

From Selecting Best Algorithm to Explaining Why It is: A General Review, Formal Problem Statement and Guidelines Towards to an Empirical Generalization

Vanesa Landero Nájera[1(✉)], Joaquín Pérez Ortega[2], Carlos Andrés Collazos Morales[3], and Sandra Silvia Roblero Aguilar[4]

[1] Universidad Politécnica de Apodaca, El Barretal, Nuevo León, México
vlandero@upapnl.edu.mx
[2] Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Departamento de Ciencias Computacionales, AP 5-164, Cuernavaca 62490, México
jperez@cenidet.edu.mx
[3] Universidad Manuela Beltrán, Bogotá, Colombia
[4] Instituto Tecnológico de Tlalnepantla, Ciudad de México, México

Abstract. It has been observed on solution algorithms for problems as sorting, forecasting, classification, clustering, constraint satisfaction, decision, optimization from several disciplines (computational complexity theory, data mining, artificial intelligence, machine learning, operations research) that algorithm performance is better in certain problem instances than other. This paper describes how has been the way for trying to reach the empirical generalization for this phenomenon existing in the experimental relation problem – algorithm. For each understanding level, research questions, problem description were formulated, using the same Rice's nomenclature and supplementing it; as well as, influence indexes and analysis approaches were described. A diagram about this long trajectory and a reflection is performed, highlighting contributions and scope. It shows that up to now the problem of explaining formally why an algorithm is the best for solving an instance set had remained open. A formal problem statement for describing this phenomenon and a general framework were proposed as a guide for working in adequate way toward generation of theories; which could contribute to build generalized indexes and self-adaptive algorithms to give the best solution to problems.

1 Introduction

There exist many approximation algorithms have been proposed to give solution to problems as constraint satisfaction, decision, optimization, forecasting, classification, clustering, sorting; from different application domains, to name a few: computational complexity theory, operations research, data mining, machine learning, artificial intelligence, bioinformatics. It is observed from the experimental relation problem-algorithm that algorithm performance is the best in a problem instances set and worst in another [1–5]. According to this phenomenon observed by specialized literature, the related works

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 694–712, 2023.
https://doi.org/10.1007/978-3-031-36805-9_45

in the majority of cases have focused in: predicting of algorithm performance (runtime) [6] or tuning parameter in an automatic way [7–12], or predicting the instances hardness [13], or building hyperheuristics [14, 15], or selecting the algorithm more appropriated in solving a problem instance [16–23]. However, these fall on predictive understanding level without obtaining formally a deeper understanding of why this phenomenon occurs [24, 25]. Cohen described in his book [26] a general diagram about empirical generalization for the analysis of behavior of an observed phenomenon. It evolves through three levels: description (D), prediction (P) and explanatory (E). The understanding and generalization were considered. It would be interesting to know how has been the way performed by reviewed literature through these levels. The focus and main contributions of this paper are next. A review of state of art and a diagram of understanding levels were performed. A formal research question is stated for each level where the specialized literature, as a sample of an extended population of related works is located with relevant indexes and applied analysis approaches. A reflection about this sample of trajectory is performed (there is not a results quantitative comparison due to diversity to before mentioned) highlighting only advantages, disadvantages and contributions. It is noteworthy that the algorithm selection problem ASP is formally formulated in predictive level and worked by several works. However, it would be interesting to know beyond this problem and understand better the phenomenon. A formal problem statement for describing it in the explanatory level is proposed, using and supplementing the Rice's nomenclature. Finally, a general framework is presented as a guide to continue the way towards the empirical generalization; where found theories can permit build generalized indexes and self-adaptive algorithms that solve efficiently real problems.

2 A General Review: Understanding Levels, Influence Indexes and Analysis Approaches

In this section a few works will be mentioned (some other in Sect. 3) with the sole purpose of exemplifying the understanding levels and the principal analysis approach applied. The Rice's formal nomenclature is highlighted, slightly modified and supplemented for describing questionings and kind of relevant information.

2.1 Formal Nomenclature

$\boldsymbol{P} = \{x_1, x_2, \ldots, x_m\}$ a set instances of problem domain **P**, the space for analysis.
$\boldsymbol{F} =$ the problem features or indexes space generated by a description process to $\boldsymbol{P}$.
$\boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}$ a set of n algorithms.
$\boldsymbol{Y} =$ the performance space, it represents the mapping of each algorithm to a set of performance metrics.
$\boldsymbol{C} = \{C_1, C_2, \ldots, C_n\}$ a partition of $\boldsymbol{P}$, where $|\boldsymbol{A}|=|\boldsymbol{C}|$.
$\boldsymbol{W} = \{(a_q \in \boldsymbol{A}, C_q \in \boldsymbol{C}) \mid Y_{aq,x} > Y_{\alpha,x}\ \forall\ \alpha \in (\boldsymbol{A} - \{a_q\}), \forall\ x \in C_q\}$, is a set of domains, ordered pairs (a_q, C_q), where each dominant algorithm $a_q \in \boldsymbol{A}$ is associated with one element C_q of partition $\boldsymbol{C}$, because this gives the best solution to partition C_q, considering set $\boldsymbol{Y}$.
$\boldsymbol{L} =$ the algorithm features or indexes space generated by a description process (Operative or/and searching behaviour) applied to $\boldsymbol{A}$.

2.2 Descriptive Understanding Level (D): The Phenomenon is Observed

The interest has been to describe the algorithm performance. Therefore,

1. What is the performance of algorithm $a_q \in \boldsymbol{A}$ to solve the problem P?

The performance can be measured, depending on application domain, by means time functions, for computing the run time or quality functions, for measuring the found solution quality; as well as other functions for obtaining iterations number, evaluations number, prediction accuracy, classification accuracy, between others. Once the performance metrics set is identified $y \in \boldsymbol{Y}$, a set of algorithms A is run over a set of problem instances $\boldsymbol{P}$ and their performances are compared in some way. One classical example for the comparison is by means some statistical analysis (statistical tests The Sign, Wilcoxon and Friedman tests, among others) or tabular analysis or graphical analysis [27]. However, the average efficiency is compared without considering its dispersion. Reeves [28] commented that a heuristic with good averaged performance, but with high dispersion, has a very high risk to show a poor or low performance in many instances. It is suggested as alternative to formulate for each algorithm, a utility function adjusted to a gamma distribution, whose parameters permit to compare the heuristics on a range of risk value. Another related work identified different performances of algorithms on problem instances different sets, where each set have one or several similar features in the context of the problem structure description; for example, particularly the performance of different algorithms was projected on a wide range of instances specifications of the Constraint Satisfaction Problem (CSP) [24]. The instances are characterized by the parameters $< w, x, y, z >$, which describe in a simplified way the structure of the problem. The process consists in generating instances sets, so that, the instances in a same set have the similar values of their relevant parameters. A map is built, which shows the algorithm with the best average performance for each set of instances. In another similar work, the relation between the performance of a backtracking algorithm and instances generated randomly of Constraint Satisfaction Problem (CSP) was modeled by two standard families of functions of continuous probability distribution, when this algorithm solves them [29]. It is to say, the resoluble instances can be modeled by the Weibull distribution and not resoluble instances by the lognormal distribution. Other papers suggest somehow characterize problem descriptive structure (meta-features), showing that the performance of algorithms on characterized instances is better than those are not characterized [30]. In the same way, other related works make changes in any of the four fundamental aspects of algorithm internal logic structure for improving the algorithms performance. It is, to redesign the methodology for: tuning parameter, generating initial solution, searching and obtaining solutions, and stopping the algorithm execution (divergence or convergence). Some related works are [31–33]. Although it is observed that the new proposal has a better performance in most instances of the problem with respect to the existing algorithms, there is always a set of instances where each existing algorithm was better at solving them. This phenomenon observed by the reviewed literature and No Free Lunch theorems [2] could be described by means of a dominions partition of the algorithms A. It is, a set $\boldsymbol{W}$ of pairs in the form (a_q, C_q), where one algorithm a_q correspond better to an instances subset C_q of specific problem P (see nomenclature for a major compression).

