mathematics

MDPI

Article

A Knowledge-Based Hybrid Approach on Particle Swarm Optimization Using Hidden Markov Models

Mauricio Castillo [1], Ricardo Soto [1], Broderick Crawford [1], Carlos Castro [2] and Rodrigo Olivares [3,*]

[1] Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile; mauricio.castillo.d@mail.pucv.cl (M.C.); ricardo.soto@pucv.cl (R.S.); broderick.crawford@pucv.cl (B.C.)
[2] Departamento de Informática, Universidad Técnica Federico Santa María, Valparaíso 2390123, Chile; carlos.castro@inf.utfsm.cl
[3] Escuela de Ingeniería Informática, Universidad de Valparaíso, Valparaíso 2362905, Chile
* Correspondence: rodrigo.olivares@uv.cl

Abstract: Bio-inspired computing is an engaging area of artificial intelligence which studies how natural phenomena provide a rich source of inspiration in the design of smart procedures able to become powerful algorithms. Many of these procedures have been successfully used in classification, prediction, and optimization problems. Swarm intelligence methods are a kind of bio-inspired algorithm that have been shown to be impressive optimization solvers for a long time. However, for these algorithms to reach their maximum performance, the proper setting of the initial parameters by an expert user is required. This task is extremely comprehensive and it must be done in a previous phase of the search process. Different online methods have been developed to support swarm intelligence techniques, however, this issue remains an open challenge. In this paper, we propose a hybrid approach that allows adjusting the parameters based on a state deducted by the swarm intelligence algorithm. The state deduction is determined by the classification of a chain of observations using the hidden Markov model. The results show that our proposal exhibits good performance compared to the original version.

Keywords: swarm intelligence method; parameter control; adaptive technique; hidden Markov model

check for updates

Citation: Castillo, M.; Soto, R.; Crawford, B.; Castro, C.; Olivares, R. A Knowledge-Based Hybrid Approach on Particle Swarm Optimization Using Hidden Markov Models. *Mathematics* **2021**, *9*, 1417. https://doi.org/10.3390/math9121417

Academic Editor: Frank Werner

Received: 25 May 2021
Accepted: 15 June 2021
Published: 18 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Swarm intelligence methods have attracted the attention of the scientific community in recent decades due to their impressive ability to adapt their methodology to complex problems [1]. These procedures are defined as bio-inspired computational processes observed in nature because they mimic the collective behavior of individuals when interacting in their environments [2]. Many of these procedures have become popular methods, such as genetic algorithms, differential evolution, ant colony system, particle swarm optimization, among several others, and they are still at the top of the main research in the optimization field [3]. Swarm intelligence methods work as a smart-flow using acquired knowledge in the iterative way to find near-optimal solutions [4]. The evolutionary strategy of these techniques mainly depends on the initial parameter configuration which is dramatically relevant for the efficient exploration of the search space, and therefore to the effective finding of high-quality solutions [5].

Finding the best value for a parameter is known as offline parameter setting, and it is done before executing the algorithm. This issue is treated even as an optimization problem in itself. On the other hand, the "online" parameters control is presented as a smart variation of the original version of the algorithm where the normal process is modified by new internal stimulants. Furthermore, according to the No Free Lunch theorem [6], there is no general optimal algorithm parameter setting. It is not obvious to define a priori which parameter setting should be used. The optimal values for the parameters mainly depend on the problem and even the instance to deal with and within the search time that the user

wants to spend solving the problem. A universally optimal parameter value set for a given bio-inspired approximate method does not exist [7].

Using external techniques of the autonomous configuration, the swarm intelligence algorithms are able to adapt their internal processes during the run, based on performance metrics in order to be more efficient [8]. In this way, the user does not require any expert knowledge for reaching efficient solving processes. The adaptation process can be handled under two schemes: offline parameter tuning and online parameter control. In the online control, the parameters are handled and updated during the run of the algorithm, whereas in the offline parameter initialization, the values of different parameters are fixed before the run of the algorithm [6].

In this work, we tackle the online parameter definition problem with an autonomous search concept. This approach focuses on the algorithm parameters adjustment while the search process is performed as guided by the information obtained from the relation between the position of the solutions in the search space. This procedure allows the algorithm to change its functioning during the run, adapting to the particular conditions of the region discovered [9]. The main contribution of this work is to provide a bio-optimization solver with the ability to self-regulate their internal operation, without requiring advanced knowledge to efficiently calibrate the solving process. We propose a hybrid schema applying hidden Markov models to recompute the parameter values into a super-swarm optimization method. Hidden Markov models classify a chain of visible observations corresponding to the relationship between the distance of solutions given by the bio-inspired optimization algorithm. Recognizing when solutions are close to each other is a capability which is not part of all optimization algorithms. An external strategy that satisfies this problem is always valued for its potential use in complex engineering problems.

As a solver technique, we employ the popular particle swarm optimization (PSO) technique. The decision to use the PSO was based on two assumptions: (a) many bio-inspired methods that use the paradigm to generate solutions by updating velocity and position can be improved under this proposal; and (b) PSO is one of the most popular optimizer algorithms, so there is extensive literature that reports its excellent efficiency [10].

To evaluate this proposal, we solve a well-known optimization problem: the set covering problem. We treat a set of the hardest instances taken from the OR-Library [11] in order to demonstrate that the improved behavior of the particle swarm optimization exhibits a better yield than its original version and bio-inspired approximate methods of the state of the art.

The manuscript continues as follows: Section 2 presents a bibliographic search for related work in the field. The results of this search justify the proposal of this work; Section 3 explains offline and online parameter adjustment showing their differences; Section 4 describes the developed solution and the concepts used; Section 5 details the experimental setup, while Section 6 presents and discusses the main results obtained; finally, conclusions and future work are outlined in Section 7.

2. Related Work

Recent works show that swarm intelligence methods remain favorites in the optimization field [12–16], and their popularity has led them to be used in different application domains, such as resource planning, telecommunications, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, signal classification, and manufacturing, among others [17].

In [18], a recent survey on a new generation of nature-based optimization methods is detailed. This study presents metaheuristics as efficient solvers able to solve modern engineering problems in reduced time. Nevertheless, and despite this quality, these techniques present some complications inherent to the phenomenon that defines them. Among them, we find the adjustment and control of their input parameters that directly impact the exploration process of the search space [3,19]. This task is generally done when an external parameter is overcome by a non-deterministic move operator. The exploration

phase operates with a set of agents that attempt to escape from local optima [20–22]. In this context, the autonomous search paradigm [23] describes how to empower metaheuristics to adjust its parameters during the resolution process, reacting to the information obtained. This technique accelerates the convergence of the algorithm, reducing the operational cost, and providing robustness to the process, making it react to the problem that is being solved.

By focusing on online parameter setting in optimization algorithms, we can find [24]. Here, the authors provide a perspective (from 30 years) explaining that an automatic tuning allows adapting the parameter values depending on the instance and the evolution of the algorithm. One of the first attempts using a controlled parameter variation in the particle swarm optimizer was proposed in [25]. A linear decreasing function was used for the inertia coefficient parameter w, with a start value of $w_s = 0.9$ and a final value of $w_f = 0.4$, testing this configuration for well-known benchmark functions. In [26], the parameter adjustment tries to make a transition between exploration and exploitation states, linearly decreasing and increasing the personal and social coefficients, respectively. In [27], an ideal velocity for the particles is defined, and the value of ω is updated to adjust the current average velocity to a value closer to the ideal. In [28,29], hybrid approaches to the PSO algorithm were developed. The first one includes a two-fold adaptive learning strategy to guarantee the exploration and exploitation phases of the algorithm. The second one proposes a learning strategy using a quasi-entropy index when local search works.

Related works that adjust the parameters of the PSO algorithm based on the fitness obtained along the iterations have been discussed. For example, in [30], two values that describe the state of the algorithm are defined: the evolutionary speed factor and the aggregation degree. Both values are used to update the ω values for each particle. In [31], the inertial and best solution acceleration coefficients are adjusted for each particle based on the relation between the current particle's fitness and the global best fitness. In [32], authors related the value of ω with the convergence factor and the diffusion factor to dynamically set its value. In [33], the value of ω for each particle in the swarm is computed based on the ratio of the personal best fitness value with the personal best fitness average, for all particles. In [34], the value of ω is calculated from the relation to fitness-based ranking for each particle and the problem dimension number. In [35], the value of ω is updated to accelerate the PSO convergence. In [36], a success rate for the PSO is defined based on the proportion of particles that improved their personal best position at iteration t. This method aims to increase the value of w when the proportion of particles that improved their personal best is high and decrease it otherwise. In [37], the inertial coefficient w is increased for the best particle and decreased for all others, based on the idea that the best particle is more confident on the direction of its movement. The inertial coefficient decreases linearly for the rest of particles.

Taking into account the fitness, but considering the relationship between the position of the particles in relation to the best, efforts have been made to adjust the value of the parameters. In [38], acceleration and inertial parameters are adjusted according to state deduction using a fuzzy classification system. The state classification depends on the calculation of the evolutionary factor. In [39], inertia weight and acceleration coefficients are adjusted using the gray relational analysis, proposed by [40], using the relation of the particle compared to the global best.

In recent years, the hybridization of metaheuristics with supervised and unsupervised methods has emerged as a promising field in approximation algorithms. In 2017, the term "Learnheuristic" was introduced in [41] to address the integration between metaheuristics and machine learning algorithms, and they provide a survey of the closest papers. In this research, a simple but robust idea is proposed. There are two work-groups: machine learning algorithms to enhance metaheuristics and metaheuristics to improve machine learning techniques. For the first group, is it possible to find (i) metaheuristics for improving clustering methods using the artificial bee colony algorithm [42], local search [43], particle swarm optimization [44] and ensemble-based metaheuristics [45]; (ii) metaheuristics to

efficiently address the feature selection topic [46]; and (iii) metaheuristics for improving classification algorithms [47–49].

