to Fault Diagnosis Problem
<b>Main Objective</b>	Development of two versions of a novel flocking-based modified CSA named AJS to solve continuous optimization problems	Development of a binary version of AJS for the Feature Selection (FS) problem
<b>Application</b>	10 fixed-dimension multimodal and public domain benchmark continuous functions	Wrapper-based FS task in a Fault Diagnosis dataset in the Steel Industry
<b>Compared Methods</b>	Sooty Tern Optimization Algorithm (STOA), Seagull Optimization Algorithm (SOA), Crow Search Algorithm (CSA) and Particle Swarm Optimization (PSO)	Binary Coyote Optimization Algorithm (BCOA), Binary Crow Search Algorithm (BCSA), Binary Dragonfly Algorithm (BDA), and Binary Particle Swarm Optimization

		(BPSO)
<b>Classification Algorithms</b>	-	Naïve Bayes, K-Nearest Neighbor and Random Forest
<b>Evaluation Criteria</b>	Average accuracy, standard deviation, and computational cost	Average training accuracy, standard deviation, computational cost, and number of selected features
<b>Validation Criteria</b>	-	10-fold Cross Validation
<b>Statistical Test</b>	Friedman test and Balanced two-way ANOVA (parameter tuning) and Wilcoxon signed-rank test	Wilcoxon signed-rank test
<b>Results</b>	Despite a relatively higher computational cost than the other compared techniques, both versions of AJS algorithm achieved good results in terms of average accuracy	Despite a relatively higher computational cost than the other compared techniques, the BAJ algorithm achieved good results in terms of average accuracy and number of selected features

Source: (THE AUTOR, 2021).

The first paper, according to Table 1, is proposed to solve continuous optimization problems, while the second paper is specifically proposed for the Feature Selection task. Both papers are included so that the main objective of this dissertation is achieved.

## PAPER 1

---

The first paper described in this chapter, contextualizes the research, describes the motivation for proposition of two versions of a new SI metaheuristic called Azure Jay Search (AJS) and presents a brief description of the two inspirations used in the construction of the algorithm, such as the CSA algorithm and the behavior of another bird found in nature that precisely gives its name to the AJS algorithm. Furthermore, it describes the step-by-step operation of the two versions of the proposed metaheuristic, the entire mathematical formulation behind the algorithms and the parameter tuning considering 625 different settings tested from Friedman's statistical test and the post-hoc Hochberg procedure. Finally, to evaluate the performance of the two versions of AJS (AJS<sub>1</sub> and AJS<sub>2</sub>), the paper describes the continuous optimization benchmark functions used in the comparison between the respective methods and other swarm-based metaheuristics proposed in the literature, in terms of average accuracy, standard deviation, computational cost and the statistical significance of the results using the Wilcoxon signed-rank test.

### **Azure Jay Search (AJS): A Flocking-based modified Crow Search Algorithm (CSA)**

#### **Abstract**

The optimization process seeks to find a solution (or a set of ideal solutions) that minimizes or maximizes the result of a function or problem. Metaheuristics, in general, are higher-level

heuristics that combine random choices and choices based on prior information to try to explore the solutions space as best as possible. One of the types of metaheuristics that exist are those inspired by nature. Within this category, we can highlight the Swarm Intelligence (SI) metaheuristics. A well-known SI metaheuristic is the Crow Search Algorithm (CSA), proposed in 2016. CSA seeks to solve optimization problems simulating the collective and intelligent behavior of a flock of crows in the environment. Despite a good performance, CSA has problems dealing with a multimodal search space. Thus, this paper aims to development two versions of a novel flocking-based modified CSA named Azure Jay Search (AJS) to solve continuous optimization problems. The performance of AJS, compared to other techniques, is evaluated in terms of average accuracy, standard deviation and computational cost when applied to 10 fixed-dimension multimodal and public domain benchmark functions. Despite a relatively higher computational cost, both versions of the AJS algorithm find good solutions in terms of average accuracy when compared to the other algorithms. The second version, which performed better than the first, when compared to the original CSA, obtained better average accuracy in 3 functions and statistically equivalent performance in 5 functions out of 10.

**Keywords:** modified Crow Search Algorithm (CSA), Azure Jay Search (AJS), Fixed-dimension multimodal functions.

## 1. INTRODUCTION

Mathematical Optimization deals with the process of selecting the solution or a set of ideal solutions for a specific mathematical function (or a problem) to minimize or maximize the result of such a function (FAUSTO et al., 2020). According to Cavazzuti (2013), the optimization methods can be divided into deterministic and stochastic ones and can make use of different search engines, such as Exhaustive Search (MNICH & RUDNICKI, 2020), Random Search (OZBEY et al., 2020; SABRI-LAGHAIE & KARIMI-NASAB, 2019), Local Search (BRANDÃO, 2020; TUBISHAT et al., 2020), Heuristic Search (LÓPEZ-SANTILLÁN et al., 2020; WANG et al., 2020) and Metaheuristic Search (MACEDO et al., 2016; THOM DE SOUZA et al., 2020, 2018). Exhaustive Search (also known as “the brute force” method) consists of a search strategy that explores the entire space of solutions, that is, it generates all feasible solutions for a given problem, aiming to find the best solution. According to Denil et al. (2014), Exhaustive Search is not recommended for most problems due to the high computational cost, since many solutions are generated. Random Search explores the solution space randomly, as mentioned by Denil et al. (2014), this technique is used both to obtain an

overview of the solution space and to create a starting point for other search strategies that require candidate solutions to start the optimization process. Local Search explores the solution space, walking through the neighborhood of a given solution. In other words, given a solution, this strategy explores solutions with features close to a previously chosen solution. According to Talbi (2014), one of the disadvantages of Local Search is that it converges toward local optima. According to Lu and Zhang (2013), Heuristic Search refers to a “search strategy that tries to optimize a problem, improving iteratively the solution based on a given heuristic function or a cost measure”. In other words, according to Thom de Souza (2008), heuristics use only information about the function to be optimized in their operation and operate in a “random-oriented” way to find good or acceptable solutions with a plausible computational cost, without the guarantee of finding the optimal solution.

In particular, metaheuristics consist of higher-level heuristics that are proposed for the solution of a wide range of optimization problems, ranging from single to multiobjective, continuous to discrete and constrained to unconstrained (DOKEROGLU et al., 2019). Moreover, many metaheuristics are finding success when applied to problems considered intractable, mainly due to the ability to find good solutions in a reasonable computational time (DOKEROGLU et al., 2019; HUSSAIN et al., 2019). To explore the search space on a global (exploration) and local (exploitation) scale, the operation of a metaheuristic, in general, combines choices based on experiences acquired from previous results and random choices. According to Fausto et al. (2020), for the global optimization achievement, it is important for a metaheuristic that presents a good trade-off between exploitation and exploration.

As reported by Talbi (2014), there are several ways to classify a metaheuristic based on its features. One way is to divide them into nature-inspired versus uninspired. Several sources of inspiration are found in nature for the development of optimization algorithms, according to Molina et al. (2020), five branches can be highlighted: Breeding-based Evolution (XU et al., 2020), Physics and Chemistry (AZIZI, 2021; TARAMASCO et al., 2020), Social Human Behavior (BOGAR & BEYHAN, 2020), Plants Based (EMAMI & SHARIFI, 2020; MENG et al., 2021) and Swarm Intelligence (SI). SI is inspired by the collective behavior of simple agents (biological or artificial), which according to Hornischer et al. (2019), can exhibit a stunning degree of organization.

As mentioned by Dokeroglu et al. (2019), most of the state-of-the-art metaheuristics were developed before the year 2000 and among the SI algorithms considered “classics” are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). Although, different recent works that address SI algorithms are found in the literature, such as Tunicate Swarm

Algorithm (TSA) (KAUR et al., 2020), Chameleon Swarm Algorithm (CSA) (BRAIK, 2021), Sine-cosine and Spotted Hyena-based Chimp Optimization Algorithm (SSC) (DHIMAN, 2021), Artificial Jellyfish Search (AJS) (CHOU & TRUONG, 2021; SHAHEEN et al., 2021), Meerkat Optimization Algorithm (MOA) (SRINIVASAN et al., 2021), Multiprocess Salp Swarm Optimization (SSO) (MARTINEZ-RIOS & MURILLO-SUAREZ, 2021a), Multi-threaded Spotted Hyena Optimizer (MT-SHO) (MARTINEZ-RIOS & MURILLO-SUAREZ, 2021b), Improved Whale Optimization Salp Swarm Algorithm (IWOSSA) (SAAFAN & EL-GENDY, 2021), Hybrid Firefly Algorithm with Grouping Attraction (HFA-GA) (CHENG et al., 2021), Enhanced Harris Hawks Optimization (RLHHO) (LI et al., 2021), Improved Bat Algorithm with Extremal Optimization Algorithm (IBA-EO) (CHEN et al., 2021), Artificial Gorilla Troops Optimizer (GTO) (ABDOLLAHZADEH et al., 2021b), African Vultures Optimization Algorithm (AVOA) (ABDOLLAHZADEH et al., 2021a), Coot Optimization Algorithm (NARUEI & KEYNIA, 2021), Rat Swarm Optimizer (DHIMAN et al., 2021), Honey Badger Algorithm (HASHIM et al., 2022), among others.

Proposed by Askarzadeh (2016), the Crow Search Algorithm (CSA) is a population-based metaheuristic that seeks to solve optimization problems simulating the collective and intelligent behavior of a flock of crows in the environment. CSA is easy to implement and only two parameters that influence its operation i.e. the flight length and awareness probability (MOHAMMADI & ABDI, 2018). In general, despite a good performance, the CSA has problems dealing with a multimodal search space, presenting premature convergence, stagnation and inability to avoid local optimum (COELHO et al., 2017; ISLAM et al., 2019; 2021);

Thus, the main goal and contribution of this paper include:

1. development two versions of a novel flocking-based modified CSA named Azure Jay Search (AJS) to solve continuous optimization problems.
2. use of 10 fixed-dimension multimodal and public domain benchmark functions to test the performance of the proposed methods in terms of average accuracy, standard deviation, and computational cost.
3. performance comparison between the proposed method with other SI based metaheuristics such as Sooty Tern Optimization Algorithm (STOA), Seagull Optimization Algorithm (SOA), Crow Search Algorithm (CSA) and Particle Swarm Optimization (PSO).

The remainder of this paper is structured as follows. Section 2 presents an overview of the CSA and the inspiration and mathematical formulation of the two versions of the proposed

AJS. Section 3 presents the setup of all experiments done in this paper along. Section 4 presents the obtained results and discussions. Section 5 concludes the paper and states the future work.

## 2. THE PROPOSED AJS ALGORITHM

This section details the CSA operation, the inspiration and mathematical modeling of the AJS proposed algorithms.

### 2.1 CSA OVERVIEW

Proposed by Askarzadeh (2016), the CSA is a population-based metaheuristic that simulate the intelligent behavior of crows in nature to solve continuous optimization problems. In general, crows live in flocks, hide their food, memorize the place, and protect their hiding places from other crows, as they have a habit of following each other with the sole purpose of stealing food.

The new positions that each crow will assume during the search process are calculated based on the awareness probability parameter. This means that, at a given moment, crow  $i$  decides to follow crow  $j$  with the aim of discovering its hiding place and stealing its food. In this case, two states can happen, according to Equation 1:

$$x^{j,i+1} = \begin{cases} x^{j,i} + r_i \times fl^{j,i} \times (m^{j,i} - x^{j,i}) & r_j \geq AP^{j,i} \\ \text{a random position} & \text{otherwise} \end{cases} \quad (1)$$

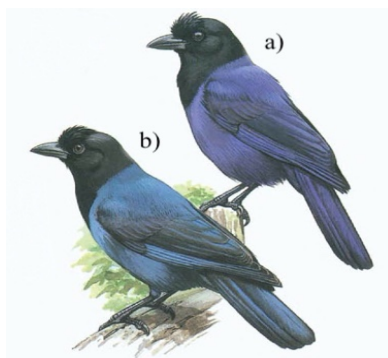
where  $r_i$  and  $r_j$  are a random number with uniform distribution between 0 and 1,  $fl^{j,i}$  denotes the flight length of crow  $j$  at iteration  $i$ ,  $m^{j,i}$  is the best position memorized that crow  $j$  has obtained so far and  $AP^{j,i}$  denotes the awareness probability of crow  $j$  at iteration  $i$ .

### 2.2 AJS INSPIRATION

Jay is a common name for various small size passeriform birds, usually noisy, which belong to the crow family, Corvidae. The Azure Jay (*Cyanocorax caeruleus*) is a near threatened globally blue jay (BirdLife International, 2021), found in the Araucaria Moist Forests, coniferous forests of the Atlantic Forest Biome, full of Parana pines (*Araucaria angustifolia*), located in southern Brazil and immediately adjacent areas, far eastern Paraguay, and far northeastern Argentina. Paraná pine and Azure Jay are, respectively, the symbol tree and the symbol bird of Paraná (state of Brazil).

Regarding its appearance, the bird has a head, neck, and upper breast black or sooty, while the rest of its plumage is blue, usually cobalt blue or purplish-blue (Figure 2a), but it can also have a greenish-blue hue (Figure 2b). However, what is most striking about this bird, as well as other crows, are its actions, which demonstrate a high degree of intelligence.

Figure 2 – Typical purplish-blue bird (a) and greenish-blue morph (b)



Source: (MADGE & BURN, 1999).

The behavior and the annual cycle of Azure Jay were observed for approximately two years (July 1985 to June 1987) on a farm in the municipality of Palmeira (70 km west of Curitiba), in the State of Paraná (ANJOS, 1991; ANJOS & VIELLIARD, 1993). Its annual cycle, according to Brady (2010), can be divided into three periods: the first period comprises the autumn season (approximately from April to July), when Azure Jays are looking for Araucaria seeds, intending to stock the pine nuts for food; the second period comprises the end of winter and the beginning of spring (approximately from August to September), when Azure Jays feed on insects and small animals, considering that pine nuts are not available at this time of year and; the third period comprises part of spring and practically all summer (approximately October to March), when they breed and care for the young. In addition, azure jays have a complex communication system (ANJOS & VIELLIARD, 1993), being able to communicate with each other through vocalizations with specific purposes, such as contact call and flight call: report your location to the group; social-call: call the group closer to itself; social alarm call: alert each other about mobbing predators; threat call: disobey an order from the dominant leader; social identity call: asking not to be disturbed; hunger call: supplicate for food; courtship call: dating/flirting; imitative calls: imitate the vocalization of another species; among others.

### 2.3 AJS IMPLEMENTATION FOR CONTINUOUS PROBLEMS

The proposed AJS algorithm is a population and blackboard-based metaheuristic that attempts to simulate the communication and search for food performed by Azure Jays in the first period



of its annual cycle to find the solution to continuous optimization problems. To assist in understanding the mathematical representation of the proposed implementation, it is assumed that:

- AJS starts with a flock of  $N$  jays flying in a  $d$ -dimensional environment.
- Jays move in space having a velocity, position, and memory.
- Jays can search for food alone or in small flocks.
- There is a dominant leader in the flock who usually has the best source of pine nuts (fitness).
- Jays communicate throughout the search process.

In addition, five main operators are considered in their operation. The first is called “hiding”, when the jays bury the excess of pine nuts in a heavily loaded Araucaria in hiding places. The second is called “flocking”, when the leader orders the formation and the direction of the group. The third is called “contempt”, when the jay can fly to a different location from the commanded (usually looking for hiding places from other jays). The fourth is called “begging”, when the jays supplicate for food, and the fifth is called “compassion”, when a jay gives up its hiding place to another jay after supplication.

### 2.3.1 Step 1: Adjustable parameters

AJS adjustable parameters are initialized offline, that is, the values of different parameters are fixed before the execution of the metaheuristic (TALBI, 2014). In addition to the dimension of problem search space (decision variables) ( $d$ ), maximum number of iterations ( $max_{iter}$ ) and population size ( $N$ ), AJS has the following parameters: cognitive parameter ( $c_1$ ), social parameter ( $c_2$ ), inertia parameter ( $w$ ) and compassion probability ( $c_p$ ).

### 2.3.2 Step 2: Initial population

The position, velocity and memory of a given jay  $j$  at time  $i$  (iteration) in the search space are represented respectively by the vectors:  $pos_d^{j,i} = [pos_1^{j,i}, pos_2^{j,i}, \dots, pos_d^{j,i}]$ ,  $vel_d^{j,i} = [vel_1^{j,i}, vel_2^{j,i}, \dots, vel_d^{j,i}]$  e  $mem_d^{j,i} = [mem_1^{j,i}, mem_2^{j,i}, \dots, mem_d^{j,i}]$ , where  $j \in [1, N]$ ,  $i \in [1, max_{iter}]$ . If the initial population is not well diversified, a premature convergence can occur (TALBI, 2014). Thus, the initial population and velocity are generated randomly in the range  $[l_b, u_b]$ , representing the lower and upper bounds allowed, as shown in Equations 2 and 3:

$$pos_d^{j,1} = l_b + (u_b - l_b) * rand_j \quad (2)$$

$$vel_d^{j,1} = l_b + (u_b - l_b) * rand_j \quad (3)$$

where  $rand_j$  is a uniformly distributed random variable in the range  $[0, 1]$ .

About the initial memory of the jays, as they have no previous experience in the first iteration, it is assumed that they hide their foods at their initial positions, that is,  $mem_d^{j,1} = pos_d^{j,1}$ .