2.3 Predictive Understanding Level (P)

The phenomenon was observed in the before level (described by means the set $\boldsymbol{W}$), an interesting question in the predictive understanding level arises:

> 2. What is the mechanism to learn W (dominions partition, pairs (a_q, C_q)) in order to predict algorithm $a_q \in \boldsymbol{A}$ that will give the best solution for an instance $x \in \boldsymbol{P}$ with features $f(x) \in \boldsymbol{F}$?

One formal problem statement in this level, similar to above questioning, is the algorithm selection problem ASP [34] which includes a selection mapping (a mechanism for learning set $\boldsymbol{W}$ and predicting). Miles describes a generalization of this problem through different research disciplines [35], its solution is important. It is stated as:

> For a given problem instance $x \in \boldsymbol{P}$, with features $f(x) \in \boldsymbol{F}$, find the selection mapping $\boldsymbol{S}(f(x))$ into algorithm space $\boldsymbol{A}$, such that the selected algorithm $\propto \in \boldsymbol{A}$ maximizes the performance mapping $y(\propto(x)) \in \boldsymbol{Y}$.

The consulted literature that has worked in a similar way to the above have obtained the knowledge in terms of prediction (selection mapping) in various guidelines. It is to say, depending of kind of features or indexes with predictive value (space of the problem (F) and the algorithm (L)) as well as the approach applied (a mechanism for learning set $\boldsymbol{W}$ and predicting). Some approaches will be described of a general manner and exemplary; for more detail and formal description about these and other as regression analysis, functions of probability distribution and other related works can be found in [6, 14, 35–41].

Functions of Rules. Some related works have proposed rules by means functions to select the more appropriate algorithm. The majority of these functions are simple rules about problem structural features that is being solved, or about history of performance in past experimentations or about the computational environment. As well as, the human experience could contribute to definition of these rules. Some related works [42, 43] were exemplified by [14]. Another work is [4], where a setup time dominance matrix was developed considering results from graphical and statistical analysis; it is used for selecting the algorithm for a problem particular instance size.

Montecarlo Approach. Some works are based in a short execution of an algorithm on an instance to obtain a representative sample of statistical data; which can predict the algorithm general behaviour on this instance. In the work Knuth [44], a sample is taken by means of the exploration of a way from the root of the tree to a leaf drawn up by Chronological Backtracking algorithms. The way is randomly chosen. The statistics of interest are the cost of processing a node and the number of children of the node. If the number of children of level i is d_i then: $(1) + (d_1) + (d_1 \times d_2) + \ldots + (d_1 \times \ldots \times d_n)$ is considered an approximation of the number nodes of the tree. If the cost of processing of each node in the level i is c_i, then $c_1 + c_2\ (d_1) + c_3\ (d_1 \times d_2) + \ldots + c_n + 1\ (d_1 \times \ldots \times d_n)$ is considered an approximation of the cost of processing all the tree nodes. Purdom [45] allows the exploration of more than a path, and receives as input the number of children that will explore themselves from each visited node. The method proposed

in [46] for algorithms of Backward movement with Propagation of Restrictions, allows the complete exploration of the tree until certain level of depth, and from there, the number of nodes is calculated by applying a sampling similar to the one of Purdom. This technique is applied to algorithms of Backward movement with Propagation of Restrictions and Tabu Search algorithms in [47] and algorithms of Ramification and Demarcation in [48]. Another related work was [49], where the algorithm selection problem is focused on minimization problem of execution total time, which is solved with a Reinforced Learning algorithm (RL). The well-known Q-learning algorithm is adapted for this case in a way that combines both Monte-Carlo and Temporal Difference methods. Two classical problems were focused: selecting and ordering. A function that predicts the best algorithm for a new instance using its problem size is determined by means of training. It combines several recursive algorithms to improve its performance: the actual problem is divided in subproblems in each recursive step, and the most adequate algorithm in size is used for each of them; it is extended to backtracking algorithms to SAT problem in [3].

Algorithm Portfolio. The strategy of running several algorithms in parallel, potentially with different algorithms being assigned different amounts of CPU time, or when some stop criterion is satisfied [50]. The performance of all algorithms is measured by means some function, so too, it is characterized and adjusted to a model (model-based portfolio). For example, a probability distribution model [36]. The framework employs three core components: a portfolio of algorithms (SAT solvers); a generative model, which is fit to data on those algorithms' past performance, then used to predict their future performance; and a policy for action selection, which repeatedly chooses algorithms based on those predictions. The models of solver behaviour are two latent class models: a multinomial mixture model of the outcomes of solver-duration pairs, and a mixture model DCM (Dirichlet compound multinomial) that additionally captures the burstiness of those outcomes. Another example is [21], where a multiple evolutionary algorithm portfolio to solve multiobjective optimization problems is proposed for solving benchmark functions: bi-objective, tri-objective, five-objective. Two principal parts are included for each population generation. The first part, for each multiobjective evolutionary algorithm (MOEA), its score is computed. The second part, a bootstrap probability distribution is built and predicted performance is obtained at a common future point and selects the algorithm more appropriated to generate the next population and so on. The algorithms switch automatically as a function of the computational budget. Another guideline of approach is to consider instances features to build the model (feature-based portfolio). Classical examples, a supervised learning model was built by decision tree algorithm in [18], a regression model including parameter control information to select the best in [7]; other recent related works [51–53].