Recently, machine learning methods have also been used for the parameter control issue. An example of auto-tuning in a deep learning technique can be seen in [50]. Here, the authors provide an improvement to support and store a large number of parameters required by deep learning algorithms. Now, considering improvements that include machine learning in optimization algorithms, we can find in [51] that the PSOs parameters are adjusted by using an agent that chooses actions from a set in a probabilistic way, then measures the results and sends a reinforcement signal. In [52], a decision tree was used to perform the tuning of parameters. Hop-field neural networks were used in [53] to initiate solutions of a genetic algorithm applied to the economic dispatch problem. A mechanism for identifying and escaping from extreme points is punished in [54]. Here, the whale swarm algorithm includes new procedures to iteratively discard the attenuation coefficient and it enables the identification of extreme points during the run. An integration between the Gaussian mutation and an improved learning strategy were also proposed to boost a population-based method in [55]. New interactions between machine learning and optimization methods have recently been published in [56–58]. Moreover, improved machine learning techniques have been used for action recognition from collaborative learning networks [59], for the automatic recognition and classification of ECG and EEG signals [60–62], for complex processing on images [63], for health monitoring systems using IoT-based techniques [64], and several others works. In [65], a support vector machine is employed as a novel methodology to compute the genetic algorithm's fitness. A similar work can be seen in [66]. Clustering techniques were studied for the exploration of the search space [67] and for dynamic binarization strategies on combinatorial problems [68]. In [69], case-based reasoning techniques were investigated to the identify sub-spaces of searches to solve a combinatorial problem. In [70], an incremental learning technique was applied to constrained optimization problems. Finally, in [71], an alternative mechanism for the incorporation of negative learning on the ant colony optimization is proposed.

Finally, a few works explore the integration of hidden Markov models and optimization algorithms. In [72,73], a population-based method is proposed to train hidden Markov models. Another work which uses a bio-solver to optimize a hidden Markov model is presented in [74]. On the other hand, recent studies have studied how hidden Markov models improve the optimization algorithms. In [75], authors studied the relation of particle distances to determine the state of the particle swarm optimizer. The states were inspired by [38,76]. Parameters are updated according to the determined state. Similar work can be seen in [75,77,78].

3. Preliminaries

Parameter setting is known as a strategy for providing larger flexibility and robustness to the bio-inspired techniques, but requires an extremely careful initialization [7,19]. Indeed, the parameters of these procedures influence the efficiency and effectiveness of the search process [79]. To define a priori which parameter setting should be used is not an easy-task. The optimal values for the parameters mainly depend on the problem and even the instance to deal with and the search time within which the user wants to solve the problem.

This strategy is divided into two key approaches: the offline parameter tuning and the online parameter control (see Figure 1).

The adaptation process is called online when the performance information is obtained during solving, while the process is considered offline when a set of training instances is employed to gather the feedback [80]. The goal of parameter tuning is to obtain parameter values that could be useful over a wide range of problems. Such results require a large number of experimental evaluations and are generally based on empirical observations. Parameter control is divided into three branches according to the degree of autonomy of the strategies. Control is deterministic when parameters are changed according to a previously established schedule, adaptive when parameters are modified according to rules that take

into account the state of the search, and self-adaptive when parameters are encoded into individuals in order to evolve conjointly with the other variables of the problem.

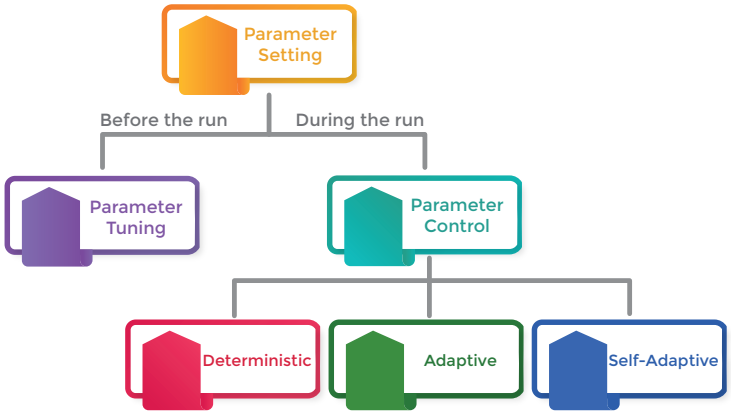

Figure 1. Scheme of the parameter adaptation process in bio-inspired methods.

The offline approaches require a high computational cost [19,81]. This cost increases when we use the offline approach for solving each input instance of a problem. Indeed, the optimal parameter values depend on the instances of the problem to be addressed. Furthermore, according to the No Free Lunch theorem, there is no generic optimal parameter setting. A universally optimal parameter value set for a given bio-inspired approximate method does not exist [6,82].

The great advantage of the online approach against the offline approach is that the effectiveness of the parameter control may change during the search process. That is, at different moments of the search, different optimal values are found for a given parameter. Hence, online approaches that change the parameter values during the search must be designed. Online approaches may be classified as follows [6]:

- Dynamic update: A random or deterministic updates the parameter value. This operation is performed without taking into account the search progress.
- Adaptive update: In this approach, parameter values evolve during the search progress. To change the parameter values, a function that mimics the behavior of the phenomenon is performed. For that, the memory of the search is mainly used. Hence, the parameters are associated with the representation and these are subject to updates in function of the problem's solution.

Online control is only recent but also interesting and challenging as the feedback is uniquely gathered during solving time with no prior knowledge from training phases and no user experts.

4. Developed Solution

Swarm intelligence methods have been developed for almost 30 years. The term swarm intelligence was coined in 1993 [83] and since its appearance, it has become a popular optimization method [84]. The distributed structure presents possible advantages over centralized methods, such as the simplicity of the search agents, the robustness provided by the redundancy of components, and the ability to escape local optimums [6]. This type of structure is typical in many biological systems, such as insect colonies, flocks of birds, and schools of fish. The synergy between the swarm members provides each of them with advantages that they could not achieve on their own, such as protection against predators and a more reliable supply of food [14,85].

Particle swarm optimization is a most popular population-based bio-inspired algorithm [86,87]. This method intelligently mimics the collaborative behavior of individuals or "particles" through two essential components: the position and the velocity. A set of particles (candidate solutions) forms the swarm that evolves during several iterations.

This procedure describes a powerful optimization method [88]. The technique operates by altering velocity through the search space and then updates its position according to its own experience and neighboring particles.

Particle swarm optimization can be identified as an intelligent system with two phases: (a) when the algorithm reaches large velocities in the initial phase, the current solutions focus more on diversification; (b) as velocities tend towards zero, the current solution focuses on intensification. The best reached solutions are memorized as *pBest*. The standard particle swarm optimization is governed by the movement of particles through two vectors: the velocity $V_i = \langle v_i^1, v_i^2, \ldots, v_i^D \rangle$ and the position $X_i = \langle x_i^1, x_i^2, \ldots, x_i^D \rangle$. First, the particles are randomly positioned in a D-dimensional heuristic space with random velocity values. During the evolution process, each particle updates its velocity via Equation (1) and position through Equation (2):

$$v_i^d = \omega v_i^d + c_1 \phi_1^d (pBest_i^d - x_i^d) + c_2 \phi_2^d (gBest^d - x_i^d) \tag{1}$$

$$x_i^d = x_i^d + v_i^d \tag{2}$$

where $d = \{1, 2, \ldots, D\}$ represents the size of the problem; the positive constants ω, c_1, and c_2 are acceleration coefficients; ϕ_1 and ϕ_2 are two uniformly distributed random numbers in the range $[0, 1]$; $pBest_i$ is the best position reached by ith particle; and $gBest$ is the global best position found by all particles during the resolution process.

4.1. Evolutionary Factor *f*

Diversity measures explain the distribution of a set of particles in the search space [89]. In this sense, ref. [38] proposes a measure derived from the distances between the PSO particles, known as the evolutionary factor f. This factor is computed in Equation (3):

$$f = \frac{d_p - d_w}{d_g - d_w} \in [0, 1] \tag{3}$$

where d_p represents the pBest fitness (position) reached by a particle until that moment. The d_w and d_g values describe the worst and gBest distance of the swarm, respectively.

Figure 2 shows the relationship between the position of the PSO particles and the state in which they will be classified: sub-figure (a) depicts the exploration state, where the particles are far away from each other; sub-figure (b) shows an exploitation/convergence state, where the particles are close to each other and the best particle appears in the center of the group; sub-figure (c) illustrates the jumping-out state, where the particles are close to each other, but the best particle has found a better zone and appears far from the others. To compute the difference among distances, the average of all distances d_{avg} is also required.

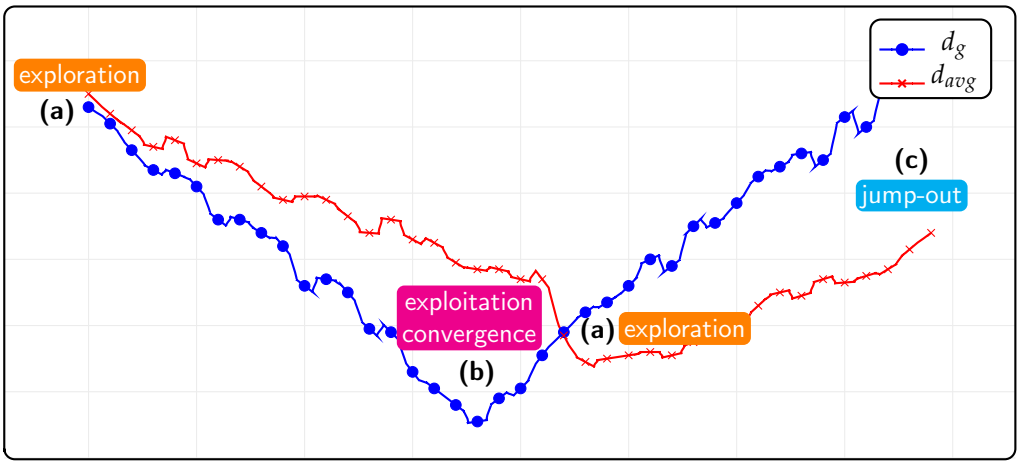

Figure 2. Example of PSO particle distribution: (**a**) $d_g \approx d_{avg}$ exploration; (**b**) $d_g \ll d_{avg}$ exploitation, convergence; and (**c**) $d_g \gg d_{avg}$ jump out.

