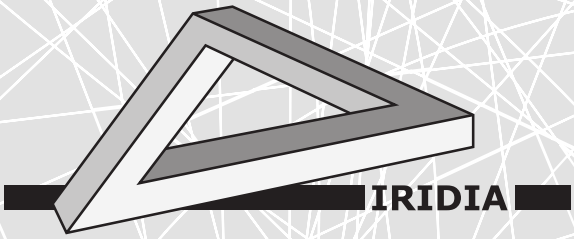

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

A causal framework for optimization algorithms

A. Franzin and T. Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2022-007

August 2022

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2022-007

A causal framework for optimization algorithms

Alberto Franzin[a], Thomas Stützle[a]

[a] *Université Libre de Bruxelles (ULB), Belgium*

Abstract

Despite the many optimization algorithms available in the literature and the several different approaches that study them, understanding the behaviour of an optimization algorithm and explaning its results are fundamental open questions in Operations Research and Artificial Intelligence. We argue that the body of literature available is already very rich, and the main obstacle to the advancements towards an answer to those questions is its fragmentation. In this work we propose a causal framework that relates the entities involved in the solution of an optimization problem. We show how this conceptual framework can be used to relate many approaches aimed at understanding how algorithms work, and how it can be used to address open problems, both theoretical and practical. In particular, we use it to show how to make use of data collected in past experiments to automatically infer plausible algorithms for unseen problems, and provide a proof-of-concept using a simple stochastic local search.

1. Introduction

In the last decades, plenty of exact and non-exact methods have been proposed to tackle optimization problems. However, if algorithms for polynomially solvable problems are supported by mathematical proofs that both guarantee and explain the results, it is far more challenging to understand how effectively algorithms for NP-hard problems obtain their results. The branch-and-bound algorithm, which is the base of many state-of-the-art methods in exact optimization, exhibits a chaotic internal behaviour [86, 49]. Hence it is almost impossible to know a priori the performance of a solver on an instance in practice. Theoretical analyses of SLS behaviour have often been limited to specific cases, idealized models, or unrealistic assumptions such as infinite running time. In particular, despite the hundreds of metaheuristic algorithms and the thousands of variants proposed in the literature, understanding how a stochastic local search (SLS) algorithm works in general is still an open research question [128, 28]. Yet, these algorithms are routinely used with successes in countless applications. Thus, the gap between theoretical analyses and practical results is still very large.

In recent years, however, the adoption of rigorous experimental practices led many researchers to develop mathematical and statistical tools to gain insights

Email addresses: alberto.franzin@ulb.be (Alberto Franzin), thomas.stuetzle@ulb.be (Thomas Stützle)

in algorithmic behaviour [64]. For example, various analyses have tried to determine bounds on either solution quality or runtime of heuristic algorithms in various scenarios, or the optimal parameter values for an algorithm on a certain instance [38, 31, 37]. Alongside the theoretical approaches, the increased availability of computational power made empirical analyses possible. After early examples [73, 74], the increased availability of cheap computational power makes it possible to perform extensive comparison between algorithms and study the impact of particular problem features thanks to careful experimental design and statistical analyses [8]. Efficient algorithms can be instantiated with reduced human efforts thanks to automatic selection and configuration tools [116, 81, 87, 63]. Selection and configuration tasks generate also huge amounts of data, that can be analyzed to gain insights on both the algorithms and the scenarios considered [69, 70, 30]. Fitness landscape analyses study properties of problems and instances, and can be used to explain algorithmic results [115, 16].

In the literature, however, little work has been done to formally connect these related areas of research. The main issue lies in the difficulty of generalizing the many results obtained for specific problems, instances and algorithms to other scenarios. In turn, this difficulty arises from the huge number of options we can choose from, on the stochasticity of the algorithms and instances, and on the computational burden of generating usable information. To understand the behaviour of an algorithm, a practitioner still needs manually inspect the results obtained, and relate them with the information available on the problem and the instance. Moreover, the knowledge acquired both with practice and from the relevant literature is often difficult to apply to the development of a new algorithm or a new application.

In this work, we show how the works in literature coming from these research areas already contain a huge amount of knowledge and methods to understand how an optimization algorithm works, and to explain its results. We start from simple assumptions about optimization algorithms to develop an unified framework that relates the elements involved in the solution of an optimization problem. We use a causality-based approach, to make use of well-established properties of causal models to show (i) how this general framework represents the relationships between the high-level entities at play when solving a problem, (ii) how several existing approaches considered in various areas of research related to the understanding of optimization algorithms from an empirical perspective, and (iii) how it can be used in practice to bridge the knowledge gap between different problems and algorithms. We argue that the various approaches proposed to understand algorithmic behaviour are inference tasks on our causal framework, essentially offering different perspectives on the same subject.

In particular, using our conceptual framework and the properties of causal models, we show how to make use of available methods and past experiments to perform transfer learning of algorithms across different optimization problems. We focus on trajectory-based SLS algorithms for combinatorial optimization problems; the framework is however sufficiently general to be also applicable to other heuristic and exact algorithms or continuous problems.

In Section 2 we outline our working hypotheses and present the causal framework. In Section 3, we review several works approaches aimed at understanding the behaviour of optimization algorithms and explaining their results, and relate

them to our causal framework. We then show two applications made possible by the framework, one more theoretical and one more practical. First, in Section 4 we relate the tasks of Algorithm Selection and Algorithm Configuration. Then, in Section 5 we show how our framework is useful to guide the development of a practical application, namely the transfer of algorithm configurations across problems. Finally, we conclude in Section 6 outlining several possible research directions.