Machine Learning (Supervised Learning). The objective is to learn patterns identified from data with predictive value by means a supervised learning model and use it to make predictions about unseen data. In the majority cases, the relevant information was highlighted by means mathematical or statistical indexes. Some examples. For solving classification problems, in terms of problem description, proportion of symbolic attributes, the mean correlation coefficient for any two continuous attributes, mean mutual information, mean of the mutual information for each discrete attribute and target

class, class entropy, median entropy of attribute, median of entropies of all attributes [54]; a k-Nearest Neighbour algorithm was applied. For clustering problems, the mean and deviation standard of eigenvectors from image histograms [17]; the algorithm Support Vector Machine was applied. For sorting problems, the size of input permutation and presortedness measures [19]; a decision tree algorithm was applied. For optimization problems, in the problem Bin-Packing in one dimension, the weight's dispersion and the relation between the weights and container capacity; a decision tree algorithm was applied [20]. For problem Quasigroup with Holes (QWH), the number of holes in rows and columns, as well as, in terms of algorithm operative behaviour were the variance of uncolored cells number in the QWH instance across rows and across columns, the ratio of the number of uncolored cells and the number of columns or rows, the minimum depth of all leaves of the search tree; Neural Networks were applied [55]. For MIP problem, some of problem description were number of variables and constraints, features of graphs (variable, variable-constraint), as well as, features about algorithm operative behaviour, it is, the CPLEX with default settings is ran by 5 s runs for obtain indexes about presolving features, cut usage and probing result; also, during the searching, the behaviour is characterized by the tour length of local minima, the tour quality improvement per search step, number of search steps to reach a local minimum, distance between two local minima, probability of edges appearing in any local minimum encountered [6, 8]. For decision problem SAT, the number of clauses and variables, the clause and variable node degree statistics, as well as information about problem space, algorithm operative behaviour and searching behaviour; it is, the SAPS and GSAT algorithms were ran in two seconds of each run for obtaining mean and variation coefficient of the number of unsatisfied clauses in each local minimum and other indexes related to confidence of variables and the probability of unconstrained variables; as well as, indexes represent information about number of steps to the best local minimum, the average improvement to best in a run and the fraction of improvement due to first local minimum [6, 8]; the Random Forest algorithm was applied. There are other works where the information about parameter tuning was considered to build a model and select the best algorithm, some interesting works are: applying Bayesian Networks [9, 10], Decision trees [11, 57]. Finally, other related works and supervised learning algorithms that were applied: Case-Base reasoning [10]; k-nearest neighbour [21, 57], Random Forest [22, 56], and Neural Networks [57–60].

Hyper-Heuristic. A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems [15, 38]. It is to say, produce an adequate combination of available low-level heuristics in order to effectively solve a given instance by means a built high-level algorithm that can select and apply an appropriate low-level heuristic or metaheuristic at each decision step. One classical example to solve the SQRP problem [14]; the goal of SQRP is locating information in a network based on a query formed by keywords, within the shortest path. Each query traverses the network, moving from the initiating node to a neighbouring node and then to a neighbour of a neighbour and so forth, until it locates the requested resource. A hyper-heuristic algorithm is proposed to automatically select, among basic-heuristics, the most promising to adjust a parameter control of an Ant Colony Optimization algorithm for routing messages. The basic-heuristics are different ways to adjust the time

to live (TTL) parameter during the execution of the algorithm. The search process for the next node is based on the pheromone table and information about the distances, the successes of past queries and number of documents which are the closest nodes that can satisfy the query. The selection process of basic-heuristics is based on data structures that store information of prior consultations about hyper-heuristic visibility states, which allows the hyper-heuristic know if is necessary to add more TTL, because the number of resources found are few and decreases the lifetime. Another recent work [61].

2.4 Explanatory Understanding Level

In above level a predictive knowledge (model) is learned from hidden patterns existing in relation problem-algorithm; it is to say, a mapping is found, a W set learned by means some approach (mechanism) applied to a data set with predictive value. The obtained model is used to select the best algorithm. In this level, delving deeper into the knowledge learned, the following question arises.

3. What is the mechanism to understand why for each pair $(a_q, C_q) \in \boldsymbol{W}$, a problem instance subset C_q correspond better to an algorithm a_q than other instances $(C_q)^c$ in a specific problem domain?

Some related works have performed graphical or statistical analysis or data exploratory analysis, or developing some visual tool, mechanisms to interpret according their own knowledge the results and obtain explanations about algorithm performance: [62–68]. There are other related works [25, 50, 57, 69–71] have reached explicative knowledge by means these approaches in conjunction with other: Transition Phase, Markov Chains Model, Supervised Learning, Causal Analysis. For more detail and other works can be found in [24, 50, 72–78].

Transition Phase Analysis. The algorithm performance is analysed by means graphical analysis of a relevant index that in the majority of the cases is about the problem structure description. So that a particular value of this index (phase boundary) the algorithm performance has a strong fluctuation of its value, either a decrease or an increase. This change is known as transition point. This boundary is a type of phase transition and it must be preserved under mappings between problems. For more detail and formal description about this approach can be found in [79]. A conjecture about this approach was suggested by [72]: *All NP-complete problems have at least one order parameter and the hard to solve problems are around a critical value of this order parameter. This critical value (a phase transition) separates one region from another, such as overconstrained and underconstrained regions of the problem space. In such cases, the phase transition occurs at the point where the solution probability changes abruptly from almost zero to almost 1*. A very known application of this approach has been for SAT decision problem, where explanations about the performance of several SAT solvers were obtained; a relation between an index of the problem structure description (c/v) and the algorithm runtime was found [72, 80]. It is to say, a phase transition was observed from p(c, v) when the index c/v crosses a critical value of about 4.26, where v is held constant; instances with many clauses are overconstrained and almost always unsatisfiable, so too, instances with few clauses are underconstrained and almost always satisfiable. For several SAT

solvers, when all tested values of v are fixed, even for the SAT solvers that perform best on these instances, the phase transition point occurs between these extremes when p(c, v) is exactly 0.5. Other examples, for optimization problems, the Quasigroup Completion Problem (QCP) shows for random instances that there is a peak in terms of problem hardness over a range of values for an index that represent the ratio of the number of uncoloured cells to the total number of cells [55]. So too, for the Quasigroup with Holes Problem (QWH), its problem instances hardness is marked significantly by structural properties. It is, a pattern representing a balance of the number holes in each row and column of instances. Later other related works identified interesting observations about these discoveries, which is necessary to explore more features performing a more deeply analysis [13, 37].

Markov Chains Model. The algorithm behaviour is characterized in terms of search trajectory to understand why algorithm dynamics are so effective on problem instances and deduce how these dynamics might be modified to yield further improvements in performance. For example, a Markov model is developed for studying the run-time behaviour of Tabu Search algorithm when it solves the job-shop scheduling problem (JSP) [69]. It is to say, the cost of locating both optimal and sub-optimal solutions to a wide range of problem instances. The performance was analysed (graphical statistical analysis) using different methods for constructing the initial solution and different tabu tenure sizes. Explanations are described in Sect. 3.