4.2. Markov Models

This section defines the hidden Markov model used in this paper to identify the state (or inner-phase) of the PSO. For that, we detail how the states of a PSO can be modeled as a Markov chain, allowing the state to be inferred from the evolutionary factor calculated for the particle swarm optimizer.

4.2.1. Markov Chains

The Markov chain is a statistical model that defines the probability of transition from one state to another within a finite set of states. The Markov chain assumes that the transition to the next state only depends on the current state, regardless of previous states [90].

The states within a Markov chain are finite and the transitions between states are defined according to a probability. This probability must add up to 1, indicating all probable states that can be reached from the current state. Figure 3 shows a typical Markov chain.

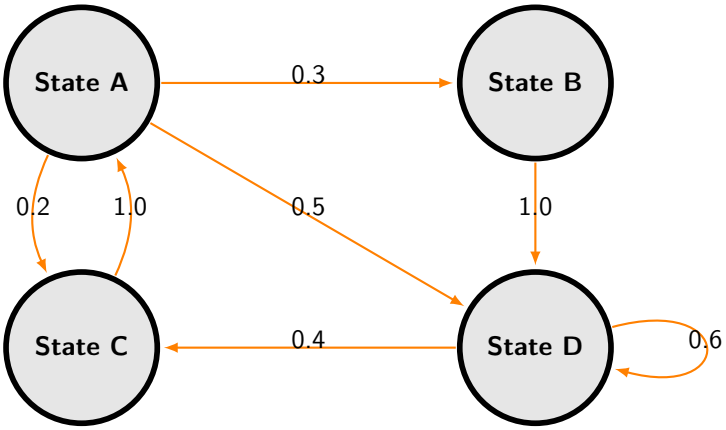

Figure 3. Markov chain. The arcs connecting the nodes/states indicate the transition probability between states.

For this Markov chain, we observe four states: A, B, C, and D. Arrows indicate which state can be accessed from a particular state and the number next to it indicates the probability that the transition occurs. These probabilities can be studied as a square matrix where the transition probability for all states is represented. The transition matrix M in the example above is defined as

$$M = \begin{bmatrix} 0 & 0.3 & 0.2 & 0.5 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.6 \end{bmatrix}$$

4.2.2. Hidden Markov Model

The hidden Markov model (HMM) is a framework that allows, through the observation of some visible state, to deduce elements of the Markov chain that are not directly visible, i.e., hidden [91,92]. The transition between states is assumed to be in the form of a Markov chain. The Markov chain can be defined by an initial probability vector π and a transition matrix A. Observable elements O are emitted according to some distribution in each hidden state, and they are noted in the emission matrix B.

There are three main tasks that an HMM solves:

1. Decoding.Given the parameters A, π, B, and the observed data O, estimate the optimal sequence of hidden states Q;
2. Likelihood. Given an HMM $\lambda = (A, B)$ and a sequence of observations O, determine the probability that those observations belong to the HMM, $P(O|\lambda)$;

3. Learning. Given a sequence of observations O and a set of states in the HMM, we learn its parameters A and B.

In our work, the decoding task will be used to determine the hidden state given a group of observations, obtained from a discretization of the evolutionary factor. The learning task will be used in each iteration to learn the value of the B emission matrix, specifically.

4.3. HMM-PSO Integration

In [38], four inner-phases (or states) through which PSO moves are defined: exploration, exploitation, convergence, and jumping-out. A previous work describes these states as a Markov chain [93] (detailed in Figure 4). This chain corresponds to the hidden chain that will be deduced using decoding and learning tasks.

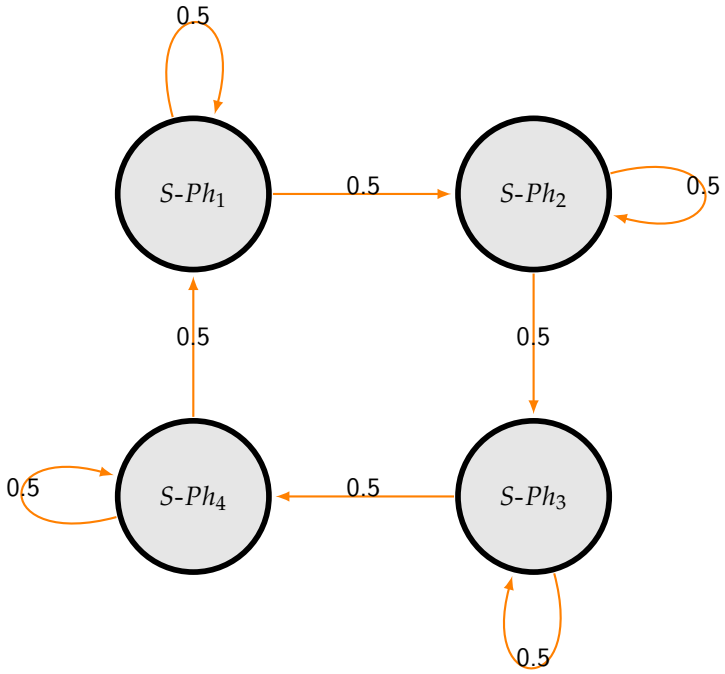

Figure 4. Evolutionary states defined for the Adaptive PSO algorithm: $S\text{-}Ph_{\{1,2,3,4\}}$ represent Exploration, Exploitation, Convergence, and Jump-out, respectively.

The HMM receives three input parameters: the first one is an initial probability vector π that computes a deterministic start in the exploration state: $\pi = [1, 0, 0, 0]$. The second parameter corresponds to the transition matrix between states A. As shown in figure, it is only possible to stay in the current state or to advance to the next state from left to right. As an initial value, all transitions have a probability of 0.5. The matrix A is defined as

$$A = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix}$$

The type of hidden Markov model used in this work obtains its classifications from a group of observations belonging to a discrete alphabet. Therefore, we apply a discretization process to the evolutionary factor f of each inner-phase of the PSO. The discretization process used in this work is defined in [75] and corresponds to identifying the interval in which the calculated evolutionary factor belongs. The seven defined intervals are: $([0, 0.2], [0.2, 0.3], [0.3, 0.4], [0.4, 0.6], [0.6, 0.7], [0.7, 0.8], [0.8, 1])$.

The emission matrix B is the third parameter of the model. This matrix corresponds to the probability with which the elements of the alphabet of observations are emitted for each state. The emission matrix that we use in this work is defined in [75,94], and it is detailed as follows:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.25 & 0.25 & 0 \\ 0 & 0.25 & 0.25 & 0.5 & 0 & 0 & 0 \\ 2/3 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix}$$

The parameters π, A and B completely define the hmm model. Once defined, the model is capable of deducing hidden states—exploration, exploitation, convergence, jumping out—from the discretization of a chain of observations—evolutionary factor f—using the Viterbi algorithm. At each iteration, it is possible to adjust the parameters of the hmm model using task 2 with Baum–Welch's algorithm. For more details about the operation of both algorithms, please refer to [95].

5. Experimental Setup

In this section, we detail the proposal of integration of HMM in PSO, for the determination of the state and control of the parameters. This hybridization was tested on the set covering problem, which is a classic combinatorial optimization problem. One of the first works was proposed in [96], and it defines the Equation (4) as the formulation for the set covering problem:

$$\begin{gathered} \texttt{minimize} \sum_{j=1}^{n} c_j x_j \\ \text{subject to:} \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall\, i \in M \\ x_j \in \{0,1\} \quad \forall\, j \in N \end{gathered} \tag{4}$$

where c_j represents positive constants of the cost vector, and a_{ij} details binary values of the constraint matrix with M-rows and N-columns. If column j covers a row i, then $x_j = 1$. Otherwise, $x_j = 0$. We take the hardest instances of the set covering problem from the OR-Library [11].

To have an overview of the components involved in the search process, state identification and parameter control, Figures 5 and 6 show the flowchart of the algorithms.

Based on experimental analysis, the parameters are adjusted according to Table 1. Table 2 shows the initial parameter settings for the original PSO algorithm and our version. The initial values for the original PSO are the same as those used by the author. The initial values for the ω and np parameters on the hidden Markov model supporting the PSO algorithm (HPSO) come from [97].