2. A causal framework for SLS algorithms

In this section we define a theory that encodes the hypotheses, and a subsequent causal framework to model the relationships between the entities involved in the solution of an optimization problem.[1] We define high-level entities, such as "algorithm" or "problem", and connect them according to their causal relationships. Later we will see how to instantiate the framework into an effective causal model for a given scenario, or set of scenarios, and the practical limitations we encounter in this instantiation.

2.1. Background

A causal model is a model that expresses the causality relationships between entities in a system [109]. We can use it to formalize a *theory* we have about the system under study in the form of a graph. Intuitively, each entity considered is a node, and two nodes are directly connected if the value taken by one of the two entities directly affects the value taken by the other one. When changing the value of the first variable, we then expect the value of the second variable to change accordingly. One common assumption is that this relationship is not bidirectional, and therefore only directed edges are used.

A causal model can be formally defined as a quadruple $\langle \mathcal{U}, \mathcal{V}, \mathcal{F}, \mathcal{P} \rangle$, where $\mathcal{U}$ is the set of *exogenous* variables whose values are determined by reasons external to the model (e.g., values we choose), $\mathcal{V}$ is the set of *endogenous* variables whose values are determined by a subset of the other variables in the model, $\mathcal{F}$ is a set of functions that we can use to compute the values of the $\mathcal{V}$ variables from the values of their parent variables, and $\mathcal{P}$ specifies a joint probability distribution over $\mathcal{U}$. Each variable can therefore be considered to model our subjective belief or knowledge about it.

The causal structure of the system can therefore be represented as a Directed Acyclic Graph (DAG) whose nodes are $\{\mathcal{U} \bigcup \mathcal{V}\}$ and the $\mathcal{V}$ nodes have incoming arcs from their parent variables. The values each node can take is defined probabilistically, for nodes in $\mathcal{U}$ by $\mathcal{P}$, and for nodes in $\mathcal{V}$ as probabilities conditional on their parent nodes, as defined using $\mathcal{F}$.

Causal models are particularly useful because they define how to reason about certain variables in a model, taking into account the other variables. There is in fact a codified set of rules that define how to update the value of an otherwise unobserved variable having observed a set of other variables, depending

[1] The framework differs slightly from a previous preliminary version [53], as a result of further analyses and discussions.

on how the variables involved are connected. This set of rules answers, case by case, the general question "how does new information about observed variables change our knowledge about the unobserved variables?". Models of this kind are particularly important because they are *falsifiable*: if the observations do not match the model, we can conclude the model (and with it, our theory about the system we represented in it) is wrong and it has to be fixed or discarded. They also enable the *integration* of observations from partially differing sources, thus being particularly useful in transferring knowledge across scenarios. Causal models have been increasingly adopted in disciplines such as medicine, biology, psychology, economy, in all cases when the goal is to understand causes and effects in a system.

Queries on causal models belong to three categories: inference, intervention, and counterfactual analysis. *Inference* is the task of observing the values a subset of variables $\mathbf{X}$ take when other variables $\mathbf{Y}$ take certain values. When performing *intervention* we instead actively modify the values of some variables $\mathbf{Y}$, regardless of whether there are existing causes for them. This is done using the $\texttt{do}(\cdot)$ operator, which performs a removal of the incoming nodes to $\mathbf{Y}$. Finally, *counterfactual analysis* aims to infer the values that some variables $\mathbf{X}$ would have taken, had other variables $\mathbf{Y}$ taken certain values. This level of analysis estimates the outcome of experiments that are not performed, and is particularly useful to provide explanations. These three actions are also referred to as the three rungs of the causality ladder, because each one is an abstraction level more powerful than the previous one: seeing the data, act on the system to obtain new data, imagine how some observations we don't have would be [111].

2.2. Working hypotheses

In this work, our goal is to propose a theory that relates the entities involved in the solution of an optimization problem. In building the theory and thus the framework, we start from the following four working hypotheses.

(H1) An algorithm can be divided into basic components and parameters. We take a component-based view of SLS algorithms [131, 89, 21], that is, we consider an algorithm as a set of basic building blocks, each one possibly coming with a set of parameters, combined together in a certain way.

(H2) Separation between algorithm-specific and problem-specific components. Following (H1), algorithmic blocks are divided in problem-specific and algorithm-specific. Problem-specific components are the parts of a SLS that require specific knowledge of the problem under study, such as the generation of an initial solution, the perturbation of a solution, or the evaluation of a solution. Algorithm-specific components are all the components that define what a SLS is. They can be used across different unrelated problems, such as the tabu list in a Tabu Search, or the cooling scheme in a Simulated Annealing[2].

[2]We include in the algorithm-specific components also those components that require problem-specific knowledge, but still operate at the algorithm level, such as the initial temperature defined for a simulated annealing for the PFSP proposed in [103].

(H3) An algorithm operates a traversal of the search space. An SLS works by traversing solutions in a search space; the problem-specific components of a SLS are needed to (i) select one starting point of this traversal, and (ii) evaluate another solution in the search space, relative to the current one.

(H4) Identification of optimal algorithm behaviour. For obtaining optimal results on a search space, an algorithm needs to reach an optimal tradeoff and alternance of diversification and intensification behaviour.

For any practical application, we can assume that our set of blocks is wide enough to cover all the needs for our case, that is, we can instantiate at least one fully working algorithm for any given problem. This is a reasonable assumption, since in case we miss the components we need to solve a certain problem, we can develop them and add them to our existing set of blocks.