Machine Learning (Supervised and Unsupervised Learning). Patterns are recognized from data, by means a supervised learning model. Relations between dependent variables are identified and analysed; key variables with respect some independent variables of interest are explained. A set of algorithms has been applied to carry out the above, some related works are K-means, Self-organizing maps [81], Decision trees [58, 82]. The kind of information utilized is about the problem description. One example, the rules generated in [58] indicated that cluster ratio seemed be a boundary for phase transition, it is around of 0.05; as well as, the fraction of distinct distances around 1.4% as a phase transition parameter [58].

Causal Analysis. This approach consists, in general terms, of identifying the principal causes of the behaviour of some phenomenon, representing them in the form of relations cause-effect in a causal model. A causal model consists of a Directed Acyclic Graph (DAG) over a set $V = \{V_1,\ldots,V_n\}$ of vertices, representing variables of interest, and a set E of directed edges, or arrows, that connect these vertices. These graphs can be causally interpreted if they have the properties: Causal Markov condition, Minimality condition, Faithfulness condition. These properties make the connection between causality and probability [74]. Causal modelling has four phases. The first phase is structure learning, where its objective is to find the principal structure of causal relations between several relevant indexes and a data set; it is to say, a causal graph G is found that represents the causal structure $G = (V, E)$; there exist several algorithms of structure learning, one of them is PC algorithm (Peter&Clark). The second phase is estimation of the found causal relations; one of algorithms that perform it is Counting algorithm [83]. The third phase is the interpretation of causal model; the causal relations with higher magnitudes are analysed and interpreted. The fourth phase is the causal model validation. Some

examples of related works performed this kind of analysis can be found in [70, 71, 84–86].

3 A Trajectory Diagram and General Discussion

The Fig. 1 shows a diagram of trajectory achieved (TA) by some works related in the analysis of the experimental relation between problem and algorithm. It is according to: the grade of the contribution to the science, the understanding level (Descriptive (D), Predictive (P), Explanatory (E)); the kind of information (problem (F), algorithm behaviour (***L***) indexes, algorithm performance (***Y***)); and principal approaches to perform the analysis in each understanding level. Focusing on recent works, firstly on predictive level and later on explanatory level; it can be seen in the figure, considering problem information (***F***), some related works included only one feature or index that characterizes the problem structure description (*). For example, the problem size feature has been considered for building a model, using the regression analysis approach, and relating it with the algorithm performance [87]. However, it is well known that it is not enough, there exists the necessity to explore more features or indexes [13, 82]. In this context, the research results of some related works [19, 54, 58, 59, 88] build formal models, using machine learning, to select the most appropriate algorithm to solve better a problem instance, where it considers several features or indexes of the problem structure (*) which adjust better to this selected algorithm; other works, included specifically indexes about problem solutions space (+), also known as problem landscape features [56, 60] to establish a relation between these features (+) and algorithm performance. Some similar to this type of information was included along with problem structure (*) years ago by work [16]; the feature-based algorithm portfolio approach with regression was used. Now, in the context of considering only algorithm information (***L***) for building an algorithm [52], in the portfolio approach or in the hyper-heuristic approach [14, 61] or tunning parameter [9], learning configurations (⬥) with the machine learning approach for giving the best solution to a problem instance. Also, this approach was used by [55], where the algorithm operative behaviour (Δ) was learned in a model to predict the best solver to a SAT problem instance. The principal advantage, in terms only of (***L***), is the independence to problem description, which can be applied to any problem. There were works that included both, problem (F) and algorithm (L) information. The problem structure description (*) was mainly used in conjunction with information about algorithm configuration (⬥) [7, 10, 11, 51] or algorithm searching behaviour (∇) [22], or with all of the above and algorithm operative behaviour (Δ) [6]. The information about the problem space (+) was mainly used in conjunction with information about algorithm searching behaviour (∇), again with the approaches portfolio with regression and machine learning [53, 60], for giving the most adequate algorithm. However, there is still the general question from the explanatory understanding level "why an algorithm is best for a certain set of instances of a problem and why not in another". The problem structure index (*) in the form (c/v ratio) within transition phase approach for SAT problem [72, 80], has helped to indicate when the instances are hard or easy for SAT solvers; so too, for TSP problem [58], a decision tree defined rules with indexes calculated from distances: between cities, normalized nearest neighbour; ratio of cluster and outliers for identifying hard or easy instances to heuristics. Another recent work, the problem

structure index (*), proportional to number of variable interactions (k/n) was used in conjunction with problem space features (+) relating to autocorrelation of random walks for multi-objective optimization problems [82]; it is found by means statistical analysis (regression) that if the index increases then the landscape ruggedness too (knowledge found also in single-objective optimization problems). So too, the problem structure index (*), correlation between objective functions (p) impacts to algorithm performance. The problem space features (+) were included by a decision tree for indicating what algorithm performs better in one instance (Global Simple Evolutionary Multi-Objective Optimizer and Iterative Pareto local search algorithms). As future work, the information about the algorithm internal logic structure will be considered for a better design of algorithms Evolutionary Multi-Objective (EMO).

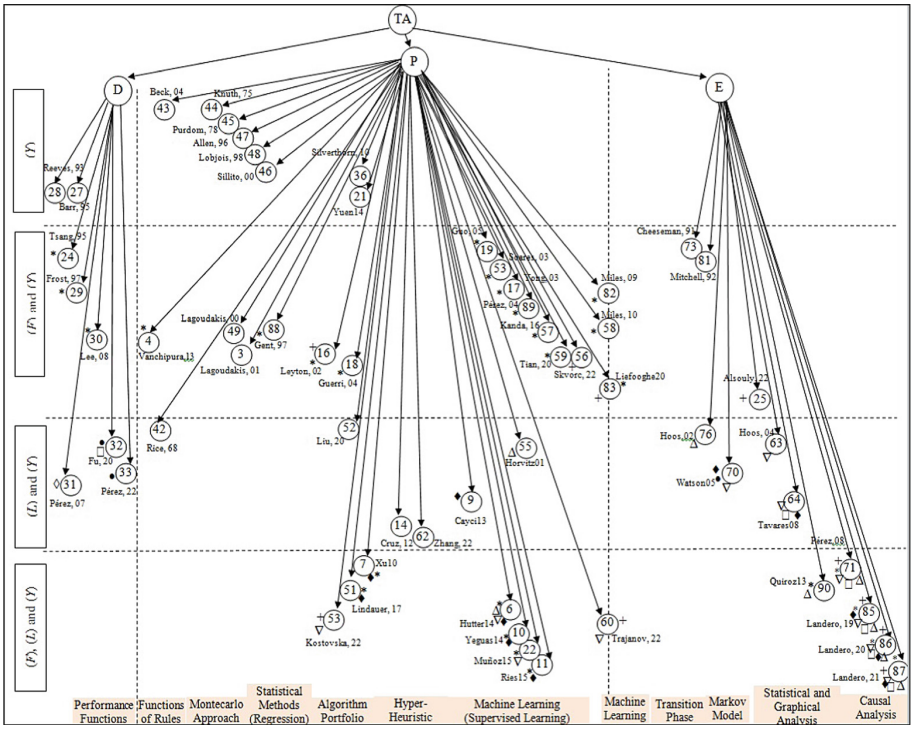

Fig. 1. Trajectory of consulted specialized literature

Another similar recent work [25], includes constraints to problem, it found, by means of the Instance Space Analysis methodology (ISA), two indexes about problem space (+). The isolation of non-dominate set and the correlation between constraints and objectives permitted explanations about the algorithm performance. The algorithm CCMO is more appropriated to converge on an isolated optima and the algorithm PPS when there is a large optimal set and there is the necessity to generate more diversity. So too, if there is negative correlation between constraints and objectives then most of analyzed CMOEA algorithms will fail to give high quality solutions. However, the analysis is