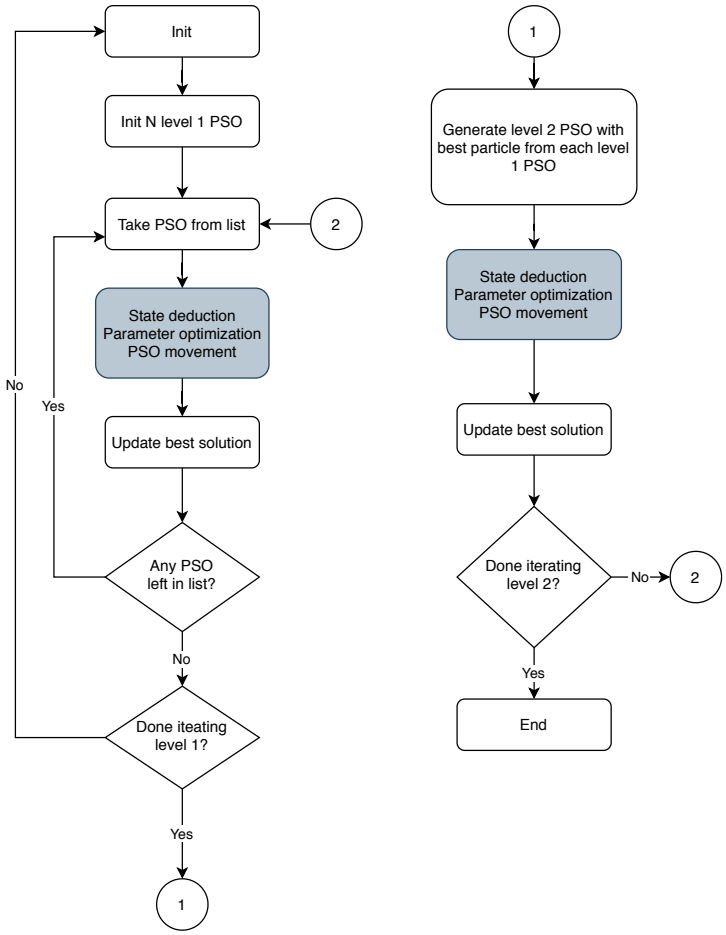

Figure 5. PSO algorithm state deduction integration.

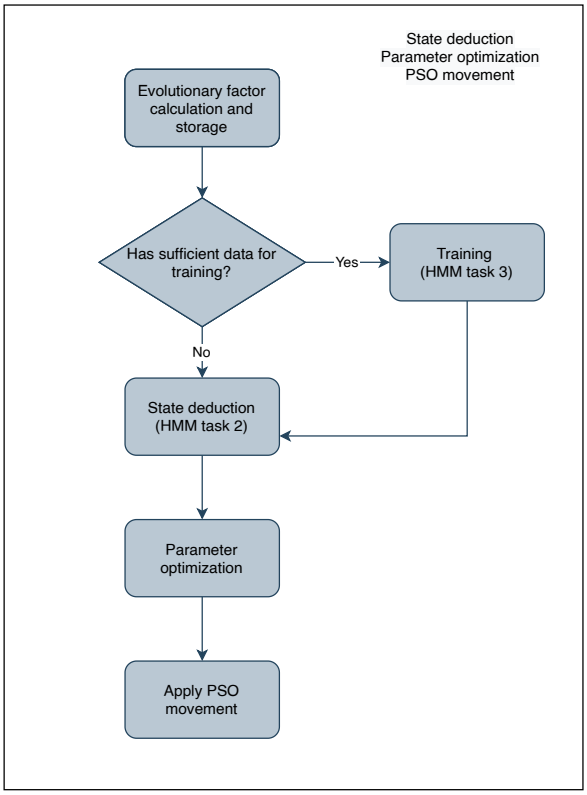

Figure 6. State deduction and parameter adjustment for PSO.

Table 1. Parameters update by identified state.

State (Inner-Phase)	Inertial Velocity w	Number of Particles np
Exploration	$\omega = \omega_{min} + (\omega_{max} - \omega_{min}) \cdot Rand(0,1)$	$np - 1$
Exploitation	$\omega = \frac{1}{1+\frac{3}{2}\exp^{-2.6f}}$	$np + 1$
Convergence	$\omega = \omega_{min}$	$np + 1$
Jump out	$\omega = \omega_{max}$	$np - 1$

Table 2. Initial configuration for input parameter values.

Orginal PSO Parameters		Proposed HPSO Parameters	
Parameter	**Value**	**Parameter**	**Value**
ω	$1 - \frac{k}{L+1}$	ω_{min}	0.4
		ω_{max}	0.9
np_{min}	5	np_{max}	30
np_{max}	50	np_{max}	30
$c1$	2.05 $rand(0,1)$	$c1$	2.05 $rand(0,1)$
$c2$	2.05 $rand(0,1)$	$c2$	2.05 $rand(0,1)$
iter. num.	50	iter. num.	50
iter. num.	250	iter. num.	250

6. Results and Discussion

In this section, we evaluated the functioning of our proposed HPSO. We compared our proposal against the original PSO. Then, we present a statistical comparison of the results obtained, and we illustrate the convergence of the search process and the percentages of exploration and exploitation.

Before integrating the adaptive approach, we analyzed the temporal complexity of the original PSO algorithm to evaluate that our proposal does not impact its performance. If we study each statement and expression, including control flows from the particle swarm optimization algorithm, we can state that time complexity is given by $(T \times np \times n)$, where T represents the maximum number of iterations, np stores is the number of particles (or solutions), and n is the dimension of each particle. In the worst case, the basic algorithm is upper bounded by $O(kn^2)$.

Then, performing a temporal analysis about our adaptive approach, we state that the temporal complexity of PSO is not altered. If we consider that: (a) this procedure operates in a determinate number of iterations (see Table 2); (b) it works with the same solutions; and (c) it runs in a way that is independent from the main algorithm, we can affirm that the upper bound is given by $(np \times n)$, which again, has an upper bound equal to $O(kn^2)$.

6.1. Original PSO Comparison

Table 3 shows the results obtained by our proposal and the original PSO in 11 hard instances of the set covering problem [68]. Each instance was executed 31 times and each run iterated 1000 cycles. These runs allow us to analyze the independence of the samples by determining the Z_{best}. The comparative includes the relative percentage distance (RPD). This value quantifies the deviation of the objective value Z_{best} from Z_{opt}, which is the minimal best-known value for each instance in our experiment, and it is computed as follows:

$$RDP = \left(\frac{Z_{best} - Z_{opt}}{Z_{opt}} \right) \tag{5}$$

Results show that the difference between both algorithms increases as the instance of the problem grows. The best results are highlighted with underline and maroon color.

For example, in the scp41, the best reached solution by HPSO overcomes than the classical PSO algorithm. The same strategy is used in all comparisons.

Table 3. Comparison of results between PSO and HPSO.

Instance	Optimum	Best HPSO	Best PSO	Avg. HPSO	Avg. PSO	RPD HPSO	RPD PSO
scp41	429	**429**	430	**429.81**	432.419	**0**	0.233
scp51	253	**253**	255	**253.68**	260.71	**0**	0.791
scp61	138	**138**	140	**138.19**	140.871	**0**	1.449
scpa1	253	**253**	256	**254.32**	258.097	**0.395**	1.186
scpb1	69	**69**	71	**69**	91.129	**0**	2.899
scpc1	227	**227**	234	**228.36**	238.258	**0**	3.084
scpd1	60	**60**	79	**60.13**	123.323	**0**	31.667
scpnre1	29	**29**	85	**29**	106.871	**0**	193.103
scpnrf1	14	**14**	39	**14**	49.29	**0**	178.571
scpnrg1	176	**176**	348	**178.17**	480.839	**0.568**	97.727
scpnrh1	63	**65**	277	**65.25**	349.452	**1.587**	339.683

Using the Wilcoxon–Mann–Whitney rank sum statistical test, we compare the results obtained by our proposal against the original PSO algorithm. It is valid to use this test because all runs are independent from each other and the results do not follow a normal distribution, since they are affected by pseudo-random numbers. Thirty-one samples of the obtained best fitness for 11 different instances of the set covering problem are compared. The test gives an p-value lower than 0.05 if it is possible to determine that one sample has statistically lower values than the other, and a value higher than 0.05 if not. Table 4 shows the comparison between the two algorithms.

Table 4. Statistical comparison.

Instance	HPSO < PSO	PSO < HPSO
scp41	0.728	0.277
scp51	**0.002**	0.998
scp61	**0.000**	1.000
scpa1	**0.000**	1.000
scpb1	**0.000**	1.000
scpc1	**0.000**	1.000
scpd1	**0.000**	1.000
scpnre1	**0.000**	1.000
scpnrf1	**0.000**	1.000
scpnrg1	**0.000**	1.000
scpnrh1	**0.000**	1.000

We can see that it was not possible to determine a statistical difference only for instance *scp41*, while for the other instances, the hypothesis that our algorithm improved the resolution process is confirmed.

6.2. Exploration/Exploitation Balance

In swarm intelligence methods, the population diversity is a measurement which evidences the performance of an algorithm, through the distribution of generated solutions [98,99]. This principle is significantly important to analyze the behavior of each solution in a swarm as well as the swarm as a whole. A recent work proposes a model

based on the dimension-wise measurement to study the yield of algorithms [100]. The formulations that calculate this metric are defined in Equations (6) and (7):

$$Div(j) = \frac{1}{np} \sum_{i=1}^{np} |mean(x^j) - x_i^j| \tag{6}$$

$$Div = \frac{1}{D} \sum_{d=1}^{D} Div(d) \tag{7}$$

where $Div(j)$ describes the computed dimensional Hussain diversity over a solution x^j, $mean(x^j)$ represents the mean over each dimension j, np stores the number of solutions (population size). Finally, D saves the dimension size. After taking the dimension-wise distance of each swarm-individual i from the mean of the dimension j, we compute the average $Div(j)$ for all the individuals. Then, the average diversity of all dimensions is calculated in Div.

Using this fundamental, in [101], a model is proposed that allows to compute the evolution of the exploration and the exploitation effects obtained by the two algorithms, in each instance through all iterations. Resulting values represent percentages of exploration and exploitation on the population at iteration t. To calculate the exploration (XPL) balance, Equation (8) is applied, and to obtain the exploitation (XPLT) impacts, Equation (9) is employed:

$$XPL\% = \left(\frac{Div}{Div_{max}} \right) \times 100 \tag{8}$$

$$XPLT\% = \left(\frac{|Div - Div_{max}|}{Div_{max}} \right) \times 100 \tag{9}$$

For both equations, Div and Div_{max} represent the measures of diversity (distance) calculated over the population. Div represents the diversity of the full set of search agents through the aggregation of the diversity of each agent. However, Div_{max} represents the maximum value of diversity found. As can be intuited, the measurement of the percentage of exploration and exploitation varies depending on the measure of diversity used.