### 2.3.3 Step 3: Fitness, Initial Dominant Leader and Global Solution

The quality of the position that each jay occupies in the search space is calculated by inserting the values of the decision variables into the fitness function, according to Equation 4. In the first iteration, the dominant leader and the global solution correspond to the jay that has the hiding place with the largest number of pine nuts, that is, the best source of food (fitness).

$$fit^{j,i} = f(pos_d^{j,i}) \quad (4)$$

The best fitness, in a minimization problem, consists of the lowest value resulting from the insertion of decision variables in the fitness function, and in a maximization problem, the highest value.

### 2.3.4 Step 4: Flocking

The dominant leader flies and orders the formation and the direction of the flock. The new velocity and position of the leader  $dom$  are updated according to Equations 5 and 6.

$$vel_d^{dom,i+1} = w * vel_d^{dom,i} + c_2 * rand * (Gbest_d - pos_d^{dom,i}) \quad (5)$$

$$pos_d^{dom,i+1} = pos_d^{dom,i} + vel_d^{dom,i+1} \quad (6)$$

where  $rand$  is a uniformly distributed random variable in the range  $[0, 1]$  and  $Gbest$  consists of the position of the best source of food so far.

The fitness of the leader's new position is calculated from Equation 4 and his memory is updated as shown in Equation 7:

$$mem_d^{dom,i+1} = \begin{cases} pos_d^{dom,i+1} & f(pos_d^{dom,i+1}) \text{ is better than } f(mem_d^{dom,i}) \\ mem_d^{dom,i} & \text{otherwise} \end{cases} \quad (7)$$

### 2.3.5 Step 5: Contempt

The jays can choose to obey (named beggars) or not obey (named rebels) the flocking order issued by the leader. In this case, three states may happen:

#### State 1: There are no rebels

In this case, all the jays follow the leader, that is, there are no rebels. The mathematical formulation that denotes whether a jay is rebel or not is shown in Equation 8:

$$\begin{cases} Rebel & f(pos_d^{j,i}) \text{ is better than } f(pos_d^{dom,i+1}) \\ Beggar & \text{otherwise} \end{cases} \quad (8)$$

The new velocity and position of the beggar jay  $b$  is updated based on the position of the leader, shown in Equations 9 and 10:

$$vel_d^{b,i+1} = w * vel_d^{b,i} + c_1 * rand * (mem_d^{b,i} - pos_d^{b,i}) + c_2 * rand * (pos_d^{dom,i+1} - pos_d^{b,i}) \quad (9)$$

$$pos_d^{b,i+1} = pos_d^{b,i} + vel_d^{b,i+1} \quad (10)$$

The fitness of the new position of the jay  $b$  is calculated from Equation 4 and his memory is updated as shown in Equation 7. From there, the global solution is updated. In this state, the new global solution consists of the best food source memorized so far considering all the jays.

#### State 2: All are rebels

In this case, all the jays disobey the flocking order of the leader and fly in another direction. The new velocity and position of the rebel jay  $r$  is updated according to Equations 11 and 12:

$$vel_d^{r,i+1} = w * vel_d^{r,i} + c_1 * rand * (mem_d^{r,i} - pos_d^{r,i}) \quad (11)$$

$$pos_d^{r,i+1} = pos_d^{r,i} + vel_d^{r,i+1} \quad (12)$$

The fitness of the new position of the jay  $r$  is calculated from Equation 4 and his memory is updated as shown in Equation 7. From there, the global solution is updated. In this state, the new global solution consists of the best food source memorized so far by the current leader.

### State 3: Begging and Compassion

In this case, there are both rebel and beggar jays. The rebel jay fly in another direction in relation to the leader, disobeying his flocking order, as shown by Equations 11 and 12, the fitness of each rebel jay is calculated according to Equation 4 and their memories are updated according to Equation 7. Each beggar jay chooses a rebel jay to beg for food, as they know better sources of pine nuts. The mathematical formulation that denotes whether a rebel jay has compassion or not with another beggar jay is shown in Equation 13.

$$\begin{cases} \text{Rebel jay has compassion} & rand < cp \\ \text{Rebel jay has no compassion} & \text{otherwise} \end{cases} \quad (13)$$

If a particular rebel jay  $r$  has compassion, the new velocity and position of the beggar jay  $b$  is updated based on the position of the same rebel jay  $r$ , shown in Equations 14 and 15:

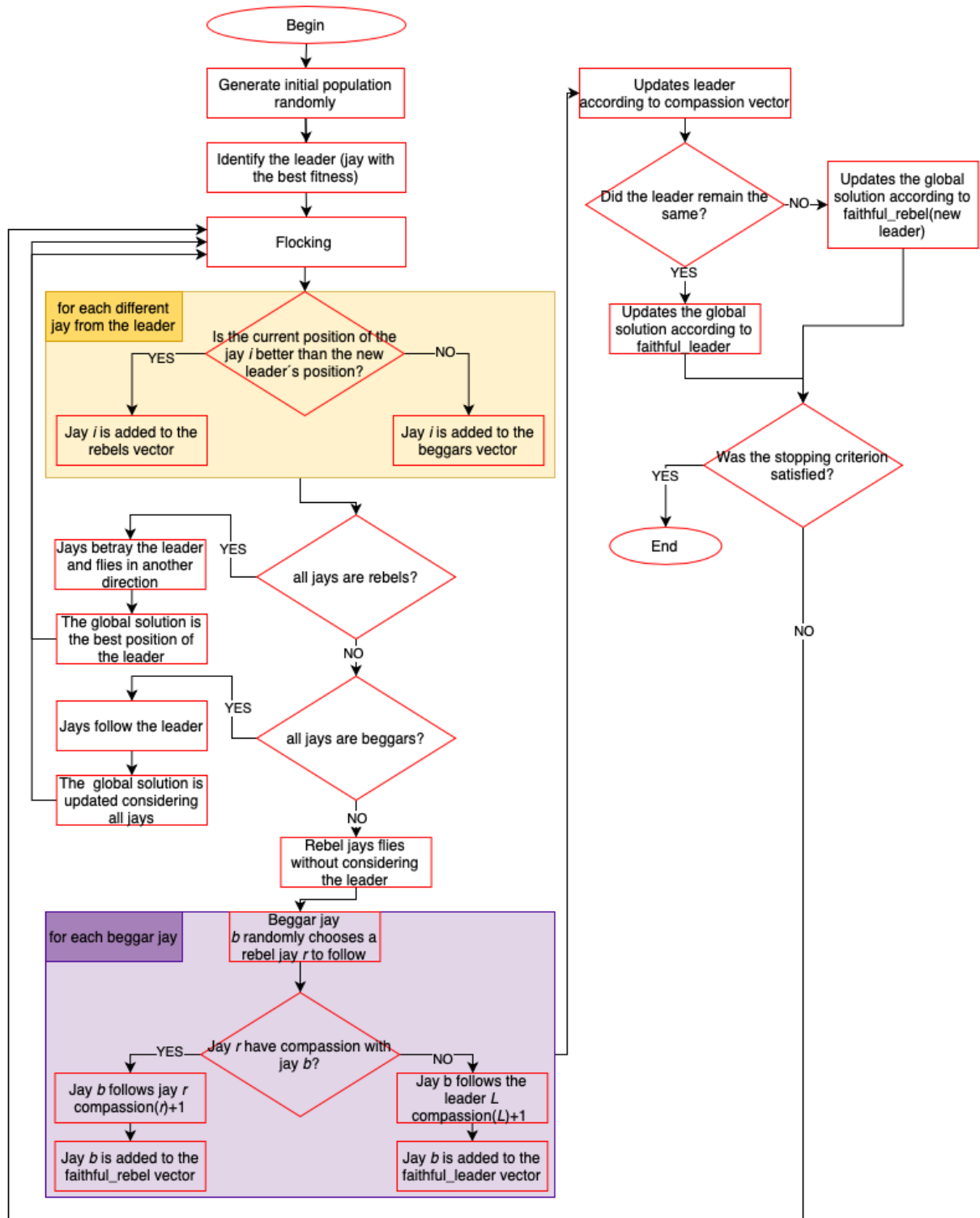
$$vel_d^{b,i+1} = w * vel_d^{b,i} + c_1 * rand * (mem_d^{b,i} - pos_d^{b,i}) + c_2 * rand * (pos_d^{r,i+1} - pos_d^{b,i}) \quad (14)$$

$$pos_d^{b,i+1} = pos_d^{b,i} + vel_d^{b,i+1} \quad (15)$$

If a particular rebel jay does not have compassion, the beggar jay repents and returns to follow the leader. Thus, the new velocity and position of the beggar jay  $b$  is updated according to Equations 9 and 10. The fitness of each jay is calculated according to Equation 4 and their memories are updated according to Equation 7.

Still in state 3 (when there are both rebel and beggar jays), the next step is to update the leader. In this case, for the position of new leader, considering the currently leading jay and the rebel jays that received supplication for food, the jay that was most compassionate with the others was chosen. After the leader update step, the global solution is updated. If the leader remained the same, the global solution is updated considering only the jays that followed the leader, otherwise, the global solution is updated considering the jays that followed the rebel jay (new leader). The Figure 3 presents the flowchart and Figure 4 the pseudocode of the proposed AJS.

Figure 3 – AJS flowchart



Source: (THE AUTOR, 2021).

Figure 4 – AJS pseudocode

**Algorithm 01: Pseudo-code of AJS**


---

```

1: Begin
2: Randomly initialize the position and velocity of a flock of  $N$  jays
3: Evaluate the position of the jays
4: Initialize the memory of each jay
5: Determine the dominant leader ( $dom$ )
6: while maximum generation is not reached do
7:    $dom$  flies and orders the formation and direction of the group (Eqs. 4 and 5)
8:   Calculate the fitness (Eq. 3) and update the  $dom$ 's memory (Eq. 6)
9:   for  $i = 1:N$  (all  $N$  jays of the flock (except the  $dom$ ))
10:    if (fitness( $i$ ) < fitness( $dom$ )) (Eq. 7)
11:      Jay  $i$  is added to the rebels vector
12:    else
13:      Jay  $i$  is added to the beggars vector
14:    end
15:  end
16:  if sum(rebels) == 0
17:    There are no rebels, all jays follow the leader (Eqs. 8 and 9)
18:    Calculate the fitness (Eq. 3) and update the memory (Eq. 6) of all jays
19:    Update global solution (considering all jays)
20:  elseif sum(beggars) == 0
21:    All are rebels, all jays disobey the flocking order and fly in another direction (Eqs. 10 and 11)
22:    Calculate the fitness (Eq. 3) and update the memory (Eq. 6) of all jays
23:    Update global solution (best food source memorized so far by the current leader)
24:  else
25:    Rebel jay(s) fly(ies) in another direction without considering flocking order (Eqs. 10 and 11)
26:    for  $i = 1:m$  (number of beggar jays)
27:      Beggar jay  $i$  choose randomly a rebel jay  $r$  to beg for food
28:      if (rand <  $cp$ ) (Eq.12)
29:        Rebel jay  $r$  has compassion with beggar jay  $i$ 
30:        Calculate the new position of the beggar jay  $i$  (Eqs. 13 and 14)
31:        Compassion( $r$ )+= 1
32:      else
33:        Rebel jay  $r$  has no compassion with beggar jay  $i$ 
34:        Calculate the new position of the beggar jay  $i$  (Eqs. 8 and 9)
35:        Compassion( $dom$ )+= 1
36:      end
37:    end
38:    Calculate the fitness (Eq. 3) and update the memory (Eq. 6) of all jays
39:    Update  $dom$  according to compassion vector
40:    Update global solution (If the leader remained the same, the global solution is updated consider only the jays followed the leader, otherwise, the global solution is updated considering the jays that followed the rebel jay (new leader)
41:  end
42: end
43: end

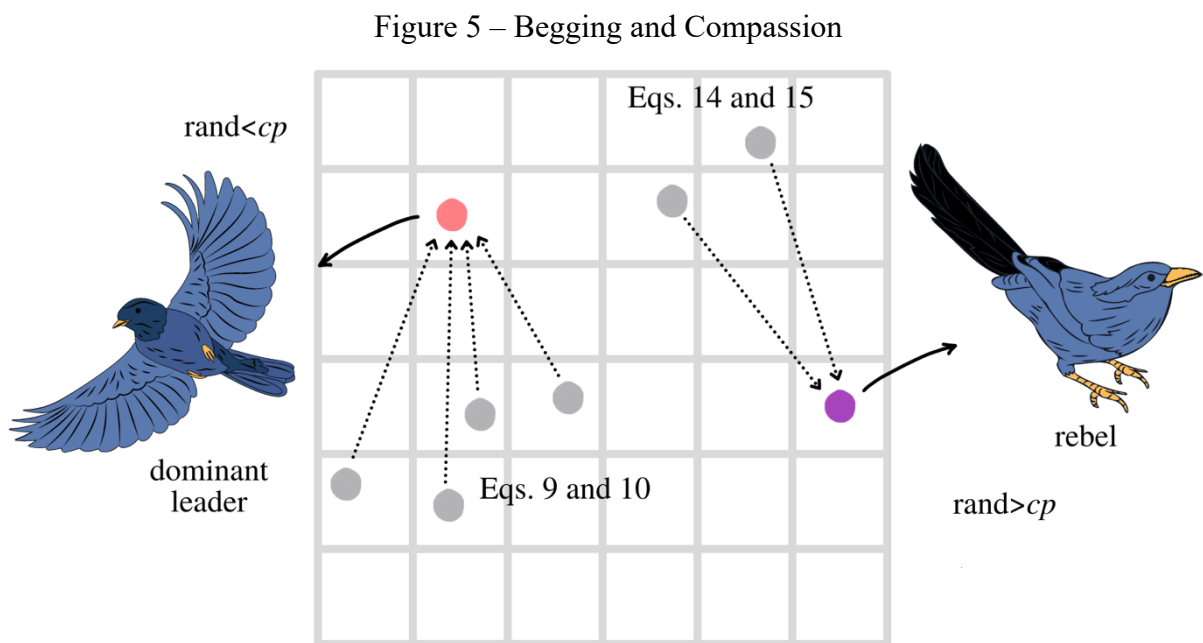
```

---

Source: (THE AUTOR, 2021).

The AJS algorithm tries to explore the solution space as best as possible and turns between states of exploitation and exploration subtly and throughout the search process. At each iteration, as explained above, the dominant leader (jay that usually has the best food source) flies in a certain direction and orders the jays in the flock to follow. Two moments after

flocking can be highlighted as exploitation states: 1) State 1: There are no rebels and 2) State 3: Begging and Compassion. In the first moment of exploitation, as the name suggests, all jays obey the order of the dominant leader and fly to follow him, concentrating on that neighborhood of solutions, that is, the search process in a local region intensifies. About the state of exploration, two moments can also be highlighted: 1) State 2: All are rebels and 2) State 3: Begging and Compassion. At the first moment of exploration, as the name also suggests, all jays disobey the leader's order of direction and fly to a different way (place where they believe they can find better solutions when compared to the leader), that is, the process of search in different regions intensifies. Finally, about the second moment (both from the state of exploration and the state of exploitation), the local and global search occurs simultaneously, as there are jays following the leader and rebel jays seeking to explore other places in the search space at the same time, exactly as shown in Figure 5.



Source: (THE AUTOR, 2021).

To further encourage the explorability of the algorithm, a second version is proposed. The only difference between the first version already described and the second is a random element added to version 2, described in detail in the following section.

#### 2.4 AJS VERSION TWO (AJS<sub>2</sub>)

As already mentioned, the only difference between version 01 proposed and version 02 is the insertion of a random element. This random element is inserted in step 5 (subsection 2.2.5), specifically in state 3: Begging and Compassion. Version 02, unlike version 01 (as can be

compared in Figure 6), the begging jays first beg for food to a rebel jay, this rebel jay can be compassionate or not compassionate. If the rebel jay does not have compassion ( $rand > cp$ ), the beggar jay tries to follow the leader again, but at that time the leader has the freedom to decide whether to accept that beggar jay back. If the leader does not have compassion for the beggar jay ( $rand > cp$ ), a random position in the search space is generated.