looked only from the side of the problem. Other works viewed the instances hardness from the algorithm's viewpoint, it is analyzed, the algorithm internal logic structure and indexes that characterize the operative behaviour (Δ) [75]. In [69], the searching behaviour (∇) is modelled by a Markov process and indicated that the method for generating initial solution not impact the algorithm performance (•) and a small tabu list (⬥) is recommend to avoid recession in the searching of algorithm Tabu for TSP problem instances. In [63], the metrics autocorrelation length and fitness distance correlation characterized the searching behaviour (∇) to encodings of bit-flip mutation, crossover (one point, uniform) and different heuristics (□) for genetic algorithms applied to five solution representations for Multidimensional Knapsack problem; it is observed that adding heuristics and local improvement to binary representation, so too use a log-normal mutation operator to weight-coding representation are the best logical internal structures for this problem. Other works, considering information of problem and algorithm; specifically problem structure description (*) for One-Dimension Bin-Packing [89], indexes of central tendency and variability of the weights of items, and indexes about the genetic algorithm operative behaviour (Δ), average deviation in the fitness of the solutions of the population and deviation in the fitness of the best solutions of each generation, indicate by means data exploratory and causal analysis, the algorithm had good results for instances with large items but do not appear to be adequate to address instances with medium and small items. Explanations permitted to redesign the algorithm, incorporating efficient random heuristics to diversify the algorithm behaviour and prevent a premature converge. For the same problem, in the works [70, 84–86] was included more information about problem and algorithm for Threshold Accepting (AT) and Tabu Search (TS). Knowledge was found by causal analysis; it is to say, for a better performance of the algorithms, if problem instance structure description indexes (*) (statistics about weights dispersion, frequencies, relation weights-container capacity), problem space description indexes (+) and operative and searching behaviour indexes (Δ,∇) indicate a facility for generating a diversity of solutions, then it is recommend to manage a Tabu list size bigger than 7 (⬥) (TS), a smaller temperature (⬥) (AT), consider several methods for generating neighbour solutions (□) and intensify the searching in a big space. In case of the opposite of the above, it is recommended to manage a Tabu list size smaller than 7 (⬥) (TS), a bigger temperature (⬥) and one method for generating neighbour solutions (□). There is no necessity to work much when the space is small. The algorithms were redesigned with these explanations and their performance improved to original algorithms. The method for generating the initial solution (•) was not significant to identify the dominations partition for both algorithms (TS) and (TA). This knowledge was similar for the problem TSP and Tabu Search algorithm [69]. Finally, this reviewing indicates there is no work that analyses all kind of information; problem and algorithm indexes in conjunction with the algorithm logic structure. In the majority of cases, it is seen as a black-box. However, it is important to open the box and identify logical parts relevant to the connection of the algorithm and problem during the solution process [27]. There is still hidden knowledge significant to understand the phenomenon and give an answer to questioning in the explanatory understanding level.

4 Setting the Problem Statement

It is necessary to break ground with a starting point, something very essential, simple, and important in the way for the phenomenon comprehension in the explanatory understanding level. A formal formulation that considers all significant information from problem (structure, solutions space), algorithm (operative and searching behaviour) and algorithm logical internal structure; beyond to algorithm selection problem ASP. There is no work has made this before. The next problem statement is fixed.

> For a set of algorithms A, with logical internal structure each one, applied to a set of problem instances $\boldsymbol{P}$, with problem indexes $\boldsymbol{F}$, algorithm indexes $\boldsymbol{L}$, the algorithms performance space $\boldsymbol{Y}$, the set of algorithms domination partitions $\boldsymbol{W}$, according to $\boldsymbol{Y}$, and an ordered pair $(a_q, C_q) \in \boldsymbol{W}$; find the Explanation Mapping $\boldsymbol{E}(a_q, \boldsymbol{P}, \boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L}, \boldsymbol{Y}, \boldsymbol{W})$ that discovers an explanation formal model $\boldsymbol{M}$, such that $\boldsymbol{M}$, represents the latent knowledge from relations between problem and algorithm indexes and provides solid foundations to explain, why certain problem instances, being the partition C_q correspond better to algorithm a_q with certain logical internal structure, according to performance space $\boldsymbol{Y}$, and why other partitions $(C_q)^c$ do not correspond to algorithm a_q with certain logical internal structure.

5 A Framework General Proposal and Guidelines

Figure 2 illustrates a general framework proposed for starting in an adequate way and giving a solution to above problem, firstly for a specific domain through next phases and after of a repetitive process, build the theoretical bases towards generalization. It is based from the recommendations and contributions by reviewed literature, as well as, from previous works results [84–86]. The validation is recommended to be performed in Phase V and Phase VII, but it can be left to the researcher's choice in conjunction with other specific details. So too, it is recommended that the results (successful or failed) of these phases and all of its concerning information must be registered [27] for identifying the explored ways and continuing with the unexplored.

Phase I. Identification of problem and algorithms. A kind of problem and algorithms of interest are fixed as study objects. It can be that one algorithm only be changed, keeping everything else fixed. It will depend of the explored and unexplored ways.
Phase II. Designing of experiments. The principal goal of the designing of experiments must be revealing [27]. It must offer, in general terms, insights of why one algorithm is the best to solve a problem instances subset and why not in other; more formally to solve the problem formal statement described in the last section. For it, is necessary to obtain a sample of instances representative to problem. It is recommended obtain real instances and to generate random instances representative of problem space [25, 27]. The set $\boldsymbol{P}$ is defined. So too, is necessary analyze all factors involved in the solution process between problem-algorithm [27] (the environmental and programming factors would be controlled). It was observed in the Sect. 3 that it is important to include information about structure (*) and space (+) of problem, operative (Δ) and searching (∇) behavior of algorithm [27]; mathematical or statistical indexes of above and performance metrics

are defined. So too, the algorithm internal logical structure is important to analyze [27, 82]. According to all above and gain insight into the effect of specific factors on an algorithm, it is recommended test two codes that are identical except for the singled-out one part of their internal logical structure within a well-designed experiment (it must be registered); such as a factorial design of experiments that permits an analysis scenario of the smallest detail in algorithm logical in conjunction with the other problem and algorithm factors [27]. The set $\boldsymbol{A}$ and one algorithm of interest a_q are fixed.