Thus, an SLS is defined by its algorithmic-specific components and their combination, and an instantiation of a SLS for a problem is the combination of the algorithm-specific components with a set of problem-specific components. This separation also means that, if we have the problem-specific components for a problem of interest, we can use them to instantiate several algorithms for that problem, that will differ at the algorithm-specific level. In other words, with (H1) and (H2) we can apply the Programming by Optimization paradigm. We will also use interchangeably the terms "parameters" and "set of components," since an algorithm will be instantiated by choosing a set of blocks that, together, form a valid procedure for the task. Likewise, in this work the term "algorithm" can be considered synonym with "configuration."

Hypothesis (H3) formalizes the intuition that if we assume to have complete knowledge of the entire search space (or to have an oracle that gives us the desired information about a solution when polled), then any optimization problem becomes a search problem, and we do not need additional knowledge on the problem anymore. This is of course a far-fetched assumption when solving a problem instance in practice, but we show that, for the purpose of understanding algorithm behaviour, this lets us bridge the insights we obtain across different problems. The separation of the components of (H2) is an effective simulation of the oracle, because the search part of a SLS is devoted to traverse the search space, using the problem-specific components to retrieve information about the value of each solution considered.

Two different landscapes define, in principle, a different optimal behaviour, but several different algorithm can potentially reach this desired behaviour [52]. Building upon (H1) and (H2), with (H4) we fully realize the PbO paradigm, by assuming we can identify a subset of sufficiently powerful components that can reach this behaviour, regardless of the form (and name) of the resulting algorithm. Often, in practice, we do not have this subset, or we are not able to identify it among the components we have. So we can relax this hypothesis by assuming we can identify and configure an algorithm that obtains results that are "good enough" for our purpose.

2.3. The causal framework

The theory. From the working hypotheses (H1)–(H4), we consider problems and instances as given inputs to tackle, using an algorithm built starting from the

basic components; the efficiency of the algorithm depends on the computational environment such as machine power or running time. These factors are the *causes* for the results obtained, which therefore are the *effects*. What relates the inputs to the results, is how (efficiently) the algorithm can traverse the search space, that is, how it can efficiently oscillate between diversification and intensification, based on the characteristics of the instance. These characteristics can be represented by the features.

This theory is rather straightforward and many if not all the works cited in this paper implicitely already assume this model. However, by making it explicit, we construct a falsifiable theory that we can follow in our goal of explaining how and why an algorithm works.

The general causal framework representing the theory is shown in Figure 1. In the general framework each node is actually a macro-node representing a set of nodes, which are grouped together by their function.

High-level entities. P represents the variables relative to the problems, such as the objective function. D is the data necessary to instantiate one specific occurrence of a problem, e.g., the instance file. While it may be argued that an instance exists for a given problem (e.g. a TSP instance), we consider them separately to emphasize the separation between the generic problem definition and its specific realizations. Following (H2), PA is the set of problem-specific components and algorithms (e.g. initial solutions, neighbourhoods, heuristics), and SA is the set of algorithm-specific components that compose the search part of the algorithm. Components in PA and SA include both algorithmic functions such as neighbourhoods or acceptance criteria, but also numerical parameters.

C represents the set of computational factors that impact an actual implementation of any algorithm instantiated, such as the computational power, running time, random seed, but also quality of the implementation, and any other possible factor affecting the computation.

A is the set of algorithms that we can use for solving an instance $i_p \in D$ of problem $p \in P$; differently from PA and SA, an algorithm $a \in A$ is an instantiated and fully configured algorithm. The separation of A from its components collected in PA and SA follows from (H1).

L is the abstraction of the landscape that is generated when a problem in P is instantiated with data from D. For any practical purposes, we can consider the landscape as not directly observed, but we can observe some characteristics that represent it. We classify these features in FS (static), features that arise from the instantiation of a problem, and in FA, features that represent the landscape as seen by an algorithm in A. For example, FS contains the set of instance features, such as the instance size, the problem-specific features, such as angle features for the Travelling Salesman Problem, or properties of the solution landscape such as the ruggedness. FA instead contains features observed with some search algorithm, such as the number of iterations a Hill Climbing takes to converge to a local optimum.

Finally, R models the results that can be obtained by an algorithm $a \in A$ on an instance i_p for a problem $p \in P$, under the computational environment specified in C.