Figure 6 – Random element added to version 02 (pseudo-code)

---

**Algorithm 01: Pseudo-code of AJS (version 01)**

---

```

26:   for  $i = 1:m$  (number of beggar jays)
27:     Beggar jay  $i$  choose randomly a rebel jay  $r$  to beg for food
28:     if ( $rand < cp$ ) (Eq.12)
29:       Rebel jay  $r$  has compassion with beggar jay  $i$ 
30:       Calculate the new position of the beggar jay  $i$  (Eqs. 13 and 14)
31:       Compassion( $r$ )+= 1
32:     else
33:       Rebel jay  $r$  has no compassion with beggar jay  $i$ 
34:       Calculate the new position of the beggar jay  $i$  (Eqs. 8 and 9)
35:       Compassion( $dom$ )+= 1
36:     end
37:   end

```

---



---

**Algorithm 02: Pseudo-code of AJS (version 02)**

---

```

26:   for  $i = 1:m$  (number of beggar jays)
27:     Beggar jay  $i$  choose randomly a rebel jay  $r$  to beg for food
28:     if ( $rand < cp$ ) (Eq.12)
29:       Rebel jay  $r$  has compassion with beggar jay  $i$ 
30:       Calculate the new position of the beggar jay  $i$  (Eqs. 13 and 14)
31:       Compassion( $r$ )+= 1
32:     else
33:       if ( $rand < cp$ )
34:         Rebel jay  $r$  has no compassion with beggar jay  $i$ 
35:         Calculate the new position of the beggar jay  $i$  (Eqs. 8 and 9)
36:         Compassion( $dom$ )+= 1
37:       else
38:         A random position is generated for beggar jay  $i$  (Eqs. 1 and 2)
39:       end
40:     end
41:   end
42: end

```

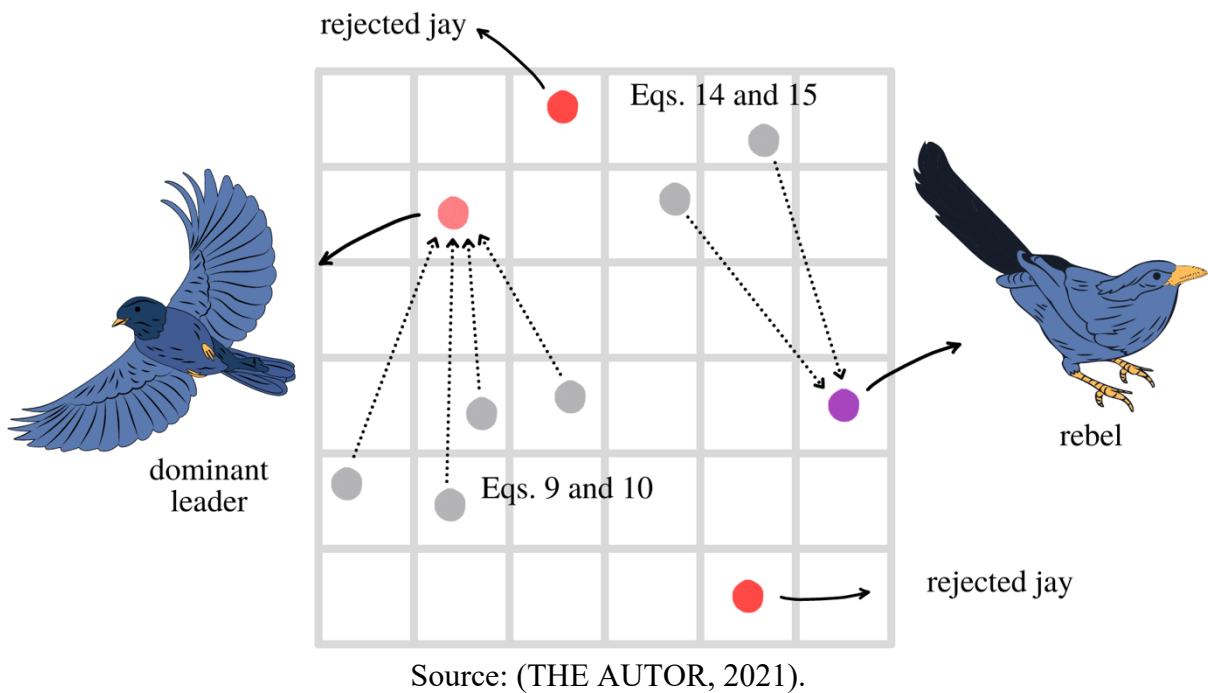
---

Source: (THE AUTOR, 2021).

The Figure 7 is an update of Figure 5 and illustrates how space can be explored more intensely in terms of global scale.



Figure 7 – Begging and Compassion (version 02)



### 3. EXPERIMENTAL SETUP

To test the performance of the proposed AJS, the experiments detailed in this section were run on MATLAB R2020b in a macOS Big Sur operating system environment with Intel(R) Core (TM) i5 Dual-Core (1.80 GHz) and 8 Gigabytes (GB) of Random Access Memory (RAM). Below are described the benchmark functions, the algorithms used in the comparison and their respective parameters.

#### 3.1 BENCHMARK FUNCTIONS

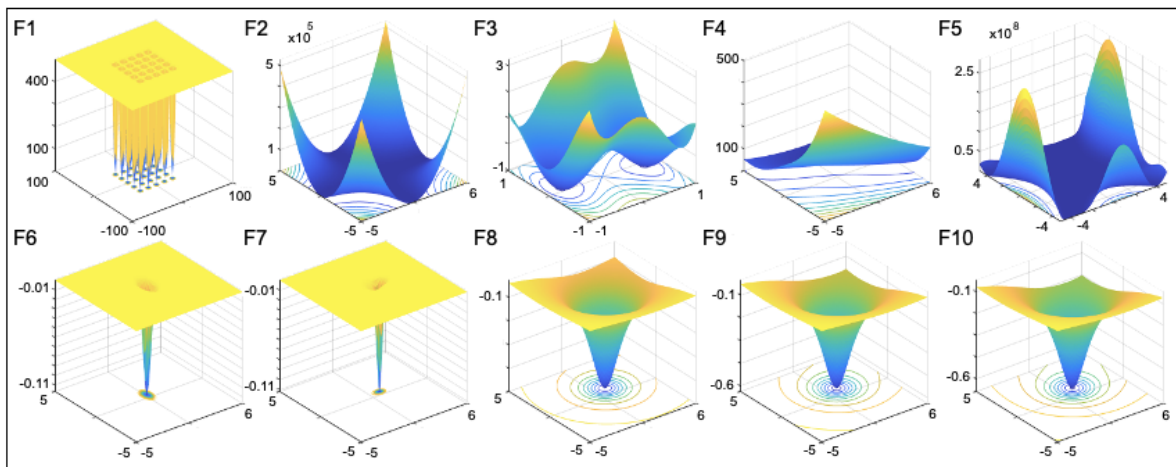
The benchmark functions used in this paper are fixed-dimension multimodal and minimization functions (F1 to F10). Furthermore, these same functions are called “classic” by (MIRJALILI et al., 2014) and can be found over the years in different works in the literature (DIGALAKIS & MARGARITIS, 2001; MENG et al., 2021; MIRJALILI et al., 2014; MIRJALILI & LEWIS, 2013; SAAFAN & EL-GENDY, 2021; XUE & SHEN, 2020; YAO et al., 1999). Figure 8 illustrate the search space of benchmark functions and Table 2 shows the mathematical formulas and other details such as dimension ( $D$ ), upper and lower bounds (Range) and optimal solution ( $F_{min}$ ) of the same.

Table 2 – Details of fixed-dimension multimodal benchmark functions.

Function	D	Range	$f_{min}$
$f_1(x) = \left( \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right)^{-1}$	2	[-65,65]	1
$f_2(x) = \sum_{i=1}^{11} \left[ a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[-5,5]	0.00030
$f_3(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316
$f_4(x) = \left( x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	[-5,5]	0.398
$f_5(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
$f_6(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2)$	3	[1,3]	-3.86
$f_7(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2)$	6	[0,1]	-3.32
$f_8(x) = -\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_9(x) = -\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4028
$f_{10}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

Source: (THE AUTOR, 2021).

Figure 8 – Benchmark functions search space.



Source: (THE AUTOR, 2021).

### 3.2 AJS PARAMETER TUNING

Generally, there are adjustable parameters that influence the operation of any metaheuristic. In this way, TALBI (2014) asserts that there is no universally ideal set of values for such parameters and as they can have a crucial influence on research efficiency and effectiveness, it is vitally important that they be carefully adjusted. The process of adjusting these parameters is considered a challenge (DHIMAN & KAUR, 2019) and can be a time-consuming task (YU & LI, 2015). Some strategies aim precisely to provide guidelines for choosing a great parameter configuration, that is, a configuration that leads the metaheuristic to obtain a reasonable

performance for solving different optimization problems. Among these strategies, we can mention off-line parameter initialization (also called meta-optimization) and online parameter tuning (TALBI, 2014).

In the present paper, we opted for the offline adjustment, whose values of different parameters are fixed before the execution of the metaheuristic. In this process, we seek the best configuration of values for four parameters of the proposed algorithms, totaling 625 configurations of different parameters, namely  $[c_1, c_2, w, c_p] \times [0.1, 0.5, 1, 5, 10] \times [0.1, 0.5, 1, 5, 10] \times [0.1, 0.5, 1, 5, 10] \times [0.1, 0.3, 0.5, 0.7, 0.9]$  for all 10 functions detailed in Table 2. For each function, the 625 settings were run 30 times independently with  $10.000 \times D$  iterations each. To carry out the analysis of the sensitivity of the parameters and verify the best combinations of values, we divided the task into two stages, according to the work of Yu & Li (2015) and Carrasco et al. (2020).

The first step consisted of applying the Friedman test to compare the 625 parameter settings for the 10 test functions. The Friedman test is similar to the classic balanced two-way ANOVA and tests the null hypothesis that all configurations perform equally against the alternative that some performance is significantly different. In the second step, if any configuration tested in the previous step is statistically different, a post-hoc Hochberg procedure is adopted to identify which configurations are in fact significantly different and which are not. Thus, it is possible to determine a good setting of values or a range of settings where the metaheuristic performs significantly better than others.

Table 3 and 4 presents the results found by the proposed  $AJS_1$  and  $AJS_2$ , respectively, in terms of average accuracy and standard deviation, best and worst mean accuracy considering the 625 parameter settings for all 10 functions.

Table 3 –  $AJS_1$  parameter tuning result

Function	Mean $\pm$ std	Best	Worst
$f_1$	45532.9 $\pm$ 24616.6	3229.8	71332.0
$f_2$	0.0770 $\pm$ 0.0792	0.0010	0.3278
$f_3$	-0.1239 $\pm$ 0.9522	-1.0316	4.7735
$f_4$	1.2443 $\pm$ 0.6934	0.3980	2.9724
$f_5$	4.8500 $\pm$ 6.0700	3.0000	63.0980
$f_6$	-3.5111 $\pm$ 0.2227	-3.8628	-3.0913
$f_7$	-1.9137 $\pm$ 0.5281	-3.2777	-0.8659
$f_8$	-1.4310 $\pm$ 1.4954	-8.8516	-0.3073
$f_9$	-1.6285 $\pm$ 1.5874	-10.2680	-0.4234
$f_{10}$	-1.7624 $\pm$ 1.4944	-8.7044	-0.7149

Source: (THE AUTOR, 2021).

Table 4 – AJS<sub>2</sub> parameter tuning result

Function	Mean $\pm$ std	Best	Worst
$f_1$	21.4993 $\pm$ 50.1512	0.9980	368.6800
$f_2$	0.0150 $\pm$ 0.0447	0.0006	0.4423
$f_3$	-0.7516 $\pm$ 0.8177	-1.0316	10.9100
$f_4$	0.5776 $\pm$ 0.4483	0.3980	3.9591
$f_5$	3.4464 $\pm$ 3.6750	3.0000	46.6450
$f_6$	-3.7180 $\pm$ 0.2126	-3.8628	-2.9405
$f_7$	-2.4247 $\pm$ 0.6635	-3.3220	-1.0951
$f_8$	-3.2624 $\pm$ 2.4451	-10.1530	-0.3501
$f_9$	-3.4678 $\pm$ 2.6106	-10.4030	-0.3989
$f_{10}$	-3.5777 $\pm$ 2.6714	-10.5360	-0.5958

Source: (THE AUTOR, 2021).

During the parameter setting process, from Tables 3 and 4, it is possible to notice that version 02 reached the optimum solution in 8 functions, while version 01 reached the optimum solution in only 4 functions. On the other hand, Table 5 and 6 deal only with the best combination and all those that presented a statistically equivalent result. Specifically, each column brings an AJS parameter, and each row presents the values that parameter assumed considering the previously selected combinations.

Table 5 – AJS<sub>1</sub> parameter values

Function	$c_1$	$c_2$	$w$	$c_p$
$f_1$	[0.5;1;10]	[0.1]	[0.1]	[0.5;0.7;0.9]
$f_2$	[0.1;0.5;1]	[1]	[0.5]	[0.3;0.5;0.7;0.9]
$f_3$	[0.1;0.5;1;10]	[0.1;0.5;1]	[0.1;0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_4$	[0.1;0.5;1]	[0.5;1]	[0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_5$	[0.1;0.5;1]	[0.1;0.5;1]	[0.1;0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_6$	[0.1;0.5;1]	[1]	[0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_7$	[0.1;0.5;1]	[0.5;1]	[0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_8$	[0.1;0.5;1;10]	[0.5;1]	[0.1;0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_9$	[0.1;0.5;1;10]	[0.1;0.5;1]	[0.1;0.5]	[0.3;0.5;0.7;0.9]
$f_{10}$	[0.1;0.5;1]	[0.5;1]	[0.5]	[0.3;0.5;0.7;0.9]

Source: (THE AUTOR, 2021).

It is possible to notice that in version 02, a high number of combinations achieved a result (in terms of average training accuracy) relatively close to optimal, while in version 01 it is easier to find a pattern of values that positively influenced the algorithm, since the number of combinations that achieved good results is reduced. In general, the values of one of the combinations that had the most evidence was permanently chosen for the AJS parameters and can be seen in Table 7.

Table 6 – AJS<sub>2</sub> parameter values

Function	$c_1$	$c_2$	$w$	$c_p$
$f_1$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7]
$f_2$	[0.1;0.5;1]	[0.1;0.5;1]	[0.1;0.5]	[0.1;0.3;0.7;0.9]
$f_3$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1]	[0.1;0.3;0.5;0.7;0.9]
$f_4$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]
$f_5$	[0.1;0.5;1]	[0.1;0.5;1]	[0.1;0.5]	[0.1;0.3;0.5;0.7;0.9]
$f_6$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]
$f_7$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]
$f_8$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]
$f_9$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]
$f_{10}$	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.5;1;10]	[0.1;0.3;0.5;0.7;0.9]

Source: (THE AUTOR, 2021).

### 3.3 PARAMETER SETTINGS

Table 7 – Parameter settings of optimization methods for comparison of the AJS

Method	Parameters	Values	Brief description
Azure Jay Search (AJS)	$c_1$ (cognitive parameter)	1.0	The AJS simulates the behavior of jays and their complex communication system when searching for food in the environment.
	$c_2$ (social parameter)	1.0	
	$w$ (inertia parameter)	0.5	
	$c_p$ (compassion probability)	0.5	
Sooty Tern Optimization Algorithm (STOA)	$C_j$ (Controlling variable)	2.0	The STOA algorithm is inspired by the migration and attacking behaviors of sea bird sooty tern in nature (DHIMAN & KAUR, 2019)
Seagull Optimization Algorithm (SOA)	$A$ (Control parameter)	[2,0]	The SOA algorithm is inspired by the migration and attacking behaviors of a seagull in nature (DHIMAN & KUMAR, 2019).
	$F_c$ (Control parameter)	2.0	
Crow Search Algorithm (CSA)	$fl$ (Flight length)	2.0	The CSA algorithm is inspired by the intelligent behavior of crows. (ASKARZADEH, 2016).
	$AP$ (Awareness probability)	0.1	
Particle Swarm Optimization (PSO)	$c_1$ (Cognitive constant)	1.0	The PSO is inspired by the social sharing of information among conspecifics (KENNEDY & EBERHART, 1995).
	$c_2$ (Social constant)	1.0	
	$w$ (Local constant)	0.3	
	$I$ (Inertia Coefficient)	0.75	

Source: (THE AUTOR, 2021).

The AJS algorithm was compared to one swarm-based algorithms considered state-of-the-art and to other optimization strategies whose inspiration for its operation also consisted in

observing the behavior of other birds in nature. Table 7 provides a brief description of each strategy and the respective adjustable parameters and their values (each value was set according to the original paper of the respective method, except PSO (KHISHE & MOSAVI, 2020)) and Table 8 presents the global parameters and the respective adopted values.

Table 8 – Global parameters

Parameter	Value
Number of iterations	10.000 * $D$
Number of independent runs	30
Number of search agents	30

Source: (THE AUTOR, 2021).

#### 4. RESULTS AND DISCUSSIONS

The experimental results are shown in Tables 9 and 10. Table 9 shows the results of AJS<sub>1</sub>, and Table 10 shows the results of the AJS<sub>2</sub>, compared to the other algorithms.

The Wilcoxon signed-rank test, proposed by Frank Wilcoxon (WILCOXON, 1945), is a nonparametric method for comparison of two paired samples, used to verify if there are significant differences between them. In the present paper, we calculate the average training accuracy after 30 independent runs (samples) and the Wilcoxon signed-rank test is applied to compare these runs of each algorithm against the correspondent 30 independent runs of AJS<sub>1</sub> and AJS<sub>2</sub>.

Each column in Tables 9 and 10 represents one metaheuristic algorithm. For each function, the average accuracy (Acc) the standard deviation (Std), and the significant statistical difference (line “T”) between the AJS<sub>1</sub> and AJS<sub>2</sub> and the other proposed algorithms are shown. The best results are highlighted in boldface and the symbols used to show if there is a significant difference between the independent runs of the AJS<sub>1</sub> and AJS<sub>2</sub> and the other algorithms are: “≈”, “+” and “-”. The “≈” symbol means that the average accuracy of the two algorithms are statistically equivalent, the “+” and “-” symbol means that there is a statistical difference between the average accuracy of the two algorithms, but the first means that the other algorithm compared to AJS<sub>1</sub> or AJS<sub>2</sub> has better performance and the second means that the other algorithm has worse performance.

Table 9 shows that the proposed AJS<sub>1</sub> got better performance than the CSA in function  $f_3$  (statistically equivalent to STOA, SOA and PSO), better than CSA and SOA in function  $f_6$  (statistically equivalent to STOA and PSO) and better than CSA, STOA and SOA in function

$f_7$  (statistically equivalent to PSO). For function  $f_9$ , although  $AJS_1$  lost to CSA, it performed better than STOA, SOA and PSO.