Figures 7 and 8 show the behavior of both algorithms. There are peaks, very noticeable in the original version and softer in our version. These peaks represent the change between inner-phases (the exploration and the exploitation processes).

6.3. Convergence Curves

We show plots for the convergence of the algorithms, PSO and HPSO, solving the set covering problem. Both algorithms reach a promising zone in the search space early on its execution. Convergence for both algorithms is very similar. Figures 9 and 10 shows the convergence achieved by both algorithms for their best execution on the instances scp41, scpa1, scpnre1, and scpnrh1.

6.4. Results Discussion

For the evaluation of the autonomous search method proposed, we used different measures that allowed us to evaluate the performance: a statistical comparison of the results of our algorithm against the original PSO, the variation of the evolutionary factor during the execution, the variation of the internal parameters of our algorithm, and the percentage of exploration and exploitation obtained in the search using the dimensional Hussain diversity measure.

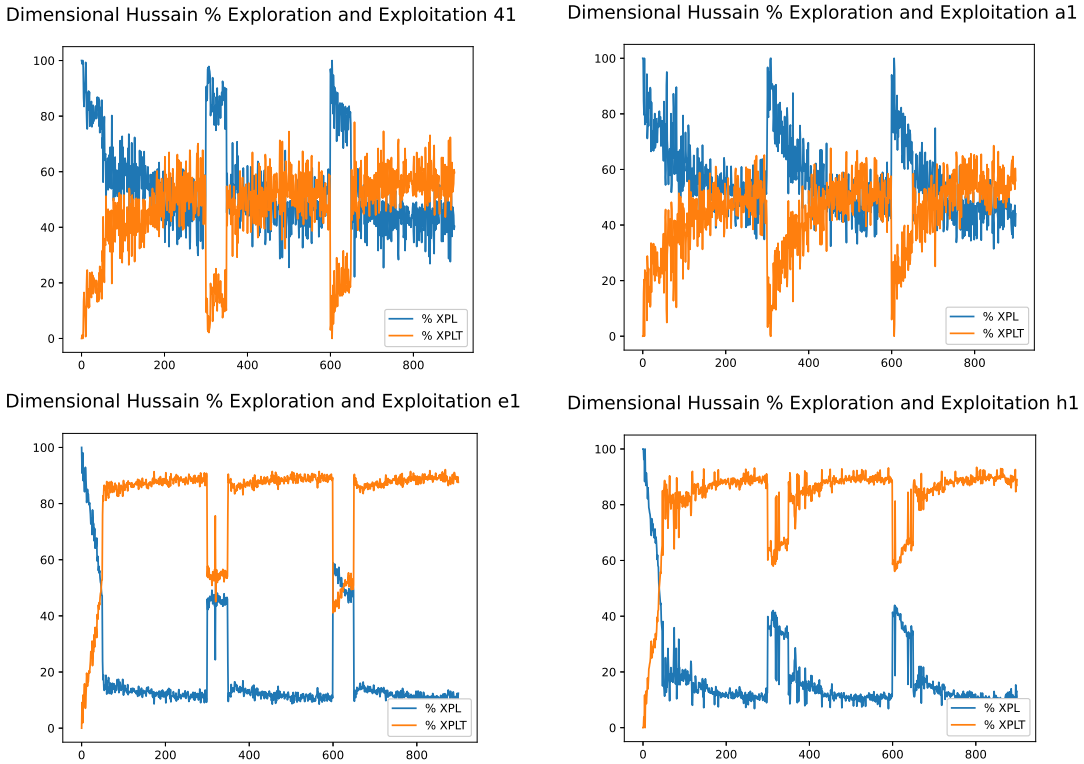

Figure 7. Exploration and exploitation percentage for original PSO. For small instances (scp41 and scpa1), the algorithm shows an exploitative behavior, for bigger instances (scpnre1 and scpnrh1), the algorithm shows an exploitative behavior. We can observe the transition between inner-phases at iterations 50, 300, and 600.

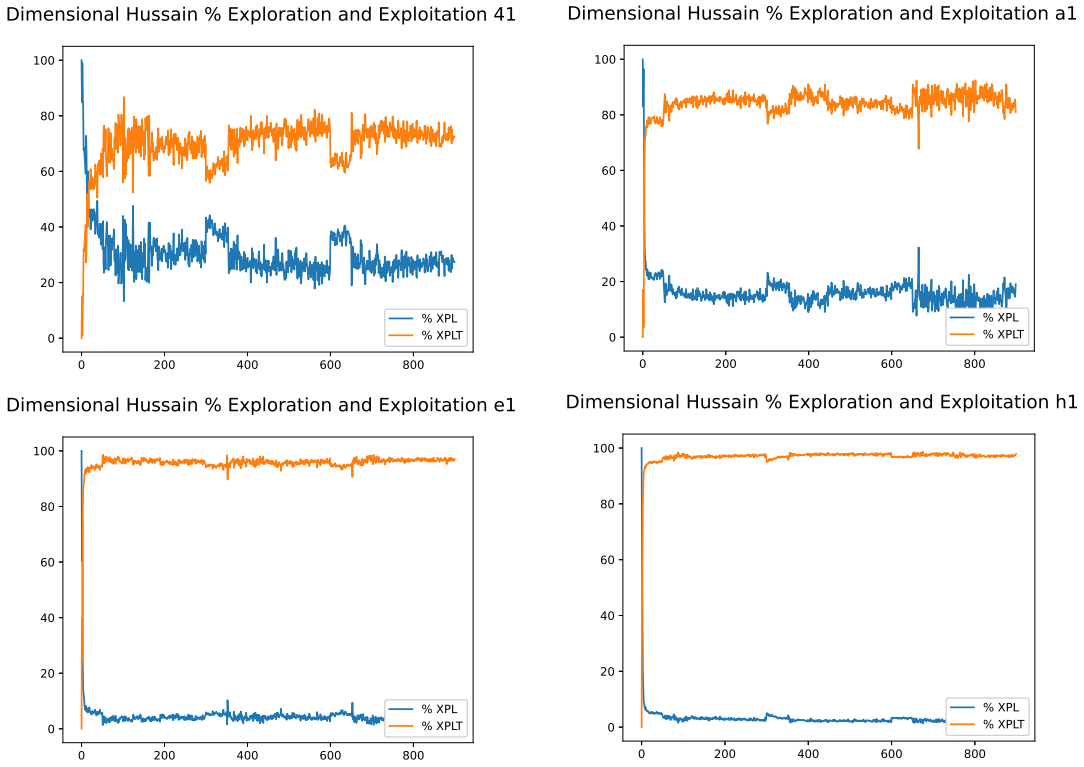

Figure 8. Exploration and exploitation percentage for HPSO. The algorithm shows a mostly exploitative behavior for small and big instances.

The statistical comparison of the results was conducted to determine whether there is an improvement to the original algorithm. Statistical tests confirm that there is a statistically distinguishable improvement when comparing the results of HPSO and PSO solving the combinatorial problem, for all tested instances.

The value of the evolutionary factor f shows a tendency to remain low, interrupted by sudden rises. Low values of f indicate that the PSO particles are close to each other and that the algorithm is converging. The higher value indicates that a particle found a solution with a better fit in an area far away from the group, which indicates that the PSO was able to avoid a local optimum.

The variation of the internal parameters shows an upward trend for the number of particles, and as the search progresses, new particles participate in the search. On the other hand, the inertia coefficient varies abruptly, going from the minimum value to the maximum value in a few iterations. This behavior did not affect the quality of the solutions; however, such an abrupt variation does not generate a recognizable pattern and the ω adjustment method must be reviewed.

The percentage of exploration and exploitation obtained shows that the algorithm maintains a mostly intensifying behavior, demonstrating that a promising area was found during the first iterations, maintaining the trend throughout the search. The transition between exploration and exploitation is more noticeable in smaller instances, which is explained by a smaller size of the search space. In general, the exploration and exploitation graphs show an efficient search.