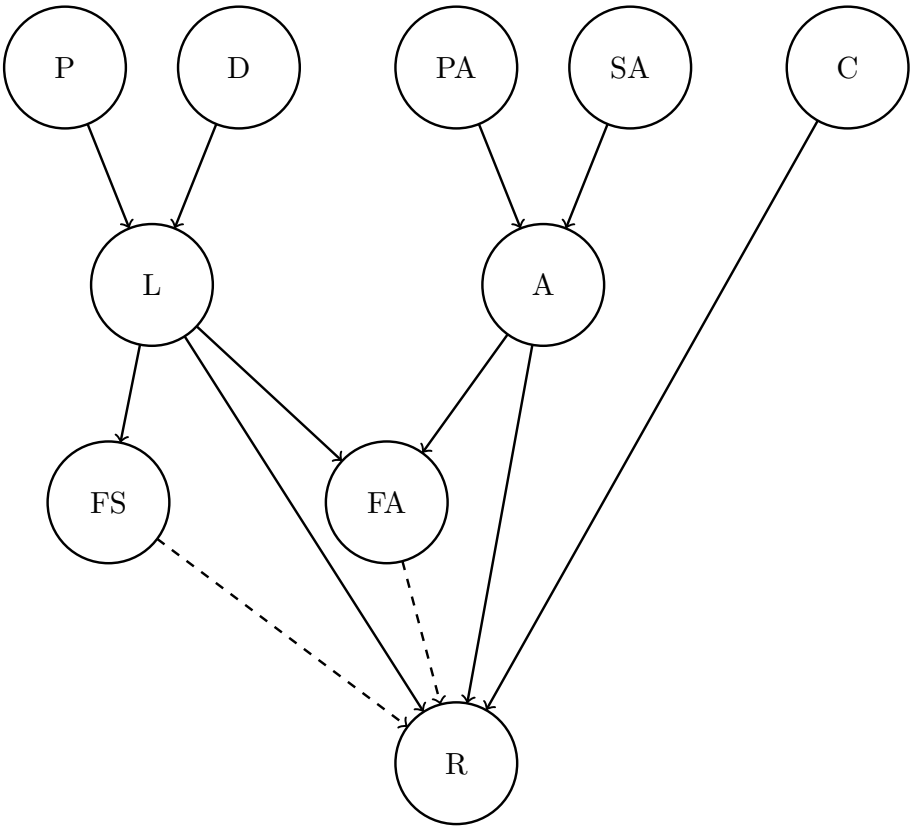

Figure 1: A generic causal framework representing the interaction between problems, instances, algorithms and results. Each node in the DAG is a macro-node modeling a set of entities: P for the problem, D for the data, PA and SA for respectively the problem-specific and algorithm-specific components, A for the search algorithm, C for the computational environment, L for the solution landscape, FS for the features of the instance, FA for the features that represent the landscape as seen by a search algorithm, and R for the results. Arcs represent the causal relationships between the high-level nodes. In practical applications, not all the nodes might be observed, and not all the nodes might be present, depending on the specific instantiation of the problem under study.

Causal relationships. $\{I, P, \text{PA}, \text{SA}, C\}$ form the set of exogenous variables, because they are the ones we set when we tackle an optimization problem; the endogenous variables are $\{L, A, \text{FS}, \text{FA}, R\}$, and their value depends on other variables, either deterministically or in a probabilistic way.

An algorithm in A can be instantiated combining components from PA and SA, according to (H1) and (H2), hence we have arcs from PA and SA to A. The relationship between PA, SA and A is entirely deterministic: an algorithm is either composed by a certain set of components or not. We remark that the PA and SA components used to compose the algorithms in A are not necessarily the same ones used to compute the landscape features in L. When it is not possible to decompose an algorithm $a \in A$ into basic building blocks from PA and SA, because we do not have or recognize them, we treat it as an intervention operation: we assume it is composed by unobservable building blocks and we manually instantiate it as $\texttt{do}(A = a)$.

The landscape L is defined solely by the instantiation of a problem, so we have incoming arcs from P and D.

The features in FS arise only from the contribution of L, which mediates P and D. The features in FA are instead observed when an algorithm in A traverses the landscape L. Note, however, that the features generated using an algorithm in E are considered mere properties of the landscape, and not an evaluation of the algorithm, which is a different task. It is also important to remark that as FS and FA represent properties, at this stage we do not impose any limit (from C) on our ability of observing them; clearly this cannot always happen in practice and we might need to settle for some approximation.

Finally, the results R for an instantiation of a problem as defined in P and D depend on the performance of an algorithm in A, under the computational environment represented in C; however, following our working hypothesis (H3), once an algorithm from A has been instantiated, it operates on a search space here represented by L and can be considered, at that point, unaware of the specific problem that generated it.

Despite L already containing (at the conceptual level) all the information about the problem and the instance, we include additional edges from FS and FA to R (dashed in Figure 1). We include these edges because FS and FA provide an observable intermediate (mediator) level of "reasons" that can help us understand the performance R of A for P, D given C, a level that we can use, conversely to the (usually unobservable) landscape L.

For the moment we do not assume any inter-node connection, that is, the nodes composing the high-level entities are not directly connected to each other, since they are defined either as exogenous variables, or as determined by other entities. This simplification may be not fully true, for example for nodes in PA and SA and their connections with A, where components can be combined in some hierarchical fashion [104], or the relationships between algorithm parameters can be represented using another DAG [15, 14], but for the sake of simplicity we choose to not investigate further this issue and to demand it to future work.

From this generic framework we can instantiate an effective causal model, depending on the specific case. For example, in the PbO paradigm, A will be the set of components in which we can decompose an algorithm. In the context of algorithm selection, instead, A is a variable taking values $a_1, a_2, \ldots, a_n$ for the n algorithms in the portfolio. Likewise, P contains at least the objective function, but if necessary it is possible to represent the constraints.

One of the advantages of the general framework is to abstract the connections between high level concepts, being therefore valid across several scenarios (in particular, problems). An instantiation of the framework for a specific problem may contain elements that do not apply to other problems (e.g. the number of clauses in SAT). It is anyway possible to select a tradeoff between generality and specificity and operate at the desired level of precision. Another consequence of this is that we do not need to worry about being too precise in the classification of the components and features, since their relationship with the other entities of the framework is far more important.

Notation and reasoning on the framework. The notation we use for this framework is somewhat an abuse of the notation used in causal models, and we use it to model the high-level interactions. For two high-level nodes X and Y, the $Y|X$ indicates the choice of Y depending on X. For example, we have that $A|\text{PA}, \text{SA}$, because the algorithms in A are generated starting from a specific instantiation of the components and parameters in PA and SA.

Causal and inferential reasoning is as in any causal model. Since there is no edge inside each macro-node, all the flows of information we define on the framework apply also when we "open" the macro-nodes into their constituting nodes.

Limitations of the framework. There are several difficulties we encounter when constructing and instantiating a specific model from the high-level framework. The first one lies in the amount of choices we need to make to properly define the model variables. How to represent the results? Do we consider absolute solution values, or relative deviations from optimal or best known solutions for the instances considered? In the latter case we can use best known values as bounds, but how do we choose to compute them? How to represent the problem or the constraints, when present? Which features to include, and how to represent them numerically? Many features will be correlated, so some feature engineering can be useful: how and to which extent can we do it? How to represent in a meaningful way the computational environment in C? Some of these questions have obvious answers when implementing an algorithm to perform experiments and collect data, but require reflection when integrating the relative information in a single model instantiation.