Table 9. Results in terms of average accuracy, standard deviation, and statistical significance of the fixed-dimension multimodal benchmark functions ( $AJS_1$ ).

		$AJS_1$	STOA	SOA	CSA	PSO
$f_1$	Acc	2.9424	0.9980	0.9980	<b>0.9980</b>	1.4619
	Std	1.8248	$4.51E - 16$	$4.51E - 16$	<b>0</b>	1.0069
	T		+	+	+	+
$f_2$	Acc	0.0009	0.0011	0.0011	<b>0.00030</b>	0.0011
	Std	0.0004	0.0002	0.0003	<b>0</b>	0.0014
	T		-	≈	+	≈
$f_3$	Acc	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	-10.316	<b>-1.0316</b>
	Std	<b>0</b>	$6.77E - 16$	$6.77E - 16$	0	$6.77E - 16$
	T		≈	≈	-	≈
$f_4$	Acc	<b>0.3979</b>	<b>0.3978</b>	<b>0.3978</b>	<b>0.3978</b>	<b>0.3978</b>
	Std	<b>0</b>	$1.69E - 16$	$1.69E - 16$	<b>0</b>	$1.69E - 16$
	T		≈	≈	≈	≈
$f_5$	Acc	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	<b>3.9000</b>
	Std	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	5.0137
	T		≈	≈	≈	≈
$f_6$	Acc	<b>-3.8628</b>	<b>-3.8549</b>	-3.0150	-38.6280	<b>-3.8370</b>
	Std	<b>0</b>	<b>0</b>	0.5906	0	0.1435
	T		≈	-	-	≈
$f_7$	Acc	<b>-3.2570</b>	-2.9351	-3.7471	-4.2790	<b>-3.2496</b>
	Std	0.0850	0.4044	0.2808	5241.5223	<b>0.0661</b>
	T		-	-	-	≈
$f_8$	Acc	-7.1383	-0.7481	-1.9940	<b>-10.1530</b>	-6.3059
	Std	2.5995	0.8406	3.3603	<b>0</b>	3.3093
	T		-	-	+	≈
$f_9$	Acc	-7.2883	-3.6309	-4.2760	<b>-10.4030</b>	-5.6121
	Std	3.3481	4.2956	4.3206	<b>0</b>	3.4343
	T		-	-	+	-
$f_{10}$	Acc	-8.3430	-7.0940	-7.9249	<b>-10.5360</b>	-5.5903
	Std	3.1464	4.4515	4.1742	<b>0</b>	3.6936
	T		≈	≈	+	-

Source (THE AUTOR, 2021).

In the Table 9, for functions  $f_2$ ,  $f_8$  and  $f_{10}$ ,  $AJS_1$  despite also losing to CSA, got better performance than STOA in function  $f_2$  (statistically equivalent to SOA and PSO), got better performance than STOA and SOA in function  $f_8$  (statistically equivalent to PSO) and got better performance than PSO in function  $f_{10}$  (statistically equivalent to SOA and STOA). For functions  $f_4$  and  $f_5$ , all techniques had statistically equivalent performance. Finally, in function  $f_1$   $AJS_1$  had the worst performance when compared to STOA, SOA, CSA and PSO.

Table 10. Results in terms of average accuracy, standard deviation, and statistical significance of the fixed-dimension multimodal benchmark functions (AJS<sub>2</sub>).

		AJS <sub>2</sub>	STOA	SOA	CSA	PSO
$f_1$	Acc	<b>1.0640</b>	<b>0.9980</b>	<b>0.9980</b>	<b>0.9980</b>	1.4619
	Std	0.2520	$4.51E - 16$	$4.51E - 16$	<b>0</b>	1,0069
	T		≈	≈	≈	–
$f_2$	Acc	0.0005	0.0011	0.0011	<b>0.00030</b>	0.0011
	Std	0.0003	0.0002	0.0003	<b>0</b>	0.0014
	T		–	–	+	–
$f_3$	Acc	<b>–1.0316</b>	<b>–1.0316</b>	<b>–1.0316</b>	–10.316	<b>–1.0316</b>
	Std	<b>0</b>	$6.77E - 16$	$6.77E - 16$	0	$6.77E - 16$
	T		≈	≈	–	≈
$f_4$	Acc	<b>0.3978</b>	<b>0.3978</b>	<b>0.3978</b>	<b>0.3978</b>	<b>0.3978</b>
	Std	<b>0</b>	$1.69E - 16$	$1.69E - 16$	<b>0</b>	$1.69E - 16$
	T		≈	≈	≈	≈
$f_5$	Acc	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	<b>3.9000</b>
	Std	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	5.0137
	T		≈	≈	≈	≈
$f_6$	Acc	<b>–3.8628</b>	–3.8549	–3.0150	–38.6280	–3.8370
	Std	<b>0</b>	0	0.5906	0	0.1435
	T		–	–	–	≈
$f_7$	Acc	<b>–3.2820</b>	–2.9351	–3.7471	–4.2790	–3.2496
	Std	<b>0.0570</b>	0.4044	0.2808	5241.5223	0.0661
	T		–	–	–	–
$f_8$	Acc	–8.6373	–0.7481	–1.9940	<b>–10.1530</b>	–6.3059
	Std	2.3547	0.8406	3.3603	<b>0</b>	3.3093
	T		–	–	+	–
$f_9$	Acc	<b>–9.6997</b>	–3.6309	–4.2760	<b>–10.4030</b>	–5.6121
	Std	1.8235	4.2956	4.3206	<b>0</b>	3.4343
	T		–	–	≈	–
$f_{10}$	Acc	<b>–10.3570</b>	–7.0940	–7.9249	<b>–10.5360</b>	–5.5903
	Std	0.9790	4.4515	4.1742	<b>0</b>	3.6936
	T		–	–	≈	–

Source (THE AUTOR, 2021).

Table 10 shows that the proposed AJS<sub>2</sub> also obtained better accuracy than the CSA in function  $f_3$  (statistically equivalent to STOA, SOA and PSO), got better performance than CSA, STOA and SOA in function  $f_6$  (statistically equivalent to PSO) and got better performance than CSA, STOA, SOA and PSO in function  $f_7$ . For functions  $f_2$  and  $f_8$ , although AJS<sub>2</sub> lost to CSA, it performed better than STOA, SOA and PSO. For functions  $f_9$  and  $f_{10}$ , AJS<sub>2</sub> got better performance than STOA, SOA and PSO (statistically equivalent to CSA). For functions  $f_4$  and  $f_5$ , all techniques had statistically equivalent performance. Finally, in function  $f_1$ , AJS<sub>2</sub> got better performance than PSO (statistically equivalent to CSA, STOA and SOA).



The results can also be seen in Figure 9, which shows the value of the best fitness found over the 30 independent runs (average training accuracy), for all techniques. Through Figure 9 and comparison of tables 9 and 10, it is possible to verify, in general, that the  $AJS_2$  performed better than the  $AJS_1$ . While  $AJS_1$  managed to reach the optimum solution in 4 functions,  $AJS_2$  reached in 8 functions. Furthermore, while  $AJS_1$  performed better than CSA for 3 functions and statistically equivalent performance in 2 functions,  $AJS_2$  also performed better than CSA in 3 functions and statistically equivalent performance in 5 functions. In addition, it is important to comment that  $AJS_2$  did not lose (in terms of performance) of the STOA, SOA and PSO techniques in any function (or gained or achieved statistically equivalent performance).  $AJS_1$  also had an interesting performance compared to STOA, SOA and PSO techniques, showing better performance or statistically equivalent performance for 9 out of 10 functions.

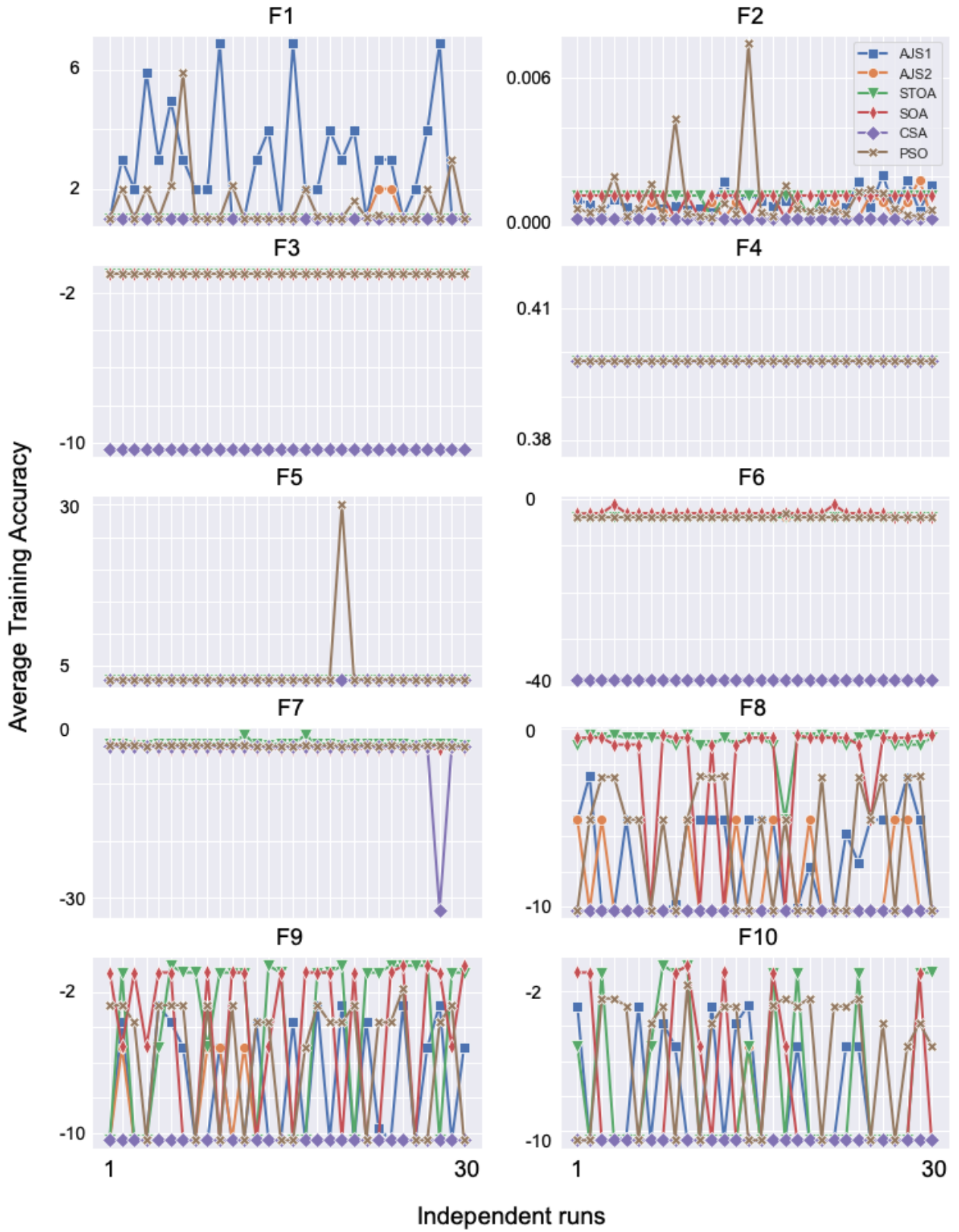
Table 11 presents the computational cost (expressed in seconds) of each of the compared techniques. Each column represents a different technique and each row a function. It is possible to see that the algorithm with the best computational cost for all functions is the PSO. For 7 functions out of 10 the CSA algorithm obtained the second-best computational cost, in third place are STOA and SOA, which had very close computational cost, and finally,  $AJS_1$  and  $AJS_2$ , which also had a similar computational cost for several functions.

Table 11. Computational cost

	$AJS_1$	$AJS_2$	STOA	SOA	CSA	PSO
$f_1$	549.39	566.42	549.44	539.30	584.97	123.12
$f_2$	134.23	123.11	90.81	90.51	70.67	32.76
$f_3$	61.21	59.32	33.96	34.15	28.70	17.98
$f_4$	41.46	43.60	30.19	29.46	27.54	16.95
$f_5$	65.27	51.04	29.65	28.30	24.99	13.71
$f_6$	116.99	133.67	66.19	64.71	61.58	47.65
$f_7$	223.81	397.52	158.20	158.48	114.06	107.33
$f_8$	167.17	207.88	108.54	106.98	110.23	81.96
$f_9$	187.27	195.88	119.01	123.00	113.63	106.76
$f_{10}$	222.06	227.10	141.56	139.05	144.59	127.23

Source: (THE AUTOR, 2021).

Figure 9 – Average training accuracy (30 independent runs) obtained by each technique



Source: (THE AUTOR, 2021).

## 5. CONCLUSIONS AND FUTURE WORKS

The CSA algorithm is a population-based metaheuristic that simulates the intelligent behavior of crows when searching for food in the environment. Despite its good performance, the algorithm has difficulty in dealing with multimodal solutions spaces, and it can often get stuck in local optimums. Thus, we propose two versions of the AJS algorithm, a modified CSA based on flocking.

The AJS aims to solve continuous optimization problems. In its operation, a flock of jays and their dominant leader, look for loaded Araucarias to store food. In an optimization process, jays are the researchers, and the best food source is the global solution to the problem. Generally, the dominant leader has the best source of pine nuts and orders the direction of the flock in the environment (solution space). In this leader-led process, the jays may eventually rebel against him and fly to a different location than the leader. When this happens, it means that the rebel jays have found good sources of food. In this way, the other jays beg the rebel jay for food. Versions 1 and 2 of the algorithms differ precisely in this stage of begging and compassion, since in version 2 a random element is added to further explore the space for solutions on a global scale. The performance of the AJS algorithm was compared, in terms of mean accuracy, standard deviation and computational cost, to other optimization strategies also inspired by the behavior of birds in nature, when submitted to 10 fixed-dimensional multimodal benchmark functions. Despite a relatively high computational cost when compared to other techniques, both versions of the AJS algorithm performed very well (in terms of average accuracy) compared to other optimization techniques. When compared to each other, the second version performed better than the first. And when compared to CSA, the first version had better performance for 3 functions and statistically equivalent performance in 2 functions, and the second version also had better performance for 3 functions, but statistically equivalent performance in 5 functions.

As future work, we intend to explore the structure of the algorithm (AJS<sub>2</sub>) to allow multiple groups (flocks). In this way, rejected jays have the possibility of forming a new group when no one shows compassion for them. It is also intended to explore different values for the parameter of compassion of a rebel jay and the parameter of compassion of the dominant leader, considering that in the current version there is only one parameter for both. Furthermore, we intend to propose a binary version of AJS, with the objective of working with Feature Selection (FS) problems.

## PAPER 2

---

The second paper described in this chapter, in addition to contextualizing the research, details the proposal of a binary version of the AJS algorithm (proposed in the previous chapter), named BAJJS, for the Feature Selection (FS) problem. To introduce the binary version, it is first described the step-by-step and mathematical formulation of the original version of the AJS, proposed for solving continuous optimization problems. After explaining the original AJS, the binarization strategy used is detailed. To achieve the main objective of this dissertation, the BAJJS algorithm is submitted to the FS task (in a wrapper-based model) to a set of Fault Diagnosis data and compared to other techniques for the same problem, in terms of average training accuracy, standard deviation, computational, number of features selected and statistical significance.

### **A novel Binary Azure Jay Search (BAJJS) for Feature Selection applied to Fault Diagnosis Problem**

#### **Abstract**

Feature Selection (FS) is considered a binary optimization problem and an important data pre-processing method. In the literature, due to good performance, it is possible to find a multitude of recent metaheuristic algorithms, especially algorithms based on swarm intelligence, proposed precisely for the FS problem. In principle, these metaheuristics are first proposed to

work with continuous optimization problems, to operate in binary solution spaces, some binarization strategy must be employed. Thus, the aim of this paper is proposing a binary version of a recent swarm intelligence algorithm known as Azure Jay Search (AJS). The proposed Binary AJS (BAJS) is applied for the FS task in a Fault Diagnosis dataset in the steel industry in a wrapper-based model. BAJS performance is evaluated in terms of average training accuracy, standard deviation, computational cost, and number of selected features, and compared to other swarm intelligence metaheuristics. Despite a relatively higher computational cost, the BAJS algorithm find good solutions in terms of average training accuracy and subsets with a relatively small number of features. For the Naive Bayes (NB) and Random Forest (RF) classifiers, the BAJS obtained, respectively, the best and the second-best average training accuracy (statistically equivalent to other two techniques).

**Keywords:** Feature Selection (FS) problem, Binary Azure Jay Search (BAJS), Fault Diagnosis (FD) problem.

## 1. INTRODUCTION

Optimization is a field of study that aims to maximize or minimize a given function (usually of several variables), often subject to a set of constraints. According to (TILAHUN & NGNOTCHOUYE, 2017), an optimization problem can be classified into three different categories: problems dealing with continuous, non-continuous and mixed decision variables. In a range, continuous variables can have any value (integer or fractional) and non-continuous or discrete variables are restricted to integer or binary values. In binary optimization, the variables involved in the process can only have two values: zero or one.