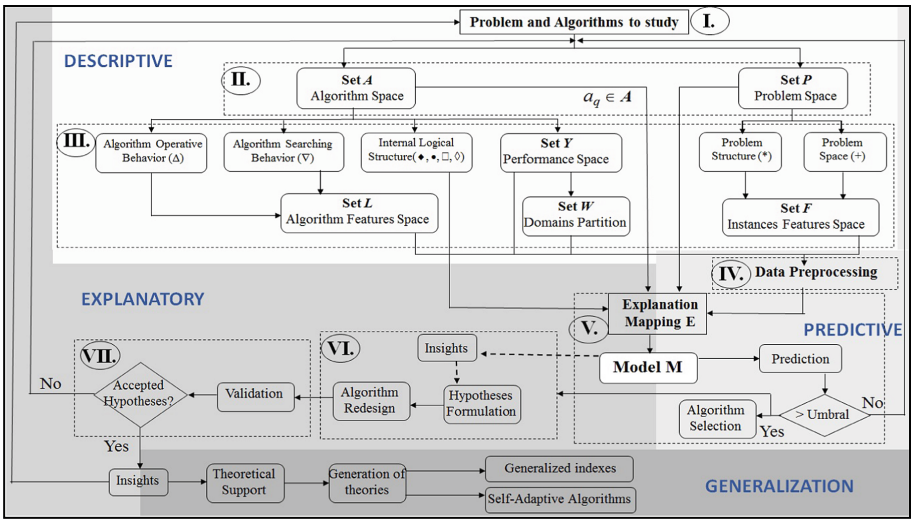

Fig. 2. Proposed General Framework (a connection between understanding levels towards an empirical generalization)

Phase III. Characterization of problem and algorithms. A process of description, by means indexes [84–86, 91] is applied before execution to set $\boldsymbol{P}$ for characterizing the structure (*) and space (+) of the problem; a set $\boldsymbol{F}$ is obtained. During the execution, mathematical or statistical indexes are applied to algorithms for characterizing the operative (Δ) and searching behavior (∇), as well as, the performance metrics are applied to measure the performance. The sets $\boldsymbol{F}$, $\boldsymbol{L}$, $\boldsymbol{Y}$, $\boldsymbol{W}$ are obtained.

Phase IV. Preprocessing of data. It is recommended to perform a step of data preprocessing (elimination of missing values, balancing datasets, discretization) that could improve the quality of the data [30, 90, 92]. Also is recommended the elimination of redundant data and identification of significant features (selection) [71].

Phase V. Explanation Mapping. Perform a mechanism that permits an explanation mapping. It is recommended an analysis approach that generate a **M** model that represents the latent knowledge of relevant relations from preprocessed dataset. Also, this model must be very clear and easy to interpret how the problem and algorithm indexes are relating; the explanations obtained would help to researchers in this area to understand the performance and behaviour of algorithm in its operation and searching process, and

identify internal logical design aspects that are relating with the problem structure and solutions space; how these relations influence to find the best solution. So too, the model must permit to predict the algorithm performance. It is important a validation of prediction process (this is not included in figure to avoid confusion) with test instances. If the results are superior to a stablished umbral then it will continue with the Phase VI (it is good to predict, it is expected to be good to explain); otherwise return to Phase I, modifying samples, sets, design of experiments or indexes and continue again with Phase II and so on. The promising analysis approaches are Machine Learning (Decision Tree, Bayesian Net), Markov Model, Causal Analysis; all these conjugating with a visual analysis.

Phase VI. Formulation of Hypotheses and Algorithm Redesign. The explanations from model **M** are insights to formulate hypothesis and redesign the algorithms.

Phase VII. Validation of Hypothesis and Feed-Back. The redesigned algorithms are run over the set $\boldsymbol{P}$, as well as it is recommended over another set $\boldsymbol{P'}$ ($\boldsymbol{P} \neq \boldsymbol{P'}$) representative of problem domain; if there is a significant improving of the redesign over those sets ($\boldsymbol{P}$ and $\boldsymbol{P'}$), then the hypotheses will be accepted, considered as found knowledge and return to Phase I; modifying samples, sets or indexes for continuing again with Phase II and so on. As well as, depending of the obtained advances, it can be chosen to apply another problem domain and other algorithms.

Finally, theoretical support is obtained from a refining process orderly and repetitive of Phase I to Phase VII through other problem domains and algorithms. Theories can be found for generalizing indexes about problem, algorithm and performance metrics; as well as, permitting the building of Self-Adaptive algorithms. It is to say, the automatic redesign of algorithms, during execution time; adapt the algorithm logical structure, operational and search behavior of algorithm to: structure and solutions space of problem, and to difficulties presented during execution to give the best optimal solution to real problems in a broad range of disciplines.

References

1. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization, Algorithms and Complexity. Prentice Hall(1982)
2. Wolpert, D., Macready, W.: No free lunch theorems for optimizations. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1996)
3. Lagoudakis, M., Littman, M.: Learning to select branching rules in the dpll procedure for satisfiability. Electron. Notes Discr. Math. **9**, 344–359 (2001)
4. Vanchipura, R., Sridharan, R.: Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence-dependent setup times. Int. J. Adv. Manuf. Technol. **67**, 1337–1353 (2013)
5. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: Survey and perspectives. Evol. Comput. **27**(1), 3–45 (2019)
6. Hutter, F., Xu, L., Hoos, H., Leyton-Brown, K.: Algorithm runtime prediction: Methods & evaluation. Artif. Intell. **206**, 79–111 (2014)
7. Xu, L., Hoos, H., Leyton-Brown, K.: Hydra: Automatically configuring algorithms for portfolio-based selection. In: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI'10), pp. 210–216 (2010)

8. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
9. Cayci, A., Menasalvas, E., Saygin, Y., Eibe, S.: Self-configuring data mining for ubiquitous computing. Inf. Sci. **246**, 83–99 (2013)
10. Yeguas, E., Luzón, M., Pavón, R., Laza, R., Arroyo, G., Díaz, F.: Automatic parameter tuning for evolutionary algorithms using a bayesian case-based reasoning system. Appl. Soft Comput. **18**, 185–195 (2014)
11. 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. **66**(5), 782–793 (2015)
12. Schede, E., et al.: A survey of methods for automated algorithm configuration. J. Artif. Intell. Res. **75**, 425–487 (2022)
13. Leyton-Brown, K., Hoos, H., Hutter, F., Xu, L.: Understanding the empirical hardness of NP-complete problems. Magaz. Commun. ACM **57**(5), 98–107 (2014)
14. Cruz, L., Gómez, C., Pérez, J., Landero, V., Quiroz, M., Ochoa, A.: Algorithm Selection: From Meta-Learning to Hyper-Heuristics, INTECH Open Access Publisher (2012)
15. Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K.: Recent advances in selection hyper-heuristics. Eur. J. Oper. Res. **285**(2), 405–428 (2020)
16. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 556–572. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46135-3_37
17. Yong, X., Feng, D., Rongchun, Z.: Optimal selection of image segmentation algorithms based on performance prediction. In: Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing, Australian Computer Society, Inc., pp. 105–108 (2003)
18. Guerri, A., Milano, M.: Learning techniques for automatic algorithm portfolio selection. In: Burke, V.A. (ed.) Proceedings of the 16th Biennial European Conference on Artificial Intelligence, pp. 475–479. IOS Press, Spain (2004)
19. Guo, H., Hsu, W.H.: A learning-based algorithm selection meta-reasoner for the real-time MPE problem. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 307–318. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30549-1_28
20. Perez O., J., et al.: A machine learning approach for modeling algorithm performance predictors. In: Torra, V., Narukawa, Y. (eds.) MDAI 2004. LNCS (LNAI), vol. 3131, pp. 70–80. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27774-3_8
21. Yuen, S., Zhang, X.: Multiobjective evolutionary algorithm portfolio: Choosing suitable algorithm for multiobjective optimization problem. In: 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, pp. 1967–1973 (2014)
22. Munoz, M., Kirley, M., Halgamuge, S.: Exploratory landscape analysis of continuous space optimization problems using information content. Evolution. Comput. IEEE Trans. **19**(1), 74–87 (2015)
23. Müller, D., Müller, M.G., Kress, D., Pesch, E.: An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. Eur. J. Oper. Res. **302**(3), 874–891 (2022)
24. Tsang, E., Borrett, J., Kwan, A.: An attempt to map the performance of a range of algorithm and heuristic combinations. In: Hallam, J. et al. (ed.) Hybrid Problems, Hybrid Solutions. Proceedings of the AISB-95, vol. 27, pp. 203–216. IOS Press, Amsterdam (1995)
25. Alsouly, H., Kirley, M., Muñoz, M.A.: An instance space analysis of constrained multi-objective optimization problems. IEEE Trans. Evolution. Comput. (2022)
26. Cohen, P.: Empirical Methods for Artificial Intelligence. The MIT Press Cambridge, London, England (1995)

27. Barr, R., Golden, B., Kelly, J., Resende, M., Stewart, W.: Designing and reporting on computational experiments with heuristic methods. J. Heurist. **1**(1), 9–32 (1995)
28. Reeves, C.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publishing, England (1993)
29. Frost, D., Rish, I., Vila, L.: Summarizing CSP hardness with continuous probability distributions. In: Proceedings of the 14th National Conference on AI, American Association for Artificial Intelligence, pp. 327–333 (1997)
30. Lee, J., Giraud, C.: Predicting algorithm accuracy with a small set of effective meta-features. In Machine Learning and Applications, In: Eleventh International Conference on IEEE, pp. 808–812 (2008)
31. Pérez O, J., Pazos R, R., Cruz R, L., Reyes S, G., Basave T, R., Fraire H, H.: Improving the efficiency and efficacy of the K-means clustering algorithm through a new convergence condition. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007. LNCS, vol. 4707, pp. 674–682. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74484-9_58
32. Fu, H., Xu, Y., Chen, S., Liu, J.: Improving WalkSAT for random 3-SAT problems. J. Univ. Comput. Sci. **26**(2), 220–243 (2020)
33. Pérez-Ortega, J., et al.: Hybrid fuzzy C-means clustering algorithm oriented to big data realms. Axioms **11**(8), 377 (2022)
34. Rice, J.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976)
35. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. **41**(1), 1–25 (2009)
36. Silverthorn, B., Miikkulainen, R.: Latent class models for algorithm portfolio methods. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Georgia (2010)
37. Xu, L., Hoos, H., Leyton-Brown, K.: Predicting Satisfiability at the Phase Transition. In AAAI (2012)
38. Burke, E., et al.: Hyper-heuristics: A survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
39. Li, B., Hoi, S.: Online portfolio selection: A survey. ACM Comp. Surveys (CSUR) **46**(3), 35 (2014)
40. Yuen, S.Y., Zhang, X.: On composing an algorithm portfolio. Memet. Comput. **7**(3), 203–214 (2015). https://doi.org/10.1007/s12293-015-0159-9
41. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. Eur. J. Oper. Res. **296**(2), 393–422 (2022)
42. Rice, J.: On the construction of poly-algorithms for automatic numerical analysis. In: Klerer, M., Reinfelds, J. (eds.) Interactive System for Experimental Applied Mathematics, pp. 301–313. Academic Press, MA (1968)
43. Beck, J.C., Freuder, E.C.: Simple rules for low-knowledge algorithm selection. In: Régin, J.-C., Rueher, M. (eds.) CPAIOR 2004. LNCS, vol. 3011, pp. 50–64. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24664-0_4
44. Knuth, D.: Estimating the efficiency of backtrack programs. Math. Comput. **29**(129), 122–136 (1975)
45. Purdom, P.: Tree size by partial backtracking. SIAM J. Comput. **7**(4), 481–491 (1978)
46. Sillito, J.: Improvements to and estimating the cost of backtracking algorithms for constraint satisfaction problems. M.Sc. thesis, Department of Computing Science, University of Alberta, Department of Computing Science, Edmonton, Alberta (2000)
47. Allen, J.A., Minton, S.: Selecting the right heuristic algorithm: Runtime performance predictors. In: McCalla, G. (ed.) AI 1996. LNCS, vol. 1081, pp. 41–53. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61291-2_40