The next issue is related to the computational task defined by a model. A full model will contain several variables, and in particular for nodes in A and R, with many parent nodes the resulting conditional probability tables will be huge. One common assumption in many practical causal models e.g. in biology is that continuous variables appear only as terminal nodes, and causes and intermediate nodes are discrete. In this context, this assumption does not hold, because our framework can have continuous causes, for example continuous parameters.

The third issue lies in the difficulty of fully instantiating a model; aside from the entities that may not be considered in some of the tasks, single tasks or studies usually ignore or underestimate some of the variables involved. For example, algorithm comparisons often restrict the analysis or the experiments to a fixed set of design and parameter choices. This makes it impossible to generalize the findings, since, for example, a proper configuration can improve the algorithms and subvert the results [52].

Nonetheless, in the next section we show two applications of this framework. In the first one we demonstrate how under our framework the algorithm selection and configuration problems can be conceptually related to each other, and the difference between them is not in the characteristics of the tasks but in their practical application. We then show how to exploit the relationship between the various entities involved to make use of data collections to automatically find configuration that work on new, unseen problems.

3. Some examples from the literature

In this section, we revise research lines aimed at understanding the behaviour of optimization algorithms, and relate them to our framework, showing how they can be represented as inference tasks on the framework. The goal of this section is to show how the framework can be used to define the research questions, highlighting the different perspectives and elements considered in each work. While a complete literature review is beyond the scope of this work, we choose to select some papers to cover a wide range of methods and applications.

3.1. Theoretical and experimental analysis of algorithms

The analysis of the results obtained by optimization algorithms is the most common kind of analysis for SLS algorithms and it appears in the literature in countless different flavours. Theoretical works aim to establish statements that are certain and true given a certain set of assumptions. However, concessions usually have to be made in the assumptions to be able to derive general statements. Nonetheless, several sophisticated mathematical tools have been developed to analyze the behaviour of an algorithm, from schema theory for genetic algorithms to methods based on Markov chains to statistical analyses [59, 47, 39, 6]. Experimental analyses are instead the only viable choice when moving from ideal assumptions to problems meant to model real-life applications or from basic algorithmic templates to complex ones. Over the years, the research community has developed best practices and guidelines about performing computational experiments [61, 62, 8, 72, 25]. Theory and experiments offer therefore complementary views on the same research questions. Under our framework, we can focus on the specific question asked in each work, and consider theory and experiments as tools to provide answers.

Historically, the theoretical analysis of SLS has focused on convergence results, to show how a particular algorithm could guarantee the discovery of an optimal solution under a set of assumptions. This means to prove that, for an algorithm $a \in A$,

$$R = r^* | A = a, P = p, D = d, C = c, \tag{1}$$

where r^* is one optimal solution, for some assumptions encoded in p, d, c. Notable examples are the convergence analyses of cooling schemes in the simulated annealing literature [95, 117, 134], where it is shown how some cooling schemes are guaranteed to find an optimal solution under the assumption than infinite runtime is available to solve an instance sampled from an uniformly random distribution. Noting that, in this case, the cooling scheme is one of the components of simulated annealing we can use (H1) and (H2) to describe the goal of these works as proving[3]

$$\lim_{\text{runtime} \in c \to \infty} Prob(R = r^* | \text{SA} = sa, \text{PA} = pa, P = p, D = d, C = c) = 1, \quad (2)$$

that is, given infinite runtime, the probability that an algorithm composed with a selected set of pa, sa that include the specific cooling scheme considered finds the optimal solution r^* for an instance of p, d approaches 1. These analyses have since then developed also for other algorithms. For example, Gutjahr studied the convergence to the optimal solution of ant colony optimization algorithms [55, 56]. Rudolph discussed conditions that allow a non-elitist $(1, \lambda)$-ES to converge to the optimal solution in infinite time [119]. He also proved that a canonical genetic algorithm will never converge to the optimal solution for any combinatorial optimization problem, but its variant that keeps the best solution after selection can, in infinite time [118]. Reviews of this area can be found in [6, 12, 101, 102].

On the other hand, runtime analyses aim to determine (bounds on) the runtime that an algorithm $a \in A$ takes to find the optimal solution of an instance $d \in D$ of a certain problem $p \in P$. Hence, the problem can be framed as

$$\text{runtime} \in C | A = a, P = p, D = d. \quad (3)$$

Examples from the theoretical community include analyses of generalized hill climbing (another name for the fixed temperature variant of simulated annealing) [71], or ant colony optimization variants [129, 41].

Works such as [95, 57, 138, 90], instead, compare alternative cooling strategies, notably the traditional geometrical cooling scheme and the fixed temperature one. A statement such as "algorithm $a_1 \in A$ is superior to algorithm $a_2 \in A$" (or, alternatively, a component sa_1 is superior to another component sa_2) can be interpreted in several ways, even when considering one specific scenario. This can make it difficult to relate the findings of two different works, because of the different perspectives. Our causal framework can help disambiguate the precise question asked. For example, we could mean that a_1 converges faster than a_2 to an optimal solution, thus having a comparative runtime analysis

$$\begin{aligned} &\text{runtime}_1 \in C_1 | A = a_1, R = r^*, P = p, D = d < \\ &\text{runtime}_2 \in C_2 | A = a_2, R = r^*, P = p, D = d \end{aligned} \quad (4)$$

where C_1 and C_2 are the respective experimental conditions, including the runtime, under which a_1 and a_2 can discover an optimal solution r^* of an instance