Feature Selection (FS) is considered a binary optimization problem and an important data pre-processing method (ZHANG et al., 2020). The FS can be applied to select the most significant features while removing those that are irrelevant/redundant or possibly problematic (noisy features). Some benefits of FS can be highlighted, such as ease of visualization and understanding of data, reduction in computational cost and data storage, improvement in classification accuracy, among others (ALHAMIDI & JATMIKO, 2020). Two well-known FS methods are called filter (CHAUDHURI & SAHU, 2021; CUI et al., 2021; OUADFEL & ABD ELAZIZ, 2021) and wrapper (AMINI & HU, 2021; LIU & WANG, 2021; TARKHANEH et al., 2021). The main difference between these two methods is the relationship between the FS strategy and a classifier algorithm, that is, while the wrapper selects features based on the

evaluation made by a classification algorithm, the filter selects features independently, without direct contact with a classifier (TUBISHAT et al., 2021).

In the literature, it is possible to find a multitude of recent metaheuristic algorithms proposed especially for the FS problem. In particular, are swarm intelligence-based algorithms, such as Binary Grasshopper Optimization Algorithm (BGOA) (WANG et al., 2020), Binary Coyote Optimization Algorithm (BCOA) (THOM DE SOUZA et al., 2020), Binary Grey Wolf Optimizer (BGWO) (HU et al., 2020), Binary Pigeon Inspired Optimizer (BPIO) (ALAZZAM et al., 2020), Binary Salp Swarm Algorithm (HEGAZY et al., 2020; NEGGAZ et al., 2020; TUBISHAT et al., 2020), Binary Dragonfly Algorithm (BDA) (HAMMOURI et al., 2020), Binary Moth-Flame Optimization (BMFO) (ELAZIZ et al., 2020), Binary Whale Optimization Algorithm (BWOA) (MAFARJA et al., 2020; NEMATZADEH et al., 2019), Binary Artificial Bee Colony (BABC) (WANG et al., 2020), Binary Particle Swarm Optimization (BPSO) (ROSTAMI et al., 2020; XUE et al., 2020), Binary Firefly Algorithm (BFA) (MARIE-SAINTE & ALALYANI, 2020; SELVAKUMAR & MUNEESWARAN, 2019), among others.

At first, these metaheuristics, in general, are designed to deal with the resolution of continuous optimization problems, that is, their agents move in a continuous search space. For them to work with problems where the search space is binary, some binarization strategy must be employed. According to Crawford et al. (2017), binarization strategies can be classified into two main groups, namely: two-step binarization and continuous-binary operator transformation. The first group consists of continuing to work with continuous search space (but adding operators that transform the solution from  $\mathbb{R}^n$  to  $\{InterSpace\}$ ) and involves techniques such as Transfer Functions (KILIÇ et al., 2021; THOM DE SOUZA et al., 2018; 2020), Great Value Priority and Angle Modulation (DONG et al., 2021; SLEZKIN et al., 2021). The second group consists to redefine the operators of the metaheuristics and involves techniques such as Boolean Approach (GUNASUNDARI et al., 2016, 2018), Set-Based Approach (ENGELBRECHT et al., 2019; JIA et al., 2018), Quantum Binary Approach (ALVAREZ-ALVARADO et al., 2021; LU & HE, 2021; ZHANG et al., 2021) and Binary Method Based on Estimation of Distribution (STRASSER et al., 2016).

Furthermore, it is possible to find in the literature the application of many of these metaheuristics to well-known binary problems, such as Knapsack Problem (ABDOLLAHZADEH et al., 2021; LIU, 2020), Traveling Salesman Problem (AL-GAPHARI et al., 2021; PANDIRI et al., 2020), Clique Problem (WANG et al., 2020; ZHOU et al., 2020), Routing problems (LI et al., 2020; SONG et al., 2020), among others.

The aim of this paper is proposing a binary version of a recent swarm intelligence algorithm known as Azure Jay Search (AJS). The proposed Binary AJS (BAJS) is applied for the FS task in a Fault Diagnosis dataset in the steel industry in a wrapper-based model. BAJS performance is evaluated in terms of average training accuracy, standard deviation, computational cost, and number of selected features, and compared to other swarm intelligence metaheuristics such as Binary Coyote Optimization Algorithm (BCOA), Binary Crow Search Algorithm (BCSA), Binary Dragonfly Algorithm (BDA), and Binary Particle Swarm Optimization (BPSO).

The remainder of this paper is structured as follows. Section 2 presents a brief description of the original AJS and mathematical formulation of the proposed BAJS. Section 3 describes the design of the experiments. The obtained results are presented and discussed in Section 4. Section 5 concludes the paper and proposes directions for future research.

## 2. AZURE JAY SEARCH (AJS)

The AJS algorithm, is inspired by a bird from the crow family (Corvidae), the symbol of Paraná (state of Brazil) and almost globally threatened, called Azure Jay. This section briefly describes the inspiration and mathematical modeling of AJS to solve continuous optimization problems and the new BAJS approach to FS problem.

### 2.1 AJS FOR CONTINUOUS OPTIMIZATION

Jays are noisy birds and have a complex communication system. In addition, its main food source is the seeds of Araucaria (pine nuts), trees found mainly in southern Brazil. From there, the objective of the AJS is precisely to simulate the behavior of these jays in the environment (focusing on their communication during the search for food) to find the solution to continuous optimization problems. Figure 10 consists of a reduced flowchart and illustrates the general operation of the algorithm.

In more detail, the algorithm has five operators (highlighted in bold throughout the explanation below and in green in Figure 10) and its operation basically resembles a flock of jays ( $N$ ) flying (usually in small groups and directed by a dominant leader (**flocking**)) in a certain  $d$ -dimensional environment, hiding their food (the excess of pine nuts of loaded araucarias), memorizing the position of the secret hiding place (**hiding**) and always looking for new sources of food. Thus, each jay moves in space having a velocity, position, and memory. The flocking moment is when the leader flies to a certain location in the solutions space and

orders the formation and direction of the group (the new leader velocity and position are updated according to Equations 16 and 17).

$$vel_d^{dom,i+1} = w * vel_d^{dom,i} + c_2 * rand * (Gbest_d - pos_d^{dom,i}) \quad (16)$$

$$pos_d^{dom,i+1} = pos_d^{dom,i} + vel_d^{dom,i+1} \quad (17)$$

where  $rand$  is a uniformly distributed random variable in the range  $[0, 1]$ ,  $Gbest$  consists of the position of the best source of food so far,  $w$  is the inertia parameter and  $c_2$  is the social parameter.

During this relentless quest, usually guided by the dominant leader (jay who usually has the best food source), as explained above, one or more rebel's jays may refuse to obey your flocking order, moving in another direction in search of new sources of food (**contempt**). The mathematical formulation that denotes whether a jay is rebel or not is shown in Equation 18:

$$\begin{cases} Rebel & f(pos_d^{j,i}) \text{ is better than } f(pos_d^{dom,i+1}) \\ Beggar & \text{otherwise} \end{cases} \quad (18)$$

where  $pos_d^{j,i}$ ,  $pos_d^{dom,i+1}$ ,  $f(pos_d^{j,i})$  and  $f(pos_d^{dom,i+1})$  are the position of a given jay  $j$  and dominant leader  $dom$  in the current iteration  $i$  and next iteration  $i + 1$  and the fitness function value corresponding to this positions.

At this moment of possible contempt, three things can happen: all jays can rebel and fly to a certain position in the solution space (Equations 19 and 20) or all jays can faithfully follow the leader (Equations 21 and 22) or when there are  $n_r$  rebel jays (where  $0 < n_r < N$ ), the remaining jays ( $N - n_r$ ), called beggars, believing that the rebellious jays can obtain a better source of food than ever before, or even find the hiding place of some other jay in the herd, choose to beg a particular rebel jay for food (**begging**).

$$vel_d^{r,i+1} = w * vel_d^{r,i} + c_1 * rand * (mem_d^{r,i} - pos_d^{r,i}) \quad (19)$$

$$pos_d^{r,i+1} = pos_d^{r,i} + vel_d^{r,i+1} \quad (20)$$

$$vel_d^{b,i+1} = w * vel_d^{b,i} + c_1 * rand * (mem_d^{b,i} - pos_d^{b,i}) + c_2 * rand * (pos_d^{dom,i+1} - pos_d^{b,i}) \quad (21)$$

$$pos_d^{b,i+1} = pos_d^{b,i} + vel_d^{b,i+1} \quad (22)$$



where  $c_1$  is the cognitive parameter,  $mem_d^{r,i}$  and  $mem_d^{b,i}$  are the hiding place (best position) so far (iteration  $t$ ) of the rebel jay  $r$  and beggar  $b$ , respectively.

In this situation, the rebel jays may or may not allow some jay to follow and feed on their source (compassion). If the rebel jays have no compassion, the beggar jay can ask the leader to follow you again. The leader also has the option to allow the jay back or not. The mathematical formulation that denotes whether a rebel jay or the leader will have compassion or not is shown in Equation 23:

$$\begin{cases} \text{Rebel/Leader jay has compassion} & rand < cp \\ \text{Rebel/Leader jay has no compassion} & \text{otherwise} \end{cases} \quad (23)$$

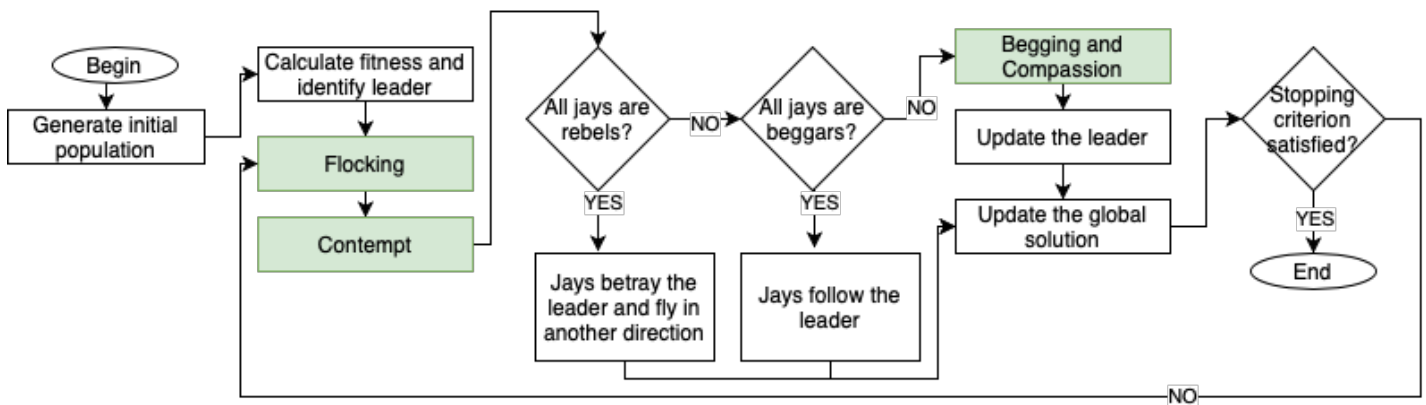
where  $cp$  is the compassion probability and  $rand$  is a uniformly distributed random variable in the range  $[0, 1]$ .

In this moment of bagging and compassion, if a rebel jay has compassion and allows a beggar jay to follow her, the new velocity and position of that beggar jay is updated according to Equations 24 and 25. If a beggar jay comes back to follow the leader, his velocity and position are updated according to Equations 21 and 22. Otherwise, if no one has compassion for a beggar jay, a random position is generated in the search space.

$$vel_d^{b,i+1} = w * vel_d^{b,i} + c_1 * rand * (mem_d^{b,i} - pos_d^{b,i}) + c_2 * rand * (pos_d^{r,i+1} - pos_d^{b,i}) \quad (24)$$

$$pos_d^{b,i+1} = pos_d^{b,i} + vel_d^{b,i+1} \quad (25)$$

Figure 10 – AJS reduced flowchart



Source: (THE AUTOR, 2021).

## 2.2 NEW BAJIS APPROACH FOR FS

According to Crawford et al. (2017), a continuous search space is a set of all possible solutions of the optimization problem that satisfy the problem's constraints. In a continuous search space, the variables can have any value in the given interval, however, in a binary search space, the variables can have only two values: 0 or 1. As already mentioned, FS is a binary problem.

In the wrapper-based FS problem, for example, the objective is select the smallest subset of features with the greatest classification accuracy. To formulate this problem, one can let  $X_i$  be the binary variable such that:

$$X_i = \begin{cases} 1 & \text{if feature } i \text{ was selected,} \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

There are some techniques that can be used to convert solutions from continuous to binary space. In general, transfer functions are used to generate the probability of changing a position's elements to 0 or 1 based on the value of the step vector of the  $j$  search agent in the  $d$  dimension in the current iteration  $t$  as an input parameter (MAFARJA et al., 2018). These transfer functions are divided into two main categories: S-shaped and V-shaped (ABDOLLAHZADEH et al., 2021). The transfer function used in BAJIS belongs to the V-shaped family and is described in Equation 27.

$$V(pos_d^{j,i+1}) = \left| \frac{2}{\pi} \arctan \left( \frac{\pi}{2} pos_d^{j,i} \right) \right| \quad (27)$$

where  $pos_d^{j,i+1}$  is a position of a given jay  $j$  at time  $i$  in dimension  $d$ .

After applying the transfer function, for each dimension  $d$  of a jay  $i$ , a binary value 0 or 1 is assigned with a probability 0.5 as follows:

$$binary(pos_d^{j,i+1}) = \begin{cases} 1 & \text{if } V(pos_d^{j,i+1}) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (28)$$

## 3. DESIGN OF EXPERIMENTS

We compare the performance of the proposed BAJIS against other techniques using the wrapper approach based on Naïve Bayes (NB), K-Nearest Neighbors (KNN) and Random Forest (RF)

classifiers to perform FS. This section describes the dataset used, the validation and evaluation criteria, the statistical test and the FS techniques and their respective parameters.

### 3.1 BENCHMARK DATA SET

Most engineering activities entail judgments concerning product, service, and system design, all of which are tied in some manner to enhancing performance, productivity, sustainability, quality, safety, and efficiency while reducing cost, energy consumption, flaws, and environmental effect (YANG et al., 2016). The new industrial revolution called Industry 4.0 refers precisely to the integration of a variety of technologies and agents such as Internet of Things (IoT) (ASHIMA et al., 2021; MARIYAPRINCY & SAMIAPPAN, 2021), Cyber-physical Systems (CPS) (NEAL et al., 2021; TRAGANOS et al. (2021), Cloud Computing (JAVED et al., 2021), Big Data (BERGES et al., 2021), Machine Learning (BRIK et al., 2019; O'DONOVAN et al., 2019; TRAN et al., 2021), among others, with the common objective of increasing a manufacturing system's efficiency and responsiveness (AHUETT-GARZA & KURFESS, 2018).

By connecting every machine and activity through network sensors to the Internet, a huge amount of data is generated (KOTSIPOULOS et al., 2021). The purpose of ML is precisely to evaluate this generated data and discover potentially useful information about the company and the manufacturing process. It is possible to find in the literature different ML tasks being applied to problems such as pattern detection and classification of health monitoring systems (BERTINO et al., 2021; JUYAL et al., 2021), fault detection and diagnosis (FAN et al., 2021; HE et al., 2021), prediction of future working conditions and remaining useful life (SHE & JIA, 2021; TONG et al., 2021; YAO et al., 2021), among others.

The Fault Diagnosis (FD) is the process of identifying the nature or cause of a failure through the analysis of a set or history of information, to improve manufacturing quality, reduce the cost of product testing and facilitate preventive maintenance of equipment (FAKHR & ELSAYAD, 2012). The data set used in this paper consists of the Steel Plate Fault, available on the UCI Machine Learning Repository (LICHMAN, 2013). This data set has 1941 instances, 27 predictive features, and classifies steel plate failures into 7 different types, such as scratches, stains, dirt, bumps, and other failures. Table 12 shows the number of cases per class present in the dataset.

Table 12 – Number of instances for each class

Class	Number of cases
Pastry	158
Z_Scratch	190
K_Scratch	391
Stains	72
Dirtiness	55
Bumps	402
Other_Faults	673

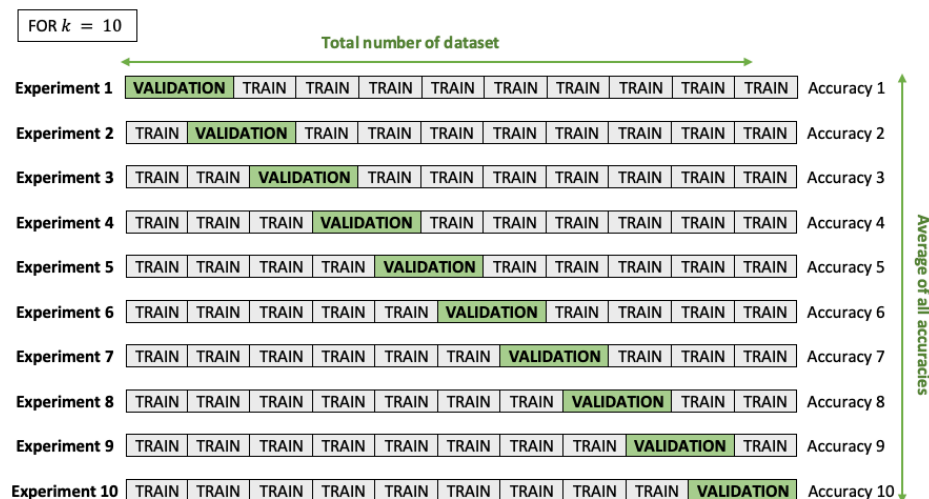
Source: (THE AUTOR, 2021).

The dataset was not separated into training and testing, as the focus of this paper was to analyze the training accuracy. Thus, for the construction of the model, the entire set of instances was used. The validation method used in the construction of the model is the Cross Validation, explained in section 3.2 below.