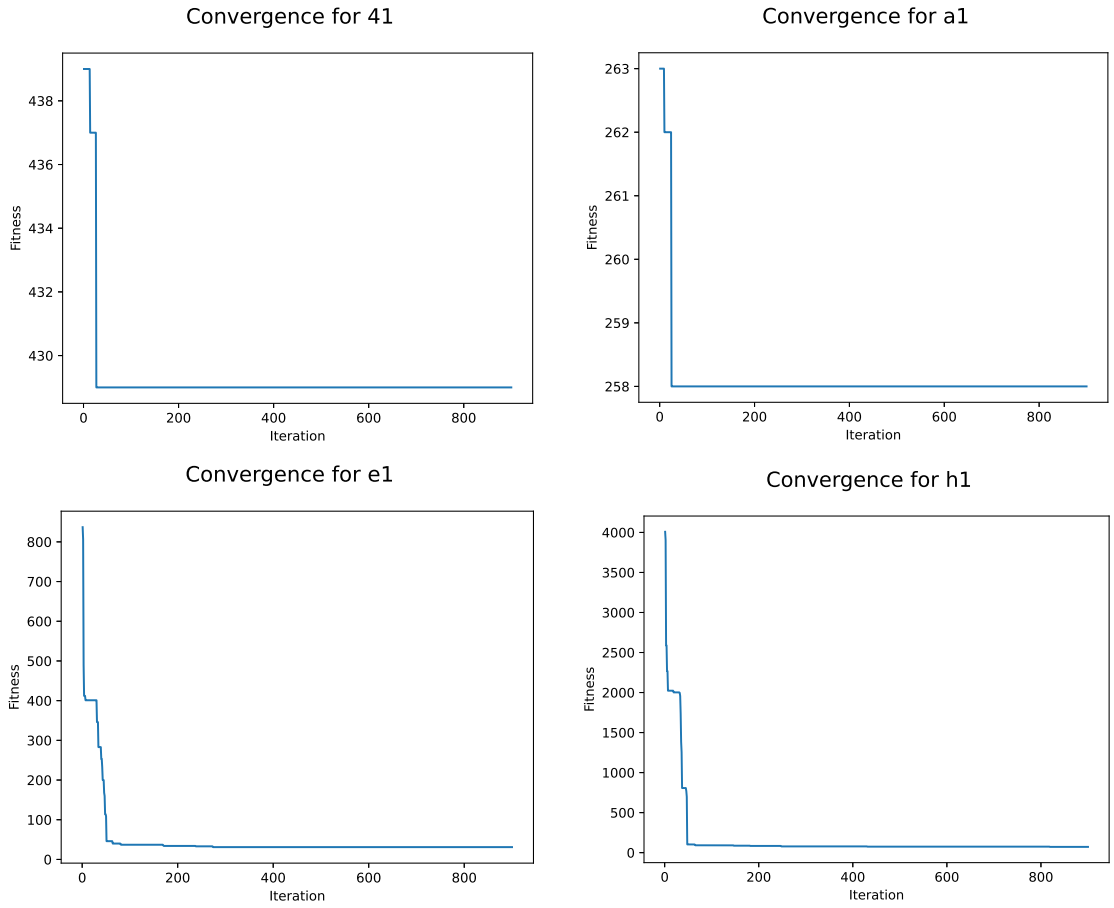

Figure 9. Convergence of PSO: The algorithm shows a premature convergence, with very few improvements after the first 50 iterations.

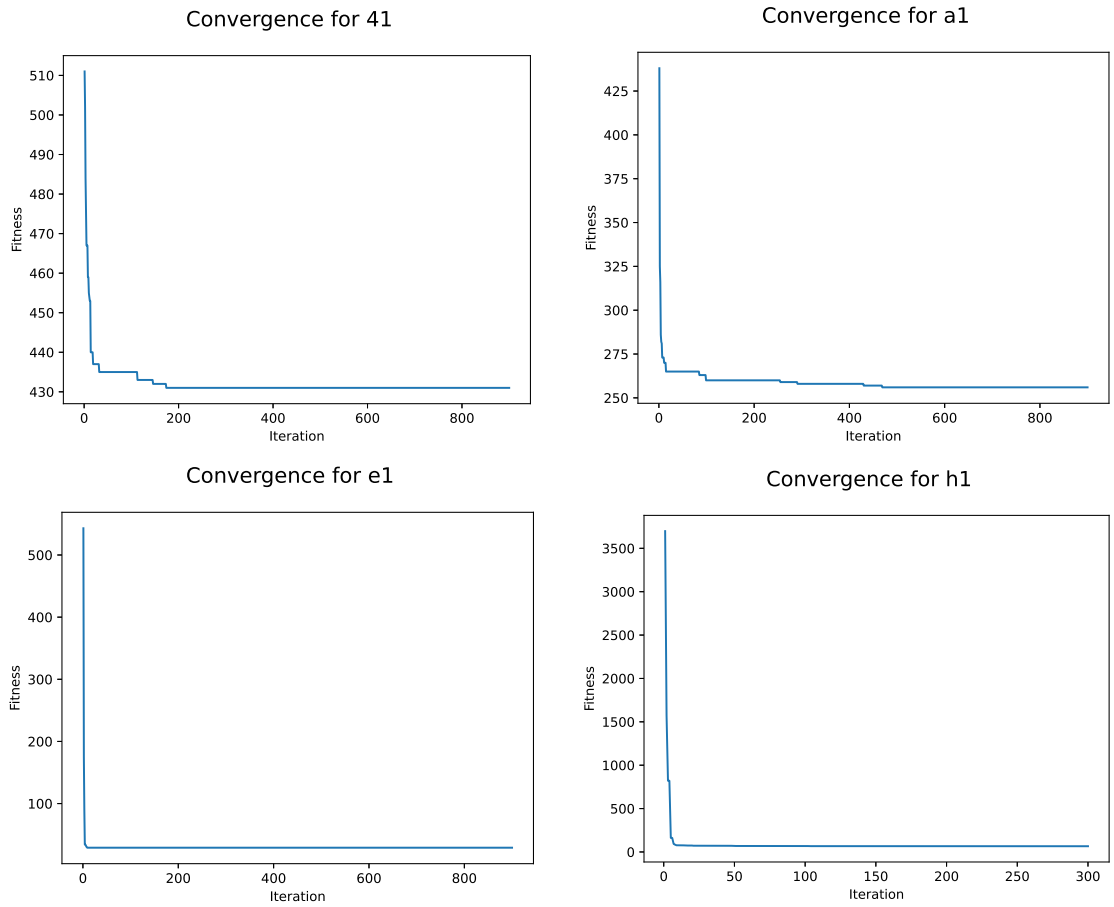

Figure 10. Convergence of HPSO. The algorithm shows improvements until approximately iteration 150, which represents 50% of the total iterations.

7. Conclusions and Future Work

In this work, we presented an autonomous search method for the PSO algorithm. This work was performed by hidden Markov models, which allow for the state identification of a PSO while the search process is running. The identification allows us to adjust PSO parameters based on a state deducted by the HMM. The deduction was made from the calculation of the evolutionary factor f metric, which gives information about the disposition of the particles inside PSO.

Different combinations of parameters to be adjusted for the PSO algorithm were evaluated, experimenting on a set of instances of the set covering problem and measuring the results. This experimentation showed that the combination of parameters w and np generates the best results. Then, the algorithm was compared against the original version of PSO without parameter control. The comparison of results was made using the Wilcoxom–Mann–Whitney statistical test, with the aim of testing the hypotheses posed for this work. The hypothesis was assumed and the parameter control shows a statistical difference in the quality of the solutions obtained. Moreover, we present figures that show the exploration and exploitation balance obtained by our proposal. If it is possible observe that the exploitation percentage increases compared to the original PSO. This behavior indicates that the HPSO was able to find better regions in the heuristic space, intensifying the search in those areas.

Future works consider verifying the impact on the classification of states when making changes to the transfer and binary functions in 0/1 optimization problems [102]. The discretizations made to the evolutionary factor f can also be adjusted, which will change the input data for the HMM model and its deductions. Finally, the PSO algorithm can

be viewed as a framework for population-based metaheuristics, therefore testing with a different base algorithm is considered.

Author Contributions: Formal analysis, R.S., B.C.; investigation, R.S., B.C., R.O., and M.C.; methodology, R.S., R.O., and C.C.; resources, R.S. and B.C.; software, R.O. and M.C.; validation, B.C. and C.C.; writing—original draft, M.C., R.O.; writing—review and editing, R.S., and B.C., R.O., M.C., and C.C. All the authors of this paper are responsible for every part of this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. Broderick Crawford is supported by Grant ANID/FONDECYT/REGULAR/1210810.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Mavrovouniotis, M.; Li, C.; Yang, S. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm Evol. Comput.* **2017**, *33*, 1–17. [CrossRef]
2. Gill, S.S.; Buyya, R. Bio-inspired algorithms for big data analytics: A survey, taxonomy, and open challenges. In *Big Data Analytics for Intelligent Healthcare Management*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 1–17.
3. Huang, C.; Li, Y.; Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* **2019**, *24*, 201–216. [CrossRef]
4. Nayyar, A.; Nguyen, N.G. Introduction to swarm intelligence. In *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2018; pp. 53–78.
5. Mejía-de Dios, J.A.; Mezura-Montes, E.; Quiroz-Castellanos, M. Automated parameter tuning as a bilevel optimization problem solved by a surrogate-assisted population-based approach. *Appl. Intell.* **2021**, 1–23. [CrossRef]
6. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 74.
7. Stutzle, T.; Lopez-Ibanez, M.; Pellegrini, P.; Maur, M.; Montes de Oca, M.; Birattari, M.; Dorigo, M. Parameter adaptation in ant colony optimization. In *Autonomous Search*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 191–215. [CrossRef]
8. Soto, R.; Crawford, B.; Aste Toledo, A.; Castro, C.; Paredes, F.; Olivares, R. Solving the manufacturing cell design problem through binary cat swarm optimization with dynamic mixture ratios. *Comput. Intell. Neurosci.* **2019**, *2019*, 4787856. [CrossRef]
9. Hamadi, Y. Autonomous search. In *Combinatorial Search: From Algorithms to Systems*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 99–122.
10. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2017**, *22*, 387–408. [CrossRef]
11. Beasley, J. OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **1990**, *41*, 1069–1072. [CrossRef]
12. Darwish, A. Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Comput. Inform. J.* **2018**, *3*, 231–246. [CrossRef]
13. Nguyen, B.H.; Xue, B.; Zhang, M. A survey on swarm intelligence approaches to feature selection in data mining. *Swarm Evol. Comput.* **2020**, *54*, 100663. [CrossRef]
14. Khan, T.A.; Ling, S.H. A survey of the state-of-the-art swarm intelligence techniques and their application to an inverse design problem. *J. Comput. Electron.* **2020**, *19*, 1606–1628. [CrossRef]
15. Shaikh, P.W.; El-Abd, M.; Khanafer, M.; Gao, K. A Review on Swarm Intelligence and Evolutionary Algorithms for Solving the Traffic Signal Control Problem. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–16. [CrossRef]
16. Tzanetos, A.; Dounias, G. A Comprehensive Survey on the Applications of Swarm Intelligence and Bio-Inspired Evolutionary Strategies. In *Learning and Analytics in Intelligent Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 337–378. [CrossRef]
17. Gendreau, M.; Potvin, J.Y. *Handbook of Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 2.
18. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
19. Eiben, Á.E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **1999**, *3*, 124–141. [CrossRef]
20. Mohamed, M.A.; Eltamaly, A.M.; Alolah, A.I. Swarm intelligence-based optimization of grid-dependent hybrid renewable energy systems. *Renew. Sustain. Energy Rev.* **2017**, *77*, 515–524. [CrossRef]