[3] We use $\text{Prob}(x)$ to denote the probability of x, to avoid possible confusion with the variable for the problem in our framework.

of a problem p and data d. Alternatively, the statement could be understood as that under the same conditions, a_1 is expected to find a better solution than a_2, which can be formalized as

$$Prob(r_1 > r_2 | P = p, D = d, C = c) > Prob(r_2 > r_1 | P = p, D = d, C = c) \quad (5)$$

where r_1 is the solution found by a_1 for an instance of P and D under constraints C, r_2 the solution found by a_2 in the same scenario, and the ">" sign in this case indicates "better quality". We could also mean that the probability of a_1 finding an optimal solution is higher than the probability for a_2, as in

$$\begin{aligned} &Prob(r = r^* | A = a_1, P = p, D = d, C = c) > \\ &Prob(r = r^* | A = a_2, P = p, D = d, C = c) \end{aligned} \quad (6)$$

for the same instance of P and D and under the same constraints C. Finally, another different interpretation is that a_1 can operate under more relaxed assumptions than a_2, that is

$$\begin{aligned} &\exists\, p \in P, d \in D, c \in C\ : \\ &Prob(R = r | A = a_1, p, d, c) > 0, \\ &Prob(R = r | A = a_2, p, d, c) = 0 \end{aligned} \quad (7)$$

that is, a_1 can potentially reach a target solution r under assumptions on the problem, the data distribution or the experimental conditions that prevent a_2 from doing so. Again, our hypotheses (H1) and (H2) make it possible to apply Equations 3 to 7 both at the algorithm and the component level.

In the experimental community, the most immediate example of analysis of algorithms is probably the comparison between metaheuristics on a certain problem. For instance, many works have compared simulated annealing and tabu search on different problems such as the QAP [9, 67], facility location [5] or constraint solving [58], or the many heuristic algorithms proposed for the flowshop problem [120, 106]. Usually this corresponds to the formulation of Equation 5, choosing the algorithm $a \in A$ that gives the best results for a problem and a set of instances, under a certain computational environment

$$\arg\max_{a \in A} Prob(R | P, D, a, C). \quad (8)$$

Some works have studied in detail the impact of each component of an algorithm, such as simulated annealing [73, 74, 52], tabu search [137] and ant colony optimization variants [130, 100]. An a priori understanding on the functioning of some component in SA or PA can be an essential part in the explanation of the results in addition to information at the mediator level, and the only way of explaining the results when no features are computed. For example, the late acceptance hill climbing cannot accept a solution that is worse than anything the algorithm has encountered in the past. Thus, it has a limited diversification potential and cannot accept arbitrarily bad solutions, contrarily to the Metropolis criterion [20, 51]. These insights at the component level make it possible to understand the behaviour of the entire algorithm. Subsequent analyses have then focused on the particular impact of several options for one specific component

or parameter of an algorithm, such as the tabu list in tabu search [35], the acceptance criterion in large neighbourhood search [122] and simulated annealing [51], or the initial population strategies [93] and crossover operators [94] in genetic algorithms. Other studies have instead evaluated the results obtained by algorithms of different complexity, such as iterated local search algorithms on flowshop scheduling problems [105], and evolutionary algorithms on knapsack, Ising model and MaxSAT [50]. From the perspective of this work, all these works aim to answer the same questions formulated in Equations 3 to 7, either at the A or at the SA level, simply using different tools.

On top of the different angles explored in the various works, theoretical analyses have for the most part focused on one specific problem and one particular instance distribution. This is another explanation for which the results can be contrasting. In [57], Hajek and Sasaki proved that there is no monotonically decreasing cooling scheme that is optimal for specific instance of the matching problem. In our framework, we have $p = \mathsf{MATCHING} \in P$ and a particular $d \in D$ that includes a specifically constructed distribution and a worst-case initial solution. The authors consider sa_d and $\text{sa}_{nd} \in \text{SA}$, respectively a decreasing and non-decreasing cooling scheme defined as components of the search part of the algorithm[4] that obtain results r_d, $r_{nd} \in R$. The particular implementation of the decreasing cooling scheme is not important in this work. In fact, Hajek and Sasaki prove that it exists a non-decreasing cooling scheme sa_{nd} such that for every monotonically decreasing cooling scheme sa_d, and using the Metropolis acceptance [92] to evaluate a randomly selected candidate solution (also in SA), we have $r_{nd} > r_d$ and for average running time in C polynomial in the size of the instance.

Wegener proved instead that on natural instances of the Minimum Spanning Tree, simulated annealing, defined as one specific decreasing cooling scheme $\text{sa}_d \in \text{SA}$, outperforms sa_{nd}, in that work called the Metropolis algorithm [138], that is, $Prob(r_d = r^*) > Prob(r_{nd} = r^*)$ within a bounded number of steps. Later Meer proves the same result for a specifically constructed TSP instance [90].

These results can, of course, be easily reconciliated by noting that the results depend on the specific problems and problem instances, which generate different landscapes. Thus, the results have to be explained by using some variables at the mediator level, as a surrogate for the unobservable landscape:

$$R|P, D, \text{SA}, \text{PA}, \text{FA}, \text{FS}, C. \tag{9}$$

Here we also mention a parallel and complementary line of research to the quantitative analyses we focus on in this paper, with recent works such as [140, 22, 23, 127] that qualitatively study and compare algorithms to show functional similarities and differences between them. This trend arose as a reaction to the proliferation of metaphor-based metaheuristics that did not include any actual algorithmic novelty [126]. The first mathematical proof of the equivalence

[4]Hajek and Sasaki reason in terms of *temperature schedule*, that is, sequences of temperature values. However, in practice we construct a function that computes such values from the instance.

of a supposedly novel metaheuristic came from a discussion between Weyland and the authors of the harmony search, that was demonstrated to be perfectly equivalent to, and its performance always bounded by, the $(\mu + 1)$ evolutionary strategy [139, 140]. Camacho, Dorigo and Stützle first showed that the intelligent water drops algorithm is a special case of ant colony optimization [22, 23], and subsequently that grey wolf, firefly and bat algorithms are variants of particle swarm optimization [24] In these works it is essentially shown how $\text{sa}_1 \in \text{SA} \equiv \text{sa}_2 \in \text{SA}$, and therefore $Prob(R|P, D, \text{sa}_1, C) = Prob(R|P, D, \text{sa}_2, C)$.