### 3.2 VALIDATION, EVALUATION CRITERIA AND STATISTICAL TEST

Cross Validation (CV) is a technique used for performance validation of machine learning models. One of the existing methods for applying CV is the  $k$ -fold method, used in this paper. The  $k$ -fold divides the training set randomly into  $k$  subsets with approximately the same number of samples in each. At each iteration, a set formed by  $k - 1$  subsets is used for training and the remaining subset will be used for validation, generating a metric result for evaluation, like accuracy. The accuracy consists of the ratio between correctly classified instances and the total number of instances submitted for classification (Equation 18).

Figure 11 – 10-fold Cross Validation



Source: (THE AUTOR, 2021).

As can be seen in Figure 11, this process ensures that each subset will be used for validation at some point in the model evaluation. At the end of the process, the average of the  $k$  accuracies is calculated with the aim of obtaining the average accuracy (main evaluation criteria used in this paper).

$$Accuracy = \frac{True\ positive(TP)+True\ negative(TN)}{TP+TN+False\ positive\ (FP)+False\ negative\ (FN)} \quad (18)$$

As we calculated the average training accuracy after 30 independent runs, The Wilcoxon signed-rank (WILCOXON, 1945) test is proposed to compare these runs of each feature selection algorithm against the correspondent 30 independent runs of BAJIS. This test is a nonparametric method for the comparison of two paired samples, which the objective is to verify if there are significant differences between two samples.

### 3.3 PARAMETER SETTINGS

The experiments carried out to compare the performance of the proposed BAJIS with the other FS techniques were run on MATLAB R2020b in a macOS Big Sur operating system environment with Intel(R) Core (TM) i5 Dual-Core (1.80 GHz) and 8 Gigabytes (GB) of Random Access Memory (RAM). Table 13 provides a brief description of each FS technique and the respective adjustable parameters and their values (each value was set according to the original paper of the respective method) and Table 14 presents the global parameters and the respective adopted values. For a fair comparison, we used the same V-shaped transfer function (Equation 27) in BCOA, BCSA and BPSO.

Table 13 – Parameter settings of optimization methods for comparison of the BAJIS

Method	Parameters	Values	Brief description
Binary Azure Jay Search (BAJIS)	$c_1$ (cognitive parameter)	1.0	The AJS simulates the search for food and the complex communication system of Azure Jays.
	$c_2$ (social parameter)	1.0	
	$w$ (inertia parameter)	0.5	
	$c_p$ (compassion probability)	0.5	
Binary Coyote Optimization Algorithm (BCOA)	$N_p$ (Number of packs)	5.0	The COA considers the social organization of the coyotes and its adaptation to the environment to solve continuous optimization
	$N_c$ (Number of coyotes)	6.0	
	$P_s$ (Scatter)	$1/D$	

	probability)		problems (THOM DE SOUZA et al., 2020)
	$P_a$ (Association probability)	$1 - P_s / 2$	
	$P_e$ (Eviction probability)	$0.005 \times N_c$	
Binary Crow Search Algorithm (BCSA)	$fl$ (Flight length)	2.0	The CSA is a population-based technique that attempts to simulate intelligent behaviors of crows to find the solution of optimization problems (THOM DE SOUZA et al., 2018)
	$AP$ (Awareness probability)	0.1	
Binary Dragonfly Algorithm (BDA)	$s$ (weight of separation)	0.1	The BDA algorithm simulates the social interaction of dragonflies in searching for food source and avoiding enemies in nature (MAFARJA et al., 2018)
	$a$ (weight of alignment)	0.1	
	$c$ (weight of cohesion)	0.7	
	$f$ (attraction towards the food source)	1.0	
	$e$ (distraction from the enemy)	1.0	
	$w$	0.9 – 0.2	
Binary Particle Swarm Optimization (BPSO)	$c_1$ (Cognitive constant)	2.0	The BPSO is one of the most traditional SI algorithms which applies concepts of social interaction (BANKA & DARA, 2015)
	$c_2$ (Social constant)	2.0	
	$w$ (Local constant)	2.0	
No FS	-	-	No FS uses all of features for the classification

Source: (THE AUTOR, 2021).

Table 14 – Global parameters

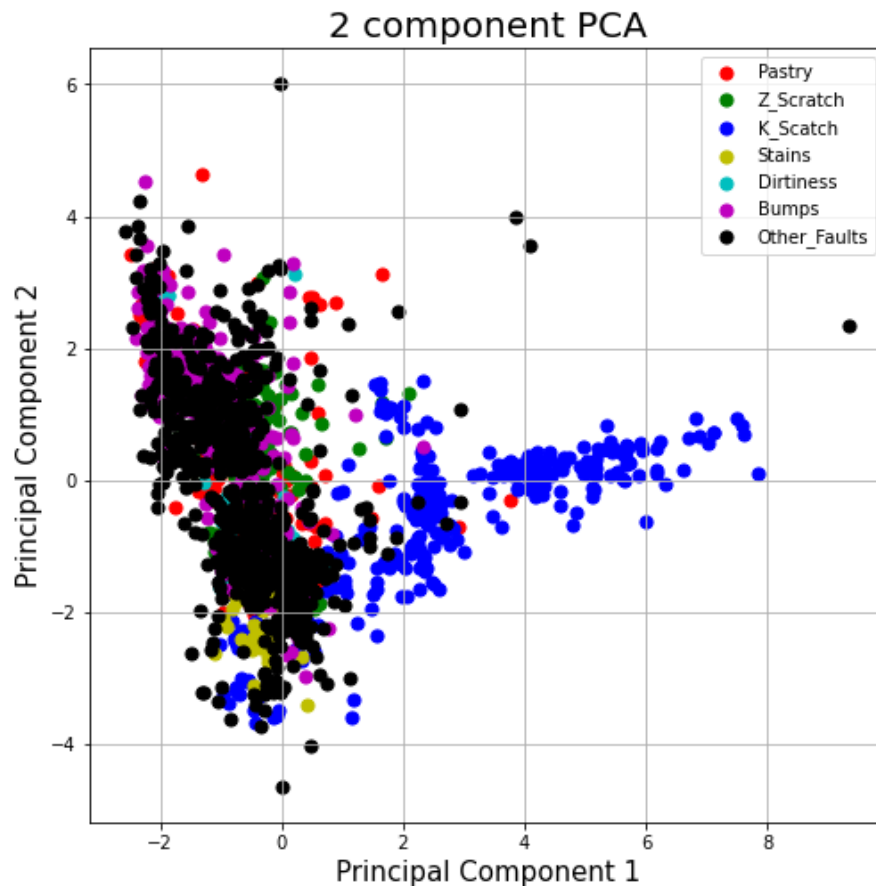
Parameter	Value
Number of generations	100
Number of independent runs	30
Search agents	30
Problem dimension	Number of features in data set
$k$ for cross validation	10
Number of neighbors (KNN)	5

Source: (THE AUTOR, 2021).

#### 4. RESULTS AND DISCUSSIONS

A statistical technique known as Principal Component Analysis (PCA) was applied to the data set, which aims to reduce the dimensionality, that is, to transform a set of data with  $n$  variables (possibly correlated), in a smaller set of  $k$  variables derived from the original set (JOLLIFFE, 2002).

Figure 12 - Point cloud of dataset analyzed after PCA application



Source: (THE AUTOR, 2021).

The  $k$  variables resulting from the PCA application consist of a linear combination of the  $n$  original variables. Furthermore, the first  $k$  variables contain the greatest amount of variation present in the data. The objective of applying the PCA in this paper was to allow the visualization of the point cloud, as shown in Figure 12. Through the Figure 12, it is possible to perceive the complexity of the data, considering the difficulty in identifying a pattern and separate them visually. This fact explains the reasonably low experimental results (in terms of average predictive accuracy) shown in Table 15.

Table 15 shows the BAJIS results (in terms of average accuracy, standard deviation, and statistical significance) when compared to the other proposed techniques (in a wrapper model

based both the NB, the KNN and the RF classifiers). Each column represents an FS strategy (except the last column) and each row the results of those techniques for the 3 different classifiers. The best results are in bold, and the “T” line shows the statistical difference between the BAJIS and the other compared techniques. The symbols “+” and “-” denote statistical difference, the first when the BAJIS obtained lower accuracy than the other compared technique and the second when the BAJIS obtained better accuracy than the other compared technique. Finally, the symbol “≈” denotes statistical equivalence in performance between the two compared techniques.

Table 15 – Results in terms of average training accuracy, standard deviation, and statistical significance.

		BAJS	BCOA	BCSA	BDA	BPSO	No FS
NB	<b>Acc</b>	67.5644	66.8662	66.7416	66.2104	67.4756	60.9112
	<b>Std</b>	0.2923	0.2152	0.1333	0.1719	0.2265	0
	<b>T</b>		-	-	-	≈	-
KNN	<b>Acc</b>	68.8115	70.8548	69.2597	57.0947	72.2916	35.5681
	<b>Std</b>	1.0246	1.5459	1.6201	1.4726	1.0233	0
	<b>T</b>		+	≈	-	+	-
RF	<b>Acc</b>	73.5354	73.9462	72.5174	72.2137	74.5101	70.1471
	<b>Std</b>	1.6338	1.0731	1.3897	0.8705	0.8861	0
	<b>T</b>		≈	-	-	+	-

Source: (THE AUTOR, 2021).

It is possible to see that the proposed algorithm obtained a reasonably good accuracy compared to other techniques. For the NB classifier, the BAJIS obtained better accuracy than BCOA, BCSA, BDA and No FS algorithms, and accuracy statistically equivalent to BPSO. For the KNN classifier, the BAJIS algorithm obtained the third-best accuracy and beat the BDA and No FS algorithms, and present accuracy statistically equivalent to BCSA. As for the RF classifier, the BAJIS obtained second best accuracy and beat the BCSA, BDA and No FS, statistically equivalent to BCOA.

Regarding the number of selected features (as shown in Figure 13), for the NB, KNN and RF classifiers, the algorithms that selected the smallest subsets of features were, respectively, the BDA (but with the second worst average accuracy), BPSO (with the best average accuracy) and BCOA (with the second-best average accuracy). Although BAJIS selects relatively small subsets of features, it selected the second worst subset for the 3 classifiers, tying with the BCOA for the NB classifier, with the BDA for the KNN classifier and tied with the BPSO for the RF classifier.



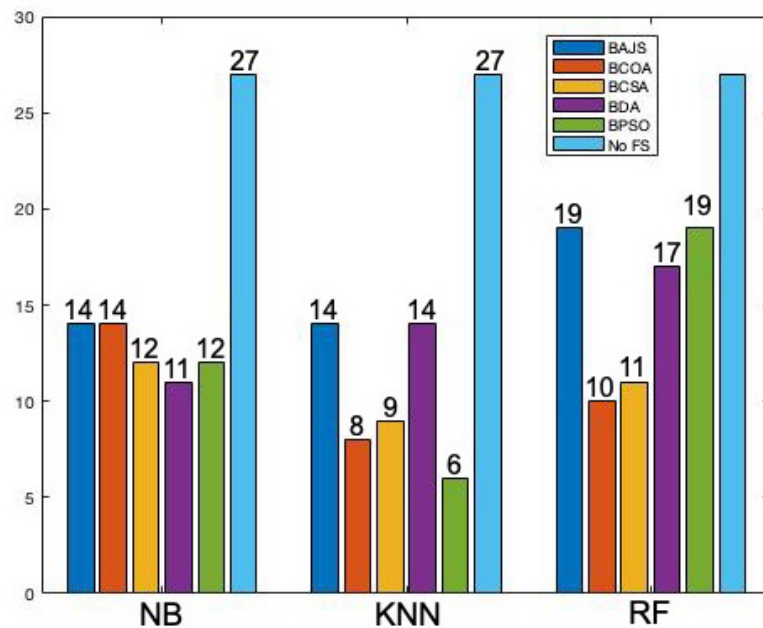
Although the computational cost value of the algorithms is close to each other, as presented in Table 16, the BAJJS obtained computational cost relatively higher than the others for the NB classifier. For the KNN and RF classifiers, the BAJJS obtained a better computational cost than the BCOA and the BDA. In general, all algorithms obtained a high computational cost for the RF classifier.

Table 16 – Computational Cost (expressed in seconds)

Classifier	BAJS	BCOA	BCSA	BDA	BPSO	No FS
NB	14.9240	14.5810	13.0212	12.0944	12.1816	12.9275
KNN	22.4620	23.4570	20.1423	23.6055	22.2376	12.6105
RF	75.5678	76.6789	73.7893	76.9837	75.2323	34.7995

Source: (THE AUTOR, 2021).

Figure 13 – Selected features



Source: (THE AUTOR, 2021).

## 5. CONCLUSIONS AND FUTURE WORKS

Azure Jay Search (AJS) is a new optimization metaheuristic that simulates the behavior of azure jays in the environment (specifically their complex communication system and food search). In principle, AJS was proposed to work with continuous optimization problems, in this paper we propose a Binary AJS (BAJS) for FS problems.

The performance of the BAJJS algorithm was compared, in terms of average training accuracy, standard deviation, computational cost and number of selected features, to other

optimization strategies, when submitted to a Fault Diagnosis dataset in the steel industry. Despite a relatively high computational cost when compared to other techniques and the fact that the dataset is complex, the proposed algorithm achieved relatively good average accuracy and feature subsets with a relatively low number of features.

As future work, we intend to compare the performance of the proposed BAJIS using other transfer functions from both the V-shaped and S-shaped families.

---

## CONCLUSIONS

---

The aim of this dissertation was to propose a new SI metaheuristic and evaluate its performance compared to other state-of-the-art metaheuristics, in terms of average accuracy, standard deviation, dimensionality reduction, computational cost and significance statistical, applied to the FS problem (in wrapper-based model) in a set of Fault Diagnosis data in the steel industry. To achieve this general objective, this dissertation was structured using the multipaper model.

As these SI metaheuristics are initially proposed for solving continuous optimization problems, the first paper deals with exactly that kind of problem. One of the inspirations for the proposed metaheuristic (in addition to the behavior of a bird, almost globally threatened, found in southern Brazil and immediately adjacent areas), was the Crow Search Algorithm (CSA), proposed in 2016. The CSA is an easy-to-implement algorithm, it has a small number of parameters and presents, in general, a good performance in solving optimization problems. An important point to highlight is that the CSA algorithm faces problems when applied to multimodal functions. Thus, the objective of the first paper was to propose two versions of a novel flocking-based modified CSA called Azure Jay Search (AJS) and apply them, in comparison to other SI metaheuristics, to 10 fixed-dimension multimodal and public domain benchmark functions. The algorithms compared to AJS were also algorithms inspired by the behavior of birds in nature, such as the Particle Swarm Optimization (PSO), the Sooty Tern Optimization Algorithm (STOA), Seagull Optimization Algorithm (SOA) and CSA. The two versions of the AJS performed well in terms of average accuracy, and despite a relatively higher computational cost than the other techniques, the first version achieved better accuracy than the

CSA in 3 functions and statistically equivalent performance in 2 functions and the second version also obtained better accuracy than CSA in 3 functions, but statistically equivalent performance in 5 functions. Regarding the other techniques, the first version of AJS won (or achieved statistically equivalent performance) in 9 of the 10 functions and the second version in 10 of the 10 functions.

To reach the general objective of this dissertation, the second paper uses as a binarization strategy a V-shaped family transfer function to propose a binary version of the AJS algorithm, called BAJS. This algorithm, in comparison to other binary SI metaheuristics, were applied to the Fault Diagnosis dataset in the Steel Industry in a wrapper-based model. BAJS performance is evaluated in terms of average training accuracy, standard deviation, computational cost, and number of features selected, and compared to other swarm intelligence metaheuristics, such as Binary Coyote Optimization Algorithm (BCOA), Binary Crow Search Algorithm (BCSA), Binary Dragonfly Algorithm (BDA), and Binary Particle Swarm Optimization (BPSO). Despite a relatively higher computational cost, the BAJS algorithm find good solutions in terms of average training accuracy and subsets with a relatively small number of features. For the Naive Bayes (NB) and Random Forest (RF) classifiers, the BAJS obtained, respectively, the best and the second-best average training accuracy (statistically equivalent to BPSO and BCOA). For the KNN classifier, the BAJS algorithm obtained the third-best accuracy and beat the BDA and No FS algorithms, and present accuracy statistically equivalent to BCSA. In addition, although BAJS selects relatively small subsets of features, it selected the second worst subset for the 3 classifiers, tying with the BCOA for the NB classifier, with the BDA for the KNN classifier and tied with the BPSO for the RF classifier.

## REFERENCES

---

ABDOLLAHZADEH, B.; BARSHANDEH, S.; JAVADI, H.; EPICOCO, N. An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem. **Engineering with Computers**, , n. 0123456789, 2021. Springer London. Disponível em: <<https://doi.org/10.1007/s00366-021-01470-z>>.

ABDOLLAHZADEH, B.; GHAREHCHOPOGH, FARHAD SOLEIMANIAN; MIRJALILI, S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. **International Journal of Intelligent Systems**, v. 36, n. 10, p. 5887–5958, 2021.

ABDOLLAHZADEH, B.; GHAREHCHOPOGH, FARHAD. SOLEIMANIAN; MIRJALILI, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. **Computers & Industrial Engineering**, v. 158, p. 107408, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.cie.2021.107408>>.

AHUETT-GARZA, H.; KURFESS, T. A brief discussion on the trends of habilitating technologies for Industry 4.0 and Smart manufacturing. **Manufacturing Letters**, v. 15, p. 60–63, 2018. Society of Manufacturing Engineers (SME). Disponível em: <<https://doi.org/10.1016/j.mfglet.2018.02.011>>.