21. Oliveto, P.S.; Paixão, T.; Pérez, J.; Sudholt, D.; Trubenová, B. How to Escape Local Optima in Black Box Optimisation: When Non-elitism Outperforms Elitism. *Algorithmica* **2017**, *80*, 1604–1633. [CrossRef] [PubMed]
22. Qi, X.; Ju, G.; Xu, S. Efficient solution to the stagnation problem of the particle swarm optimization algorithm for phase diversity. *Appl. Opt.* **2018**, *57*, 2747. [CrossRef] [PubMed]
23. Hamadi, Y.; Monfroy, E.; Saubion, F. What is autonomous search? In *Hybrid Optimization*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 357–391.
24. Jong, K.D. Parameter Setting in EAs: A 30 Year Perspective. In *Parameter Setting in Evolutionary Algorithms*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–18. [CrossRef]
25. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; IEEE: Piscataway, NJ, USA, 1999; Volume 3, pp. 1945–1950.
26. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evol. Comput.* **2004**, *8*, 240–255. [CrossRef]
27. Xu, G. An adaptive parameter tuning of particle swarm optimization algorithm. *Appl. Math. Comput.* **2013**, *219*, 4560–4569. [CrossRef]
28. Wang, F.; Zhang, H.; Li, K.; Lin, Z.; Yang, J.; Shen, X.L. A hybrid particle swarm optimization algorithm using adaptive learning strategy. *Inf. Sci.* **2018**, *436–437*, 162–177. [CrossRef]
29. Cao, Y.; Zhang, H.; Li, W.; Zhou, M.; Zhang, Y.; Chaovalitwongse, W.A. Comprehensive Learning Particle Swarm Optimization Algorithm with Local Search for Multimodal Functions. *IEEE Trans. Evol. Comput.* **2019**, *23*, 718–731. [CrossRef]
30. Yang, X.; Yuan, J.; Yuan, J.; Mao, H. A modified particle swarm optimizer with dynamic adaptation. *Appl. Math. Comput.* **2007**, *189*, 1205–1213. [CrossRef]
31. Wu, Z.; Zhou, J. A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. In Proceedings of the 2007 International Conference on Computational Intelligence and Security (CIS 2007), Harbin, China, 15–19 December 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 133–136.
32. Li, Z.; Tan, G. A self-adaptive mutation-particle swarm optimization algorithm. In Proceedings of the 2008 Fourth International Conference on Natural Computation, Jinan, China, 18–20 October 2008; IEEE: Piscataway, NJ, USA, 2008; Volume 1, pp. 30–34.
33. Li, X.; Fu, H.; Zhang, C. A self-adaptive particle swarm optimization algorithm. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December; IEEE: Piscataway, NJ, USA, 2008; Volume 5, pp. 186–189.
34. Dong, C.; Wang, G.; Chen, Z.; Yu, Z. A method of self-adaptive inertia weight for PSO. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December; IEEE: Piscataway, NJ, USA, 2008; Volume 1, pp. 1195–1198.
35. Chen, H.H.; Li, G.Q.; Liao, H.L. A self-adaptive improved particle swarm optimization algorithm and its application in available transfer capability calculation. In Proceedings of the 2009 Fifth International Conference on Natural Computation, Tianjin, China, 14–16 August 2009; IEEE: Piscataway, NJ, USA, 2009; Volume 3, pp. 200–205.
36. Nickabadi, A.; Ebadzadeh, M.M.; Safabakhsh, R. A novel particle swarm optimization algorithm with adaptive inertia weight. *Appl. Soft Comput.* **2011**, *11*, 3658–3670. [CrossRef]
37. Tanweer, M.R.; Suresh, S.; Sundararajan, N. Self regulating particle swarm optimization algorithm. *Inf. Sci.* **2015**, *294*, 182–202. [CrossRef]
38. Zhan, Z.H.; Zhang, J.; Li, Y.; Chung, H.S.H. Adaptive particle swarm optimization. *IEEE Trans. Syst. Ma, Cybern Part B (Cybern.)* **2009**, *39*, 1362–1381. [CrossRef] [PubMed]
39. Leu, M.S.; Yeh, M.F. Grey particle swarm optimization. *Appl. Soft Comput.* **2012**, *12*, 2985–2996. [CrossRef]
40. Julong, D. Introduction to grey system theory. *J. Grey Syst.* **1989**, *1*, 1–24.
41. Calvet, L.; Armas, J.; Masip, D.; Juan, A.A. Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Math.* **2017**, *15*, 261–280. [CrossRef]
42. Singh, P.; Singh, S. Energy efficient clustering protocol based on improved metaheuristic in wireless sensor networks. *J. Netw. Comput. Appl.* **2017**, *83*, 40–52. [CrossRef]
43. Alvarenga, R.D.; Machado, A.M.; Ribeiro, G.M.; Mauri, G.R. A mathematical model and a Clustering Search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas. *Comput. Ind. Eng.* **2016**, *101*, 303–312. [CrossRef]
44. Kuo, R.J.; Kuo, P.H.; Chen, Y.R.; Zulvia, F.E. Application of metaheuristics-based clustering algorithm to item assignment in a synchronized zone order picking system. *Appl. Soft Comput.* **2016**, *46*, 143–150. [CrossRef]
45. Kuo, R.J.; Mei, C.H.; Zulvia, F.E.; Tsai, C.Y. An application of a metaheuristic algorithm-based clustering ensemble method to APP customer segmentation. *Neurocomputing* **2016**, *205*, 116–129. [CrossRef]
46. Fong, S.; Wong, R.; Vasilakos, A. Accelerated PSO Swarm Search Feature Selection for Data Stream Mining Big Data. *IEEE Trans. Serv. Comput.* **2015**, *9*, 33–45. [CrossRef]
47. Chou, J.S.; Putra, J.P. Metaheuristic optimization within machine learning-based classification system for early warnings related to geotechnical problems. *Autom. Constr.* **2016**, *68*, 65–80. [CrossRef]