Qualitative analyses have been applied also to the experiments reported in some works. In notable cases such as the harmony search proposed to solve the sudoku [140] or a modification of the Clarke-Wright heuristic for the Capacitated Vehicle Routing Problem [127], it is proved both theoretically and experimentally that

$$Prob(R|P, D, A, C) = 0, \tag{10}$$

that is, the original papers report results that are impossible to obtain. This is therefore an indication of possible mistakes or misconducts.

3.2. Instance and landscape analysis

As we mentioned before, understanding the conditions encountered by an algorithm during its search is a necessary step to ultimately understand how an algorithm works.

It is very common for an algorithm that performs well on a certain set of problem instances to perform poorly on a set with different characteristics, and possibly viceversa. In practice, it is commonly observed that some problem instances and some problem classes are intrinsically harder than other ones. In our causal framework, we can first of all disambiguate what it is meant with "difficult". One possibility is that for two data distributions $d_1, d_2 \in D$ of a problem $p \in P$, a randomly chosen algorithm $a \in A$ is expected to obtain better results on the landscapes generated by (p, d_1) than on those generated by (p, d_2), for some computational constraints $c \in C$. Or, alternatively, for $a \in A$ to obtain results $r_1, r_2 \in R$ of the same quality on $d_1, d_2 \in D$ the experimental conditions $c_2 \in C$ on d_2 need to be more relaxed (e.g. a longer runtime) than $c_1 \in C$ for d_1. Still another possibility is that, given $d_1, d_2 \in D$, the set of algorithms $a_1 \in A$ that can obtain a target solution quality $r \in R$ for d_1 is greater than the set of algorithms $a_2 \in A$ that can obtain the same solution quality $r \in R$ for d_2, under the same constraints in $c \in C$. In all these cases, we normally consider d_2 to be a more difficult scenario than d_1.

One approach to understand the conditions for an instance to be "difficult" is to figure out some characteristics that the instance should have, in order to make it harder for a search algorithm to converge to the optimal solution. In our framework, this means to distinguish the results in R by conditioning them on some characteristics $d \in D$. For example, for the TSP when the ratio of the number of clusters to the number of cities is between 0.1 and 0.11 the Lin-Kernighan algorithm takes nearly six times longer to converge to a "good" solution than on a randomly generated instance [125]. Those values are said to yield a *phase transition* between practically "easy" and "difficult" instances [26]. Instances can be generated "difficult" for an algorithm, to study the relationship between instance properties and algorithm performance [124].

To have a better chance of explaining the results, we can move to the landscape and features level. This is equivalent to expanding the relationship between P, D and R to include also a selection of variables in FA and FS to represent the landscape L, thus having

$$R|P, D, \text{FA}, \text{SA}, C. \tag{11}$$

In many cases we can, however, omit the experimental conditions C. Some properties of the fitness landscape have been related to the difficulty of a problem instance. Several measures such as ruggedness, autocorrelation, fitness distance correlation, are used as proxies for higher-level properties of the search space like convexity or multimodality [115, 1, 91, 80].

Local Optima Network analyses study the relationships between solutions entailed by the neighbourhood function, observing local optima and the paths to escape from them [34, 18]. This kind of analysis aims at understanding global properties of the landscape L by computing some of its features

$$L|P, D, \text{FA}. \tag{12}$$

In case of complete exploration of the search space, we are forced to not impose any computational constraint or limitation.

Fitness Landscape Analysis can also be used for automatic selection and configuration of algorithms [16, 66, 114, 85]. In [77] the landscape is translated into a set of constraints, to understand how complex a local search needs to be on different landscapes.

A smaller number of works have combined algorithmic analysis with problem and instance features. The impact of search space features for the Job Shop Problem on the performance of tabu search was explored in [136]. In [108] six metaheuristics form the portfolio for an algorithm selection problem for the PFSP based on both problem and landscape features. Three different objective functions are instead considered in [54], where a cross-problem analysis identifies the landscape conditions necessary for two variants of simulated annealing to perform well.

Some other works have instead applied problem-specific knowledge knowledge to the design of algorithms. In [4] high quality solutions of Vehicle Routing Problems are analyzed to understand what makes them good, in order to then develop a Guided Local Search that is biased towards solutions that "look like" the good ones [3]. Optimal or high quality CVRP solutions have compact routes and "narrow", with few intersections and short edges that connect to first and last nodes of the routes to the depot. A knowledge-based algorithm can therefore penalize solutions that have, for example, overlapping routes, or nodes in the route that are spread apart. In [98] an analysis of efficient schedules for small instances of the no-idle Permutation Flow Shop Problem lead to the definition of *superjobs*, that is, subsequences of jobs likely to appear in high-quality solutions, that are treated as a single job to reduce and smoothen the search space. Two algorithms based on Iterated Greedy are then augmented with a learning subprocedure aimed at discovering these superjobs.

3.3. Modeling the algorithm behaviour

For further analysis and applications, it is necessary to represent in a mathematical form the behaviour of an algorithm.