AL-GAPHARI, G. H.; AL-AMRY, R.; AL-NUZAILI, A. S. Discrete crow-inspired algorithms for traveling salesman problem. **Engineering Applications of Artificial Intelligence**, v. 97, p. 104006, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.engappai.2020.104006>>.

AL-RIFAIE, M. M. Exploration and exploitation zones in a minimalist swarm optimiser.

**Entropy**, v. 23, n. 8, p. 1–38, 2021.

ALAZZAM, H.; SHARIEH, A.; SABRI, K. E. A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer. **Expert Systems with Applications**, v. 148, 2020.

ALHAMIDI, M. R.; JATMIKO, W. Optimal feature aggregation and combination for two-dimensional ensemble feature selection. **Information (Switzerland)**, v. 11, n. 1, p. 1–16, 2020.

ALSALIBI, B.; ABUALIGAH, L.; KHADER, A. T. A novel bat algorithm with dynamic membrane structure for optimization problems. **Applied Intelligence**, v. 51, n. 4, p. 1992–2017, 2021. *Applied Intelligence*.

ALVAREZ-ALVARADO, M. S.; ALBAN-CHACÓN, F. E.; LAMILLA-RUBIO, E. A.; RODRÍGUEZ-GALLEGOS, C. D.; VELÁSQUEZ, W. Three novel quantum-inspired swarm optimization algorithms using different bounded potential fields. **Scientific Reports**, v. 11, n. 1, p. 1–22, 2021. Nature Publishing Group UK. Disponível em: <<https://doi.org/10.1038/s41598-021-90847-7>>.

AMINI, F.; HU, G. A two-layer feature selection method using Genetic Algorithm and Elastic Net. **Expert Systems with Applications**, v. 166, p. 114072, 2021. Disponível em: <<https://doi.org/10.1016/j.eswa.2020.114072>>.

ANJOS, L. O ciclo anual de *Cyanocorax caeruleus* em floresta de Araucaria (Passeriformes: Corvidae), Ararajuba, v. 2, p. 19-23, 1991.

ANJOS, L.; VIELLIARD, J. M. E. Repertoire of the acoustic communication of the Azure Jay *Cyanocorax Caeruleus* (Vieillot) (Aves, Corvidae). **Revista Brasileira de Zoologia**, v. 10, n. 4, p. 657–664, 1993.

ASHIMA, R.; HALEEM, A.; BAHL, S.; et al. Automation and manufacturing of smart materials in additive manufacturing technologies using Internet of Things towards the adoption of industry 4.0. **Materials Today: Proceedings**, v. 45, p. 5081–5088, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.matpr.2021.01.583>>.

ASKARZADEH, A. A novel metaheuristic method for solving constrained engineering optimization problems : Crow search algorithm. **Computers and Structures**, v. 169, p. 1- 12M,

2016. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.compstruc.2016.03.001>>.

AZIZI, M. Atomic orbital search: A novel metaheuristic algorithm. **Applied Mathematical Modelling**, v. 93, p. 657–683, 2021. Elsevier Inc.

BANKA, H.; DARA, S. A Hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. **Pattern Recognition Letters**, v. 52, p. 94–100, 2015. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.patrec.2014.10.007>>.

BIRDLIFE INTERNATIONAL. Species factsheet: *Cyanocorax coeruleus*, 2021. Disponível em: <http://www.birdlife.org> on 01/09/2021.

BERGES, I.; RAMÍREZ-DURÁN, V. J.; ILLARRAMENDI, A. A Semantic Approach for Big Data Exploration in Industry 4.0. **Big Data Research**, v. 25, p. 100222, 2021. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.bdr.2021.100222>>.

BERTINO, E.; JAHANSHAHI, M. R.; SINGLA, A.; WU, R.-T. Intelligent IoT systems for civil infrastructure health monitoring: a research roadmap. **Discover Internet of Things**, v. 1, n. 1, 2021.

BOGAR, E.; BEYHAN, S. Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems. **Applied Soft Computing Journal**, v. 95, p. 106503, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2020.106503>>.

BRADY, S. Azure Jay (*Cyanocorax caeruleus*), version 1.0. In Neotropical Birds Online (T. S. Schulenberg, Editor). Cornell Lab of Ornithology, Ithaca, NY, USA, 2010. Disponível em: <https://doi.org/10.2173/nb.azujay1.01>.

BRAIK, M. S. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. **Expert Systems with Applications**, v. 174, p. 114685, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2021.114685>>.

BRANDÃO, J. A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. **European Journal of Operational Research**, v. 284, p. 559–571,

2020.

BRIK, B.; BETTAYEB, B.; SAHNOUN, M.; DUVAL, F. Towards predicting system disruption in industry 4.0: Machine learning-based approach. **Procedia Computer Science**, v. 151, n. 2018, p. 667–674, 2019. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.procs.2019.04.089>>.

CARRASCO, J.; GARCÍA, S.; RUEDA, M. M.; DAS, S.; HERRERA, F. Recent Trends in the Use of Statistical Tests for Comparing Swarm and Evolutionary Computing Algorithms: Practical Guidelines and a Critical Review. **Swarm and Evolutionary Computation**, v. 54, 2020.

CAVAZZUTI, M. **Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics**. Springer Berlin Heidelberg, 2013.

CHAUDHURI, A.; SAHU, T. P. A hybrid feature selection method based on Binary Jaya algorithm for micro-array data classification. **Computers and Electrical Engineering**, v. 90, p. 106963, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.compeleceng.2020.106963>>.

CHEN, M.-R.; HUANG, Y.-Y.; ZENG, G.-Q.; LU, K.-D.; YANG, L.-Q. An improved bat algorithm hybridized with extremal optimization and Boltzmann selection. **Expert Systems with Applications**, v. 175, p. 114812, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2021.114812>>.

CHENG, Z.; SONG, H.; WANG, J.; et al. Hybrid firefly algorithm with grouping attraction for constrained optimization problem. **Knowledge-Based Systems**, v. 220, p. 106937, 2021. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2021.106937>>.

CHOU, J.-S.; TRUONG, D.-N. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. **Applied Mathematics and Computation**, v. 389, p. 125535, 2021. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.amc.2020.125535>>.

COELHO, L. S.; RICHTER, C.; MARIANI, V. C.; ASKARZADEH, A. Modified Crow Search Approach Applied to Electromagnetic Optimization. **IEEE CEFC 2016 - 17th Biennial Conference on Electromagnetic Field Computation**, v. 59, n. 2, p. 2013, 2017.



CRAWFORD, B.; SOTO, R.; ASTORGA, G.; et al. Putting continuous metaheuristics to work in binary search spaces. **Complexity**, v. 2017, p. 1–19, 2017.

CUI, X.; LI, Y.; FAN, J.; WANG, T. A novel filter feature selection algorithm based on relief. **Applied Intelligence**, 2021. Applied Intelligence.

DENIL, J.; JUKSS, M.; VERBRUGGE, C.; VANGHELUWE, H. Search-Based Model Optimization Using Model Transformations. In: D. Amyot; P. Fonseca I Casas; G. Mussbacher (Orgs.); Proceedings of the 8th International Conference on System Analysis and Modeling: Models and Reusability (SAM 2014). **Anais...** . v. 8769, p.80–85, 2014. Valencia, Spain. Disponível em: <<http://link.springer.com/10.1007/978-3-319-11743-0>>.

DHIMAN, G. SSC: A hybrid nature-inspired meta-heuristic optimization algorithm for engineering applications. **Knowledge-Based Systems**, v. 222, p. 106926, 2021. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2021.106926>>.

DHIMAN, G.; GARG, M.; NAGAR, A.; KUMAR, V.; DEHGHANI, M. A novel algorithm for global optimization: Rat Swarm Optimizer. **Journal of Ambient Intelligence and Humanized Computing**, v. 12, n. 8, p. 8457–8482, 2021. Springer Berlin Heidelberg. Disponível em: <<https://doi.org/10.1007/s12652-020-02580-0>>.

DHIMAN, G.; KAUR, A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. **Engineering Applications of Artificial Intelligence**, v. 82, p. 148–174, 2019. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.engappai.2019.03.021>>.

DHIMAN, G.; KUMAR, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. **Knowledge-Based Systems**, v. 165, p. 169–196, 2019. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2018.11.024>>.

DIGALAKIS, J. G.; MARGARITIS, K. G. On benchmarking functions for genetic algorithms. **International Journal of Computer Mathematics**, v. 77, n. 4, p. 481–506, 2001.

DOKEROGLU, T.; SEVINC, E.; KUCUKYILMAZ, T.; COSAR, A. A survey on new generation metaheuristic algorithms. **Computers & Industrial Engineering**, v. 137, p. 106040, 2019. Elsevier. Disponível em: <<https://doi.org/10.1016/j.cie.2019.106040>>.

DONG, J.; WANG, Z.; MO, J. A phase angle-modulated bat algorithm with application to

antenna topology optimization. **Applied Sciences (Switzerland)**, v. 11, n. 5, p. 1–20, 2021.

ELAZIZ, M. A.; EWEES, A. A.; IBRAHIM, R. A.; LU, S. Opposition-based moth-flame optimization improved by differential evolution for feature selection. **Mathematics and Computers in Simulation**, v. 168, p. 48–75, 2020. Elsevier B.V. Disponible em: <<https://doi.org/10.1016/j.matcom.2019.06.017>>.

EMAMI, H.; SHARIFI, A. A. A novel bio-inspired optimization algorithm for solving peak-to-average power ratio problem in DC-biased optical systems. **Optical Fiber Technology**, v. 60, p. 102383, 2020. Elsevier Inc. Disponible em: <<https://doi.org/10.1016/j.yofte.2020.102383>>.

ENGELBRECHT, A. P.; GROBLER, J.; LANGEVELD, J. Set based particle swarm optimization for the feature selection problem. **Engineering Applications of Artificial Intelligence**, v. 85, n. July, p. 324–336, 2019. Elsevier Ltd. Disponible em: <<https://doi.org/10.1016/j.engappai.2019.06.008>>.

FAKHR, M.; ELSAYAD, A. M. Steel plates faults diagnosis with data mining models. **Journal of Computer Science**, v. 8, n. 4, p. 506–514, 2012.

FAN, C.; LIU, X.; XUE, P.; WANG, J. Statistical characterization of semi-supervised neural networks for fault detection and diagnosis of air handling units. **Energy and Buildings**, v. 234, p. 110733, 2021. Elsevier B.V. Disponible em: <<https://doi.org/10.1016/j.enbuild.2021.110733>>.

FAUSTO, F.; REYNA-ORTA, A.; CUEVAS, E.; ANDRADE, Á. G.; PEREZ-CISNEROS, M. From ants to whales: metaheuristics for all tastes. **Artificial Intelligence Review**, v. 53, p. 753–810, 2020.

GUNASUNDARI, S.; JANAKIRAMAN, S.; MEENAMBAL, S. Velocity Bounded Boolean Particle Swarm Optimization for improved feature selection in liver and kidney disease diagnosis. **Expert Systems with Applications**, v. 56, p. 28–47, 2016. Elsevier Ltd. Disponible em: <<http://dx.doi.org/10.1016/j.eswa.2016.02.042>>.

GUNASUNDARI, S.; JANAKIRAMAN, S.; MEENAMBAL, S. Multiswarm heterogeneous binary PSO using win-win approach for improved feature selection in liver and kidney disease diagnosis. **Computerized Medical Imaging and Graphics**, v. 70, p. 135–154, 2018. Elsevier

Ltd. Disponível em: <<https://doi.org/10.1016/j.compmedimag.2018.10.003>>.

HAMMOURI, A. I.; MAFARJA, M.; AL-BETAR, M. A.; AWADALLAH, M. A.; ABU-DOUSH, I. An improved Dragonfly Algorithm for feature selection. **Knowledge-Based Systems**, v. 203, p. 106131, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2020.106131>>.

HASHIM, F. A.; HOUSSEIN, E. H.; HUSSAIN, K.; MABROUK, M. S.; AL-ATABANY, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. **Mathematics and Computers in Simulation**, v. 192, p. 84–110, 2022. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.matcom.2021.08.013>>.

HE, Y.; NIE, B.; ZHANG, J.; KUMAR, P. M.; MUTHU, B. A. Fault Detection and Diagnosis of Cyber-Physical System Using the Computer Vision and Image Processing. **Wireless Personal Communications**, 2021. Springer US. Disponível em: <<https://doi.org/10.1007/s11277-021-08774-9>>.

HEGAZY, A. E.; MAKHLOUF, M. A.; EL-TAWEL, G. S. Improved salp swarm algorithm for feature selection. **Journal of King Saud University - Computer and Information Sciences**, v. 32, n. 3, p. 335–344, 2020. King Saud University. Disponível em: <<https://doi.org/10.1016/j.jksuci.2018.06.003>>.

HICHEM, H.; ELKAMEL, M.; RAFIK, M.; MESAAOUD, M. T.; OUAHIBA, C. A new binary grasshopper optimization algorithm for feature selection problem. **Journal of King Saud University - Computer and Information Sciences**, , n. In press. The Authors. Disponível em: <<https://doi.org/10.1016/j.jksuci.2019.11.007>>.

HORNISCHER, H.; HERMINGHAUS, S.; MAZZA, M. G. Structural transition in the collective behavior of cognitive agents. **Scientific Reports**, v. 9, p. 1–11, 2019. Springer US. Disponível em: <<http://dx.doi.org/10.1038/s41598-019-48638-8>>.

HU, P.; PAN, J.-S.; CHU, S.-C. Improved Binary Grey Wolf Optimizer and Its application for feature selection. **Knowledge-Based Systems**, v. 195, p. 105746, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2020.105746>>.

HUSSAIN, K.; MOHD SALLEH, M. N.; CHENG, S.; SHI, Y. Metaheuristic research: a comprehensive survey. **Artificial Intelligence Review**, v. 52, n. 4, p. 2191–2233, 2019.

Springer Netherlands. Disponível em: <<https://doi.org/10.1007/s10462-017-9605-z>>.

ISLAM, J.; RAHAMAN, M. S. A.; VASANT, P. M.; et al. A modified niching crow search approach to well placement optimization. **Energies**, v. 14, p. 1–33, 2021.

ISLAM, J.; VASANT, P. M.; NEGASH, B. M.; WATADA, J. A modified crow search algorithm with niching technique for numerical optimization. **2019 IEEE Student Conference on Research and Development, SCORED 2019**, , n. October, p. 170–175, 2019. IEEE.

JAVED, M. A.; MURAM, F. U.; HANSSON, H.; PUNNEKKAT, S.; THANE, H. Towards dynamic safety assurance for Industry 4.0. **Journal of Systems Architecture**, v. 114, n. October 2020, 2021. Elsevier B.V.

JIA, Y. H.; CHEN, W. N.; GU, T.; et al. A Dynamic Logistic Dispatching System with Set-Based Particle Swarm Optimization. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 48, n. 9, p. 1607–1621, 2018. IEEE.

JOLLIFFE, I. T. **Principal Component Analysis**. 2002.

JUYAL, S.; SHARMA, S.; SHANKAR SHUKLA, A. Smart skin health monitoring using AI-enabled cloud-based IoT. **Materials Today: Proceedings**, v. 46, p. 10539–10545, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.matpr.2021.01.074>>.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. **Journal of Global Optimization**, v. 39, n. 3, p. 459–471, 2007.

KAUR, S.; AWASTHI, L. K.; SANGAL, A. L.; DHIMAN, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. **Engineering Applications of Artificial Intelligence**, v. 90, p. 103541, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.engappai.2020.103541>>.

KENNEDY, J.; EBERHART, R. Particle Swarm Optimization. Proceedings of International Conference on Neural Networks (ICNN'95). **Anais...** . p.1942–1948, 1995.

KHISHE, M.; MOSAVI, M. R. Chimp optimization algorithm. **Expert Systems with Applications**, v. 149, p. 113338, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2020.113338>>.

KILIÇ, F.; KAYA, Y.; YILDIRIM, S. A novel multi population based particle swarm optimization for feature selection. **Knowledge-Based Systems**, v. 219, p. 106894, 2021. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2021.106894>>.

KOTSIPOPOULOS, T.; SARIGIANNIDIS, P.; IOANNIDIS, D.; TZOVARAS, D. Machine Learning and Deep Learning in smart manufacturing: The Smart Grid paradigm. **Computer Science Review**, v. 40, p. 100341, 2021. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.cosrev.2020.100341>>.

KUMAR, R. **Research Methodology: a step-by-step guide for beginners**. 3rd Editio ed. SAGE publications, 2011.

LI, C.; LI, J.; CHEN, H.; JIN, M.; REN, H. Enhanced Harris hawks optimization with multi-strategy for global optimization tasks. **Expert Systems with Applications**, v. 185, p. 115499, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2021.115499>>.

LI, J.; HAN, YUN-QI; DUAN, P.; et al. Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems. **Journal of Cleaner Production**, v. 250, p. 119464, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.jclepro.2019.119464>>.

LICHMAN, M. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Science, USA, 2013.

LIU, W.; WANG, J. Recursive elimination-election algorithms for wrapper feature selection. **Applied Soft Computing**, v. 113, p. 107956, 2021. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2021.107956>>.

LIU, Z. An Analysis of Particle Swarm Optimization of Multi-objective Knapsack Problem. Proceedings of the 9th International Conference on Industrial Technology and Management (ICITM). **Anais...** p.302–306, 2020. IEEE.