48. Al-Obeidat, F.; Belacel, N.; Spencer, B. Combining Machine Learning and Metaheuristics Algorithms for Classification Method PROAFTN. In *Enhanced Living Environments*; Lecture Notes in Computer Science; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 53–79. [CrossRef]
49. Chou, J.S.; Nguyen, T.K. Forward Forecast of Stock Price Using Sliding-Window Metaheuristic-Optimized Machine-Learning Regression. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3132–3142. [CrossRef]
50. He, S.; Li, Z.; Tang, Y.; Liao, Z.; Li, F.; Lim, S.J. Parameters Compressing in Deep Learning. *Comput. Mater. Contin.* **2020**, *62*, 321–336. [CrossRef]
51. Hashemi, A.B.; Meybodi, M. Adaptive parameter selection scheme for PSO: A learning automata approach. In Proceedings of the 2009 14th International CSI Computer Conference, Tehran, Iran, 1–2 July 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 403–411.
52. Ries, J.; Beullens, P. A semi-automated design of instance-based fuzzy parameter tuning for metaheuristics based on decision tree induction. *J. Oper. Res. Soc.* **2015**, *66*, 782–793. [CrossRef]
53. Salcedo-Sanz, S.; Yao, X. A Hybrid Hopfield Network-Genetic Algorithm Approach for the Terminal Assignment Problem. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2004**, *34*, 2343–2353. [CrossRef] [PubMed]
54. Zeng, B.; Li, X.; Gao, L.; Zhang, Y.; Dong, H. Whale swarm algorithm with the mechanism of identifying and escaping from extreme points for multimodal function optimization. *Neural Comput. Appl.* **2019**, *32*, 5071–5091. [CrossRef]
55. Sun, Y.; Gao, Y. A Multi-Objective Particle Swarm Optimization Algorithm Based on Gaussian Mutation and an Improved Learning Strategy. *Mathematics* **2019**, *7*, 148. [CrossRef]
56. Olivares, R.; Munoz, R.; Soto, R.; Crawford, B.; Cárdenas, D.; Ponce, A.; Taramasco, C. An Optimized Brain-Based Algorithm for Classifying Parkinson's Disease. *Appl. Sci.* **2020**, *10*, 1827. [CrossRef]
57. Liu, J.; Wang, W.; Chen, J.; Sun, G.; Yang, A. Classification and Research of Skin Lesions Based on Machine Learning. *Comput. Mater. Contin.* **2020**, *62*, 1187–1200. [CrossRef]
58. Haoxiang, S.; Changxing, C.; Yunfei, L.; Mu, Y. Cooperative perception optimization based on self-checking machine learning. *Comput. Mater. Contin.* **2020**, *62*, 747–761. [CrossRef]
59. Zhou, S.; Chen, L.; Sugumaran, V. Hidden Two-Stream Collaborative Learning Network for Action Recognition. *Comput. Mater. Contin.* **2020**, *63*, 1545–1561. [CrossRef]
60. Zhou, S.; Tan, B. Electrocardiogram soft computing using hybrid deep learning CNN-ELM. *Appl. Soft Comput.* **2020**, *86*, 105778. [CrossRef]
61. Munoz, R.; Olivares, R.; Taramasco, C.; Villarroel, R.; Soto, R.; Alonso-Sánchez, M.F.; Merino, E.; de Albuquerque, V.H.C. A new EEG software that supports emotion recognition by using an autonomous approach. *Neural Comput. Appl.* **2018**, *32*, 11111–11127. [CrossRef]
62. Munoz, R.; Olivares, R.; Taramasco, C.; Villarroel, R.; Soto, R.; Barcelos, T.S.; Merino, E.; Alonso-Sánchez, M.F. Using Black Hole Algorithm to Improve EEG-Based Emotion Recognition. *Comput. Intell. Neurosci.* **2018**, *2018*, 3050214. [CrossRef]
63. Gui, Y.; Zeng, G. Joint learning of visual and spatial features for edit propagation from a single image. *Vis. Comput.* **2019**, *36*, 469–482. [CrossRef]
64. Santos, M.A.; Munoz, R.; Olivares, R.; Filho, P.P.R.; Ser, J.D.; de Albuquerque, V.H.C. Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook. *Inf. Fusion* **2020**, *53*, 222–239. [CrossRef]
65. Díaz, F.D.; Lasheras, F.S.; Moreno, V.; Moratalla-Navarro, F.; de la Torre, A.J.M.; Sánchez, V.M. GASVeM: A New Machine Learning Methodology for Multi-SNP Analysis of GWAS Data Based on Genetic Algorithms and Support Vector Machines. *Mathematics* **2021**, *9*, 654. [CrossRef]
66. Minonzio, J.G.; Cataldo, B.; Olivares, R.; Ramiandrisoa, D.; Soto, R.; Crawford, B.; Albuquerque, V.H.C.D.; Munoz, R. Automatic Classifying of Patients With Non-Traumatic Fractures Based on Ultrasonic Guided Wave Spectrum Image Using a Dynamic Support Vector Machine. *IEEE Access* **2020**, *8*, 194752–194764. [CrossRef]
67. Streichert, F.; Stein, G.; Ulmer, H.; Zell, A. A Clustering Based Niching Method for Evolutionary Algorithms. In *Genetic and Evolutionary Computation—GECCO 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 644–645. [CrossRef]
68. Valdivia, S.; Soto, R.; Crawford, B.; Caselli, N.; Paredes, F.; Castro, C.; Olivares, R. Clustering-Based Binarization Methods Applied to the Crow Search Algorithm for 0/1 Combinatorial Problems. *Mathematics* **2020**, *8*, 1070. [CrossRef]
69. Santos, H.G.; Ochi, L.S.; Marinho, E.H.; Drummond, L.M. Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing* **2006**, *70*, 70–77. [CrossRef]
70. Jin, Y.; Qu, R.; Atkin, J. A Population-Based Incremental Learning Method for Constrained Portfolio Optimisation. In Proceedings of the 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 22–25 September 2014; IEEE: Piscataway, NJ, USA, 2014; p. 7031476. [CrossRef]
71. Nurcahyadi, T.; Blum, C. Adding Negative Learning to Ant Colony Optimization: A Comprehensive Study. *Mathematics* **2021**, *9*, 361. [CrossRef]
72. Rasmussen, T.K.; Krink, T. Improved Hidden Markov Model training for multiple sequence alignment by a particle swarm optimization—Evolutionary algorithm hybrid. *Biosystems* **2003**, *72*, 5–17. [CrossRef]
73. Prakash, A.; Chandrasekar, C. An Optimized Multiple Semi-Hidden Markov Model for Credit Card Fraud Detection. *Indian J. Sci. Technol.* **2015**, *8*, 165. [CrossRef]

74. Xue, L.; Yin, J.; Ji, Z.; Jiang, L. A particle swarm optimization for hidden Markov model training. In Proceedings of the 2006 8th International Conference on Signal Processing, Guilin, China, 16–20 November 2006; IEEE: Piscataway, NJ, USA, 2006; Volume 1, p. 345542.
75. Aoun, O.; Sarhani, M.; El Afia, A. Hidden markov model classifier for the adaptive particle swarm optimization. In *Recent Developments in Metaheuristics*; Operations Research/Computer Science Interfaces Series; Springer: Berlin/Heidelberg, Germany, 2018; pp. 1–15.
76. Wang, X.; Chen, H.; Heidari, A.A.; Zhang, X.; Xu, J.; Xu, Y.; Huang, H. Multi-population following behavior-driven fruit fly optimization: A Markov chain convergence proof and comprehensive analysis. *Knowl. Based Syst.* **2020**, *210*, 106437. [CrossRef]
77. Motiian, S.; Soltanian-Zadeh, H. Improved particle swarm optimization and applications to Hidden Markov Model and Ackley function. In Proceedings of the 2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings, Ottawa, AB, Canada, 19–21 September 2011; IEEE: Piscataway, NJ, USA, 2011; p. 6045560. [CrossRef]
78. Sagayam, K.M.; Hemanth, D.J. ABC algorithm based optimization of 1-D hidden Markov model for hand gesture recognition applications. *Comput. Ind.* **2018**, *99*, 313–323. [CrossRef]
79. Trindade, Á.R.; Campelo, F. Tuning metaheuristics by sequential optimisation of regression models. *Appl. Soft Comput.* **2019**, *85*, 105829. [CrossRef]
80. Wei, Z.; Yong, Z.; Chen, L.; Lei, G.; Wenpei, Z. An improved particle swarm optimization algorithm and its application on distribution generation accessing to distribution network. In Proceedings of the IOP Conference Series: Earth and Environmental Science, Malang, Indonesia, 12–13 March 2019; Volume 342, p. 012011. [CrossRef]
81. Crawford, B.; Soto, R.; Monfroy, E.; Palma, W.; Castro, C.; Paredes, F. Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. *Expert Syst. Appl.* **2013**, *40*, 1690–1695. [CrossRef]
82. Pellegrini, P.; Stützle, T.; Birattari, M. A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intell.* **2012**, *6*, 23–48. [CrossRef]
83. Beni, G.; Wang, J. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?* Springer: Berlin/Heidelberg, Germany, 1993; pp. 703–712.
84. Beni, G. Swarm intelligence. In *Complex Social and Behavioral Systems: Game Theory and Agent-Based Models*; 2020; pp. 791–818. Available online: https://www.springer.com/gp/book/9781071603673 (accessed on 25 May 2021).
85. Zhu, H.; Wang, Y.; Ma, Z.; Li, X. A Comparative Study of Swarm Intelligence Algorithms for UCAV Path-Planning Problems. *Mathematics* **2021**, *9*, 171. [CrossRef]
86. Freitas, D.; Lopes, L.G.; Morgado-Dias, F. Particle Swarm Optimisation: A Historical Review Up to the Current Developments. *Entropy* **2020**, *22*, 362. [CrossRef]
87. Khare, A.; Rangnekar, S. A review of particle swarm optimization and its applications in Solar Photovoltaic system. *Appl. Soft Comput.* **2013**, *13*, 2997–3006. [CrossRef]
88. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
89. Erwin, K.; Engelbrecht, A. Diversity Measures for Set-Based Meta-Heuristics. In Proceedings of the 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI), Stockholm, Sweden, 14–15 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 45–50.
90. Gavira-Durón, N.; Gutierrez-Vargas, O.; Cruz-Aké, S. Markov Chain K-Means Cluster Models and Their Use for Companies' Credit Quality and Default Probability Estimation. *Mathematics* **2021**, *9*, 879. [CrossRef]
91. Naranjo, L.; Esparza, L.J.R.; Pérez, C.J. A Hidden Markov Model to Address Measurement Errors in Ordinal Response Scale and Non-Decreasing Process. *Mathematics* **2020**, *8*, 622. [CrossRef]
92. Koike, T.; Hofert, M. Markov Chain Monte Carlo Methods for Estimating Systemic Risk Allocations. *Risks* **2020**, *8*, 6. [CrossRef]
93. El Afia, A.; Sarhani, M.; Aoun, O. Hidden markov model control of inertia weight adaptation for Particle swarm optimization. *IFAC-PapersOnLine* **2017**, *50*, 9997–10002. [CrossRef]
94. El Afia, A.; Aoun, O.; Garcia, S. Adaptive cooperation of multi-swarm particle swarm optimizer-based hidden Markov model. *Prog. Artif. Intell.* **2019**, *8*, 441–452. [CrossRef]
95. Rabiner, L.; Juang, B. An introduction to hidden Markov models. *IEEE ASSP Mag.* **1986**, *3*, 4–16. [CrossRef]
96. Beasley, J. An algorithm for set covering problem. *Eur. J. Oper. Res.* **1987**, *31*, 85–93. [CrossRef]
97. Harrison, K.R.; Engelbrecht, A.P.; Ombuki-Berman, B.M. Self-adaptive particle swarm optimization: A review and analysis of convergence. *Swarm Intell.* **2018**, *12*, 187–226. [CrossRef]
98. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Comput. Appl.* **2018**, *31*, 7665–7683. [CrossRef]
99. Mattiussi, C.; Waibel, M.; Floreano, D. Measures of Diversity for Populations and Distances Between Individuals with Highly Reorganizable Genomes. *Evol. Comput.* **2004**, *12*, 495–515. [CrossRef]
100. Cheng, S.; Shi, Y.; Qin, Q.; Zhang, Q.; Bai, R. Population Diversity Maintenance In Brain Storm Optimization Algorithm. *J. Artif. Intell. Soft Comput. Res.* **2014**, *4*, 83–97. [CrossRef]

101. Morales-Castañeda, B.; Zaldivar, D.; Cuevas, E.; Fausto, F.; Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm Evol. Comput.* **2020**, 54, 100671. [CrossRef]
102. Crawford, B.; Soto, R.; Astorga, G.; García, J.; Castro, C.; Paredes, F. Putting Continuous Metaheuristics to Work in Binary Search Spaces. *Complexity* **2017**, *2017*, 8404231. [CrossRef]