48. Lobjois, L., Lemaître, M.: Branch and bound algorithm selection by performance prediction. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), Madison, Winsconsin, pp. 353–358 (1998)
49. Lagoudakis, M., Littman, M.: Algorithm selection using reinforcement learning. In: Kaufmann, M. (ed.), International Conference on Machine Learning (ICML 2000), pp. 511–518 (2000)
50. Huberman, B., Hogg, T.: Phase transitions in artificial intelligence systems. Artif. Intell. **33**(2), 155–171 (1987)
51. Lindauer, M., Hoos, H., Leyton-Brown, K., Schaub, T.: Automatic construction of parallel portfolios via algorithm configuration. Artif. Intell. **244**, 272–290 (2017)
52. Liu, S., Tang, K., Yao, X.: Generative adversarial construction of parallel portfolios. IEEE Trans. Cybernet. **52**(2), 784–795 (2020)
53. Kostovska, A., et al.: Per-run algorithm selection with warm-starting using trajectory-based features. In: Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, Proceedings, Part I, pp. 46–60. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_4
54. Soares, C., Pinto, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. J. Mach. Learn. **50**(3), 251–277 (2003)
55. Horvitz, E, Ruan, Y.: A Bayesian approach to tackling hard computational problems. In: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, pp. 235–244. Morgan Kaufmann Publishers Inc., San Francisco (2001)
56. Skvorc, U., Eftimov, T., Korošec, P.: Transfer learning analysis of multi-class classification for landscape-aware algorithm selection. Mathematics **10**(3), 432 (2022)
57. Kanda, J., De Carvalho, A., Hruschka, E., Soares, C., Brazdil, P.: Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features. Neurocomputing **205**, 393–406 (2016)
58. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 2010. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13800-3_29
59. Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K.C., Jin, Y.: A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. IEEE Trans. Artif. Intell. **1**(1), 5–18 (2020)
60. Trajanov, R., Dimeski, S., Popovski, M., Korošec, P., Eftimov, T.: Explainable landscape analysis in automated algorithm performance prediction. In: Jiménez Laredo, J.L., Hidalgo, J.I., Babaagba, K.O. (eds.) EvoApplications 2022. LNCS, vol. 13224, pp. 207–222. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-02462-7_14
61. Zhang, Y., Bai, R., Qu, R., Tu, C., Jin, J.: A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. Eur. J. Oper. Res. **300**(2), 418–427 (2022)
62. Hoos, H.H., Smyth, K., Stützle, T.: Search space features underlying the performance of stochastic local search algorithms for MAX-SAT. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 51–60. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_6
63. Tavares, J., Pereira, F., Costa, E.: Multidimensional knapsack problem: A fitness landscape analysis. IEEE Trans. Syst. Man Cybern. B Cybern. **38**(3), 604–616 (2008)
64. Le, M., Ong, Y., Jin, Y.: Lamarckian memetic algorithms: local optimum and connectivity structure analysis. Memet. Comput. **1**, 175–190 (2009)
65. Taghavi, T., Pimentel, A., Sabeghi, M.: VMODEX: A novel visualization tool for rapid analysis of heuristic-based multi-objective design space exploration of heterogeneous MPSoC arquitectures. Simul. Model. Pract. Theory **22**, 166–196 (2011)

66. Cruz, L., Gómez, C., Castillo, N., Quiroz, M., Ortíz, C., Hernández, P.: A visualization tool for heuristic algorithms analysis. In: Uden, L., Herrera, F., Bajo, J., Corchado, J. (eds.) 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing, Advances in Intelligent Systems and Computing, vol. 172, pp. 515–524. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30867-3_46
67. Lopez, T.T., Schaefer, E., Domiguez-Diaz, D., Dominguez-Carrillo, G.: Structural effects in algorithm performance: A framework and a case study on graph coloring. In: Computing Conference, 2017, pp. 101–112. IEEE (2017)
68. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
69. Watson, J., Darrell, W., Adele, E.: Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying Tabu search. J. Artif. Intell. Res. **24**, 221–261 (2005)
70. Pérez, J., Cruz, L., Pazos, R., Landero, V., Pérez, V.: Application of causal models for the selection and redesign of heuristic algorithms for solving the bin-packing problem. Polish J. Environ. Stud. **17**(4C, Hard) 25–30 (2008)
71. Pérez, J., et al.: A causal approach for explaining why a heuristic algorithm outperforms another in solving an instance set of the bin packing problem. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 591–598. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68123-6_64
72. Cheeseman, P., Kanefsky, B., Taylor, W.: Where the really hard problems are. In: The 12th IJCAI 91, pp. 331–337 (1991)
73. Thiebaux, S., Slaney, J., Kilby, P.: Estimating the hardness of optimization. In: ECAI, Berlin, pp. 123–130 (2000)
74. Spirtes, P., Glymour, C.: Causation, prediction, and search. MIT Press (2001)
75. Hoos, H.: A mixture-model for the behaviour of SLS algorithms for SAT. In: AAAI/IAAI, pp. 661–667 (2002)
76. Zhang, W.: Phase transitions and backbones of the asymmetric traveling salesman problem. J. Artif. Intell. Res. **21**, 471–497 (2004)
77. Watson, J.: An introduction to fitness landscape analysis and cost models for local search. In: Gendreau, M., Potvin, J. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146, pp. 599–623. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1665-5_20
78. Gao, W., Nallaperuma, S., Neumann, F.: Feature-based diversity optimization for problem instance classification. Evol. Comput. **29**(1), 107–128 (2021)
79. Hogg, T., Huberman, B., Williams, C.: Phase transitions and the search problem. Artif. Intell. **81**(1), 1–15 (1996)
80. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. Proc. Conf. Artif. Intell. **92**, 459–465 (1992)
81. Smith-Miles, K.A., James, R.J.W., Giffin, J.W., Tu, Y.: A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance. In: Stützle, T. (ed.) LION 2009. LNCS, vol. 5851, pp. 89–103. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-11169-3_7
82. Liefooghe, A., Daolio, F., Verel, S., Derbel, B., Aguirre, H., Tanaka, K.: Landscape-aware performance prediction for evolutionary multiobjective optimization. IEEE Trans. Evol. Comput. **24**(6), 1063–1077 (2019)
83. Korb, K.: Bayesian Artificial Intelligence. Chapman and Hall, London (2004)

84. Landero, V., Pérez, J., Cruz, L., Turrubiates, T., Ríos, D.: Effects in the algorithm performance from problem structure, searching behavior and temperature: A causal study case for threshold accepting and bin-packing. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11619, pp. 152–166. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24289-3_13
85. Landero, V., Ríos, D., Pérez, J., Cruz, L., Collazos-Morales, C.: Characterizing and analyzing the relation between bin-packing problem and Tabu search algorithm. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12249, pp. 149–164. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58799-4_11
86. Landero, V., Ríos, D., Pérez, O.J., Collazos-Morales, C.A.: A composite function for understanding bin-packing problem and Tabu search: Towards self-adaptive algorithms. In: Gervasi, O., et al. (eds.) ICCSA 2021. LNCS, vol. 12949, pp. 592–608. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86653-2_43
87. Gent, I., Macintyre, E., Prosser, P., Walsh, T.: The scaling of search cost. In: AAAI'97, pp. 315–320. Mit Press, Rhode Island (1997)
88. Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Romero, D., Cruz, L.: A statistical approach for algorithm selection. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 417–431. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24838-5_31
89. Quiroz, M., Cruz, L., Torrez, J., Gómez, C.: Improving the performance of heuristic algorithms based on exploratory data analysis. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Intelligent Systems, Studies in Computational Intelligence, vol. 452, pp. 361–375. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33021-6_29
90. Cruz R., L., Pérez, J., Landero N., V., del Angel, E.S., Álvarez, V.M., Peréz, V.: An ordered preprocessing scheme for data mining. In: Zhang, C., W. Guesgen, H., Yeap, W.-K. (eds.) PRICAI 2004. LNCS (LNAI), vol. 3157, pp. 1007–1008. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28633-2_137
91. Gómez, S., et al.: Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system. J. Comput. Appl. Math. **259**, 965–977 (2014)
92. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. **32**, 565–606 (2008)