LÓPEZ-SANTILLÁN, R.; MONTES-Y-GÓMEZ, M.; GONZÁLEZ-GURROLA, L. C.; RAMÍREZ-ALONSO, G.; PRIETO-ORDAZ, O. Richer Document Embeddings for Author Profiling tasks based on a heuristic search. **Information Processing and Management**, v. 57, n. 4, p. 102227, 2020. Elsevier. Disponível em: <<https://doi.org/10.1016/j.ipm.2020.102227>>.

LU, J. J.; ZHANG, M. **Heuristic Search**. 2013<sup>o</sup> ed. New York, NY, USA: Springer, 2013.

LU, X. LI; HE, G. QPSO algorithm based on Lévy flight and its application in fuzzy portfolio. **Applied Soft Computing**, v. 99, p. 106894, 2021. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2020.106894>>.

MACEDO, C. A.; THOM DE SOUZA, R. C.; GUEDES FILHO, O. Computer Vision and Feature Selection with BBIL for Recognition of Agricultural Management Zones. 5rd International Conference on Engineering Optimization (EngOpt). **Anais...** . p.19–23, 2016.

MADGE, S.; BURN, H. *Crows and Jays*, Princeton University Press, 1999.

MAFARJA, M.; ALJARAH, I.; HEIDARI, A. A.; et al. Binary dragonfly optimization for feature selection using time-varying transfer functions. **Knowledge-Based Systems**, v. 161, n. July, p. 185–204, 2018. Elsevier. Disponível em: <<https://doi.org/10.1016/j.knosys.2018.08.003>>.

MAFARJA, M.; HEIDARI, A. A.; HABIB, M.; et al. Augmented whale feature selection for IoT attacks: Structure, analysis and applications. **Future Generation Computer Systems**, v. 112, p. 18–40, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.future.2020.05.020>>.

MARIE-SAINTE, S. L.; ALALYANI, N. Firefly Algorithm based Feature Selection for Arabic Text Classification. **Journal of King Saud University - Computer and Information Sciences**, v. 32, n. 3, p. 320–328, 2020. King Saud University. Disponível em: <<https://doi.org/10.1016/j.jksuci.2018.06.004>>.

MARIYAPRINCY, A.; SAMIAPPAN, D. Analysis of Internet of Things enabled by artificial intelligence for automatic based model in educational institution. **Materials Today: Proceedings**, , n. In press, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.matpr.2020.11.791>>.

MARTINEZ-RIOS, F.; MURILLO-SUAREZ, A. A multiprocess Salp swarm optimization with a heuristic based on crossing partial solutions. *Procedia Computer Science*. **Anais...** . v. 179, p.440–447, 2021a. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.procs.2021.01.027>>.

MARTINEZ-RIOS, F.; MURILLO-SUAREZ, A. Multi-threaded Spotted Hyena Optimizer with thread-crossing techniques. *Procedia Computer Science*. **Anais...** . v. 179, p.432–439, 2021b. Elsevier B.V. Disponible em: <<https://doi.org/10.1016/j.procs.2021.01.026>>.

MENG, O. K.; PAULINE, O.; KIONG, S. C. A carnivorous plant algorithm for solving global optimization problems. **Applied Soft Computing Journal**, v. 98, p. 106833, 2021. Elsevier B.V. Disponible em: <<https://doi.org/10.1016/j.asoc.2020.106833>>.

MIRJALILI, S.; LEWIS, A. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. **Swarm and Evolutionary Computation**, v. 9, p. 1–14, 2013. Elsevier. Disponible em: <<http://dx.doi.org/10.1016/j.swevo.2012.09.002>>.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey Wolf Optimizer. **Advances in Engineering Software**, v. 69, p. 46–61, 2014. Elsevier Ltd. Disponible em: <<http://dx.doi.org/10.1016/j.advengsoft.2013.12.007>>.

MNICH, K.; RUDNICKI, W. R. All-relevant feature selection using multidimensional filters with exhaustive search. **Information Sciences**, v. 524, p. 277–297, 2020. Elsevier Inc.

MOHAMMADI, F.; ABDI, H. A modified crow search algorithm (MCSA) for solving economic load dispatch problem. **Applied Soft Computing Journal**, v. 71, p. 51–65, 2018. Elsevier B.V. Disponible em: <<https://doi.org/10.1016/j.asoc.2018.06.040>>.

MOLINA, D.; POYATOS, J.; SER, J. DEL; et al. Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration versus Algorithmic Behavior, Critical Analysis and Recommendations. **Cognitive Computation**, v. 12, n. 5, p. 897–939, 2020.

MORALES-CASTAÑEDA, B.; ZALDÍVAR, D.; CUEVAS, E.; FAUSTO, F.; RODRÍGUEZ, A. A better balance in metaheuristic algorithms: Does it exist? **Swarm and Evolutionary Computation**, v. 54, n. February, 2020.

MUÑOZ, M. A.; LÓPEZ, J. A.; CAICEDO, E. An artificial beehive algorithm for continuous optimization. **International Journal of Intelligent Systems**, v. 24, n. 11, p. 1080–1093, 2009.

NARUEI, I.; KEYNIA, F. A new optimization method based on COOT bird natural life model. **Expert Systems with Applications**, v. 183, n. February 2020, p. 115352, 2021. Elsevier Ltd. Disponible em: <<https://doi.org/10.1016/j.eswa.2021.115352>>.

NEAL, A. D.; SHARPE, R. G.; VAN LOPIK, K.; et al. The potential of industry 4.0 Cyber Physical System to improve quality assurance: An automotive case study for wash monitoring of returnable transit items. **CIRP Journal of Manufacturing Science and Technology**, v. 32, p. 461–475, 2021. CIRP. Disponível em: <<https://doi.org/10.1016/j.cirpj.2020.07.002>>.

NEGGAZ, N.; EWEES, A. A.; ELAZIZ, M. A.; MAFARJA, M. Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. **Expert Systems with Applications**, v. 145, p. 113103, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2019.113103>>.

NEMATZADEH, H.; ENAYATIFAR, R.; MAHMUD, M.; AKBARI, E. Frequency based feature selection method using whale algorithm. **Genomics**, v. 111, n. 6, p. 1946–1955, 2019. Elsevier. Disponível em: <<https://doi.org/10.1016/j.ygeno.2019.01.006>>.

NOCEDAL, J.; WRIGHT, S. J. **Numerical optimization**. 2006.

O'DONOVAN, P.; GALLAGHER, C.; LEAHY, K.; O'SULLIVAN, D. T. J. A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real-time embedded machine learning engineering applications. **Computers in Industry**, v. 110, p. 12–35, 2019. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.compind.2019.04.016>>.

OUADFEL, S.; ABD ELAZIZ, M. Efficient High-Dimension Feature Selection Based on Enhanced Equilibrium Optimizer. **Expert Systems with Applications**, v. 187, p. 115882, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2021.115882>>.

OZBEY, N.; YEROGLU, C.; ALAGOZ, B. B.; et al. 2DOF multi-objective optimal tuning of disturbance reject fractional order PIDA controllers according to improved consensus oriented random search method. **Journal of Advanced Research**, v. 25, p. 159–170, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.jare.2020.03.008>>.

PANDIRI, V.; SINGH, A.; ROSSI, A. Two hybrid metaheuristic approaches for the covering salesman problem. **Neural Computing and Applications**, v. 32, p. 15643–15663, 2020.

ROSTAMI, M.; FOROUZANDEH, S.; BERAHMAND, K.; SOLTANI, M. Integration of multi-objective PSO based feature selection and node centrality for medical datasets. **Genomics**, v. 112, n. 6, p. 4370–4384, 2020. Elsevier. Disponível em: <<https://doi.org/10.1016/j.ygeno.2020.07.027>>.



SAAFAN, M. M.; EL-GENDY, E. M. IWOSSA: An improved whale optimization salp swarm algorithm for solving optimization problems. **Expert Systems with Applications**, v. 176, p. 114901, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2021.114901>>.

SABRI-LAGHAIE, K.; KARIMI-NASAB, M. Random search algorithms for redundancy allocation problem of a queuing system with maintenance considerations. **Reliability Engineering and System Safety**, v. 185, p. 144–162, 2019. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.ress.2018.12.010>>.

SELVAKUMAR, B.; MUNEESWARAN, K. Firefly algorithm based feature selection for network intrusion detection. **Computers and Security**, v. 81, p. 148–155, 2019. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.cose.2018.11.005>>.

SHAHEEN, A. M.; EL-SEHIEMY, R. A.; ALHARTHI, M. M.; GHONEIM, S. S. M.; GINIDI, A. R. Multi-objective jellyfish search optimizer for efficient power system operation based on multi-dimensional OPF framework. **Energy**, v. 237, p. 121478, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.energy.2021.121478>>.

SHE, D.; JIA, M. A BiGRU method for remaining useful life prediction of machinery. **Measurement: Journal of the International Measurement Confederation**, v. 167, n. August 2019, p. 108277, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.measurement.2020.108277>>.

SLEZKIN, A. O.; HODASHINSKY, I. A.; SHELUPANOV, A. A. Binarization of the Swallow Swarm Optimization for Feature Selection. **Programming and Computer Software**, v. 47, n. 5, p. 374–388, 2021.

SONG, M.; LI, J.; HAN, YUN-QI; et al. Metaheuristics for solving the vehicle routing problem with the time windows and energy consumption in cold chain logistics. **Applied Soft Computing Journal**, v. 95, p. 106561, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2020.106561>>.

SRINIVASAN, V.; BOOPATHI, C. S.; SRIDHAR, R. A new meerkat optimization algorithm based maximum power point tracking for partially shaded photovoltaic system. **Ain Shams Engineering Journal**, 2021. THE AUTHORS. Disponível em: <<https://doi.org/10.1016/j.asej.2021.03.017>>.

STRASSER, S.; GOODMAN, R.; SHEPPARD, J.; BUTCHER, S. A new discrete Particle Swarm Optimization algorithm. **GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference**, p. 53–60, 2016.

TALBI, E.-G. **Metaheuristics: From Design to Implementation**. Hoboken, New Jersey: John Wiley and Sons Inc., 2014.

TARAMASCO, C.; CRAWFORD, B.; SOTO, R.; CORTÉS-TORO, E. M.; OLIVARES, R. A new metaheuristic based on vapor-liquid equilibrium for solving a new patient bed assignment problem. **Expert Systems with Applications**, v. 158, p. 113506, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2020.113506>>.

TARKHANEH, O.; NGUYEN, T. T.; MAZAHERI, S. A novel wrapper-based feature subset selection method using modified binary differential evolution algorithm. **Information Sciences**, v. 565, p. 278–305, 2021. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.ins.2021.02.061>>.

THOM DE SOUZA, R. C. **Previsão de Séries Temporais utilizando Rede Neural treinada por Filtro de Kalman e Evolução Diferencial**, 2008. Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós- Graduação em Engenharia de Produção e Sistemas.

THOM DE SOUZA, R. C.; COELHO, L. S.; MACEDO, C. A.; PIEREZAN, J. A V-Shaped Binary Crow Search Algorithm for Feature Selection. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2018). **Anais...** . p.1–8, 2018. Rio de Janeiro, Brasil: IEEE.

THOM DE SOUZA, R. C.; MACEDO, C. A.; COELHO, L. S.; PIEREZAN, J.; MARIANI, V. C. Binary coyote optimization algorithm for feature selection. **Pattern Recognition**, v. 107, p. 107470, 2020. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.patcog.2020.107470>>.

TILAHUN, S. L.; NGNOTCHOUYE, J. M. T. Firefly Algorithm for Discrete Optimization Problems: A Survey. **KSCE Journal of Civil Engineering**, v. 21, n. 2, p. 535–545, 2017.

TONG, Z.; MIAO, J.; TONG, S.; LU, Y. Early prediction of remaining useful life for Lithium-ion batteries based on a hybrid machine learning method. **Journal of Cleaner Production**, v. 317, n. July, p. 128265, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.jclepro.2021.128265>>.

TORABI, S.; SAFI-ESFAHANI, F. Improved Raven Roosting Optimization algorithm (IRRO). **Swarm and Evolutionary Computation**, v. 40, p. 144–154, 2018.

TRAGANOS, K.; GREFEN, P.; VANDERFEESTEN, I.; et al. The HORSE framework: A reference architecture for cyber-physical systems in hybrid smart manufacturing. **Journal of Manufacturing Systems**, v. 61, n. September, p. 461–494, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.jmsy.2021.09.003>>.

TRAN, M. Q.; ELSISI, M.; MAHMOUD, K.; et al. Experimental Setup for Online Fault Diagnosis of Induction Machines via Promising IoT and Machine Learning: Towards Industry 4.0 Empowerment. **IEEE Access**, v. 9, p. 115429–115441, 2021. IEEE.

TUBISHAT, M.; IDRIS, N.; SHUIB, L.; ABUSHARIAH, M. A. M.; MIRJALILI, S. Improved Salp Swarm Algorithm based on opposition based learning and novel local search algorithm for feature selection. **Expert Systems with Applications**, v. 145, p. 113122, 2020. Elsevier Ltd.

TUBISHAT, M.; JA'AFAR, S.; ALSWAITTI, M.; et al. Dynamic Salp swarm algorithm for feature selection. **Expert Systems with Applications**, v. 164, p. 113873, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2020.113873>>.

WANG, D.; CHEN, H.; LI, T.; WAN, J.; HUANG, Y. A novel quantum grasshopper optimization algorithm for feature selection. **International Journal of Approximate Reasoning**, v. 127, p. 33–53, 2020. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.ijar.2020.08.010>>.

WANG, H.; WANG, W.; ZHOU, X.; et al. Artificial bee colony algorithm based on knowledge fusion. **Complex & Intelligent Systems**, v. 7, n. 3, p. 1139–1152, 2021. Springer International Publishing. Disponível em: <<https://doi.org/10.1007/s40747-020-00171-2>>.

WANG, R.; WANG, C.; LIU, G. A novel graph clustering method with a greedy heuristic search algorithm for mining protein complexes from dynamic and static PPI networks. **Information Sciences**, v. 522, p. 275–298, 2020. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.ins.2020.02.063>>.

WANG, X. HAN; ZHANG, Y.; SUN, X. YAN; WANG, Y. LI; DU, C. HE. Multi-objective feature selection based on artificial bee colony: An acceleration approach with variable sample

size. **Applied Soft Computing Journal**, v. 88, p. 106041, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2019.106041>>.

WANG, Y.; CAI, S.; CHEN, J.; YIN, M. SCCWalk: An efficient local search algorithm and its improvements for maximum weight clique problem. **Artificial Intelligence**, v. 280, p. 103230, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.artint.2019.103230>>. .

WOLPERT, D. H.; MACREADY, W. G. No Free Lunch Theorems for Optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, 1997.

XU, X.; HU, Z.; SU, Q.; LI, Y.; DAI, J. Multivariable grey prediction evolution algorithm: A new metaheuristic. **Applied Soft Computing Journal**, v. 89, p. 106086, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2020.106086>>.

XUE, J.; SHEN, B. A novel swarm intelligence optimization approach: sparrow search algorithm. **Systems Science and Control Engineering**, v. 8, n. 1, p. 22–34, 2020.

XUE, Y.; TANG, T.; PANG, W.; LIU, A. X. Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers. **Applied Soft Computing Journal**, v. 88, p. 106031, 2020. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.asoc.2019.106031>>.

YANG, E. O. X. **Applied Optimization and Swarm Intelligence**.

YANG, X. S. A new metaheuristic Bat-inspired Algorithm. **Studies in Computational Intelligence**, v. 284, p. 65–74, 2010.

YANG, X. S. Swarm intelligence based algorithms: A critical analysis. **Evolutionary Intelligence**, v. 7, n. 1, p. 17–28, 2014.

YANG, X. S.; DEB, S.; FONG, S.; HE, X.; ZHAO, Y. X. From Swarm Intelligence to Metaheuristics: Nature-Inspired Optimization Algorithms. **Computer**, v. 49, n. 9, p. 52–59, 2016.

YAO, D.; LI, B.; LIU, H.; YANG, J.; JIA, L. Remaining useful life prediction of roller bearings based on improved 1D-CNN and simple recurrent unit. **Measurement: Journal of the International Measurement Confederation**, v. 175, n. February, p. 109166, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.measurement.2021.109166>>.

YAO, X.; LIU, Y.; LIN, G. Evolutionary programming made faster. **IEEE Transactions on Evolutionary Computation**, v. 3, n. 2, p. 82–102, 1999.

YU, J. J. Q.; LI, V. O. K. Parameter sensitivity analysis of Social Spider Algorithm. 2015 IEEE Congress on Evolutionary Computation (CEC). **Anais...** . p.3200–3205, 2015.

ZHANG, Q.; LI, H.; LIU, Y.; et al. A new quantum particle swarm optimization algorithm for controller placement problem in software-defined networking. **Computers and Electrical Engineering**, v. 95, n. September, p. 107456, 2021. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.compeleceng.2021.107456>>.

ZHANG, Y.; GONG, D.; GAO, X.; TIAN, T.; SUN, X. Binary differential evolution with self-learning for multi-objective feature selection. **Information Sciences**, v. 507, p. 67–85, 2020. Elsevier Inc. Disponível em: <<https://doi.org/10.1016/j.ins.2019.08.040>>.

ZHOU, Q.; BENLIC, U.; WU, Q. An opposition-based memetic algorithm for the maximum quasi-clique problem. **European Journal of Operational Research**, v. 286, n. 1, p. 63–83, 2020. Elsevier B.V.