To completely separate the general algorithmic template from its implementation for a problem, it is often assumed that the algorithm has no access to any information about the instance, and therefore the algorithms treats the problem as a black-box function [43, 44, 40]. This corresponds to ignoring any prior observation of FA, FS or L, and to learn the distribution of some target variables, being in R, L, in A, or at the component level, during the progress of the search. The use of artificial problems, whose characteristics are perfectly known in advance, make it possible also to evaluate how close the behaviour of the algorithm in the black-box scenario gets to the optimal one, computed analytically, and in some simple cases to generate optimal algorithms [84, 36].

Surrogate models can model knowledge to approximate the outcome of experiments [83, 96, 97, 121]. A surrogate model can be considered an alternative model $(P, D, \mathrm{PA}, \mathrm{SA}, C, A, L', \mathrm{FS}', \mathrm{FA}', R')$ that approximates the real model $(P, D, \mathrm{PA}, \mathrm{SA}, C, A, L, \mathrm{FS}, \mathrm{FA}, R)$. This is useful not only in case of expensive objective functions, but also to filter candidate solutions and guide the search by focusing on the most promising ones [27, 32, 17]. A notable case is algorithm configuration, where the expensive evaluation of an algorithm is replaced by a query on an Empirical Performance Model (EPM) that predicts the solution quality of a configuration on an instance, and only the most promising configurations are actually evaluated [68, 46, 112].

The data collected in the selection and configuration tasks, either on actual experiments or using surrogate models and EPM, is particularly useful to analyze the performance of algorithms. For example, several techniques can be used to estimate the importance of features or algorithmic components and parameters. Determining the most important feature or subset of features means to find the subset $\mathrm{FA}', \mathrm{FS}' \subseteq \mathrm{FA},\mathrm{FS}$ such that the proportion of influence mediated from P, D to R remains approximately the same than using FA,FS. Finding instead the most important parameter means to estimate the subset $\mathrm{PA}', \mathrm{SA}' \subseteq \mathrm{PA}, \mathrm{SA}$ that has the greatest impact on the results, that is,

$$\underset{\mathrm{PA}',\mathrm{SA}' \subseteq \mathrm{PA},\mathrm{SA}}{\arg\max} \Delta R | P, D, \mathrm{do}(\mathrm{PA}'), \mathrm{do}(\mathrm{SA}'), C \tag{13}$$

with a slight abuse of the $\mathrm{do}(\cdot)$ notation to indicate a generic intervention on the variable rather than a precise value assignment. These are again different questions, but they share the innately causal goal of estimating the effect of an intervention. Hence, the following methods can be applied to both tasks.

Functional ANOVA [60] is one such approach to quantify the importance of the variables of a function and their interactions according to the proportion of variance explained. One possible alternative is to apply forward selection to the features observed in the EPM [69]. Random forests [19] and surrogate models based on them also provide a native way of estimating the variable importance, that has been used to compute importance of parameters [70, 113, 13, 52]. Ablation analysis [48] is another technique proposed to determine the subset of parameters having the highest impact on the results. It is a path

relinking procedure that connects two configurations flipping one parameter a time, choosing the one that yields the maximum gain (or the minimum loss) from the previous configuration. Data obtained in the configuration phase can be analyzed also to find similarities in the final configurations [141], to infer what makes a configuration perform well on a scenario.

The multilevel regression proposed in [30] uses a two-level regression to relate the algorithms components with the problem characteristics. In the first level, the algorithm components and parameters are used to predict the objective function value; in the second level, the problem features are used to predict the coefficients of the first regression. This is equivalent to performing a series of tasks, the first one being

$$R|\mathrm{PA}, \mathrm{SA}, C \tag{14}$$

followed by

$$x|\mathrm{FA}, \mathrm{FS}, P, D \; \forall x \in \mathrm{PA}, \mathrm{SA}. \tag{15}$$

The use of regression in this order is aimed at explaining first the results in terms of the algorithm used, and then the algorithm in function of the landscape, represented by the features. We note that this approach is justified by the implicit assumption of a set of causes and effects that prescinds from the mathematical method itself, which alone is not sufficient to determine the causal relationship between the entities. It provides a more detailed analysis that the functional ANOVA, relating single instances and configurations to their results, instead of their average values [29].

Studies on the scaling behaviour of algorithms such as [65, 45] instead aim to predict the results obtained by an algorithm on instances larger than those that have been observed. This corresponds to inferring $R|P, D, A, C_2$, using data from a model (R, P, D, A, C_1) with C_2 in this case being a longer runtime than C_1.

3.4. Discussion

In this partial literature review, we have seen how virtually any method proposed to explain the behaviour of an optimization algorithm can be represented as an inference task on a subset of entities of our causal framework. By outlining the research question of different lines of works we have also shown how they can be related to each other. This is very important to navigate the research literature on this subject, and understanding precisely why two works that deal with the same algorithms on the same problems can offer different conclusions.

Of course, how to perform the inference task in practice depends on the observable variables, the tools available, and the computational environment that can be used. Nonetheless, we argue that in order to properly understand and explain the behavior of an algorithm and its results, it is required to collect information at the mediator level, that is, the features that represent the landscape. This is also a necessary step to furthen the generalization and automation of algorithm instantiation, an example of which we will discuss in the next section. This perspective is shared by other works such as [42], who effectively outline a sequence of operations that can be considered an effective implementation of one such inference tasks.

4. First application: A unified view of selection and configuration

Given a set of instances arisen from a problem P and data D, a set of algorithms A and a cost function $c(\cdot)$ to measure the quality of each algorithm $a \in A$ on P, D, the Algorithm Selection Problem (ASP) [116, 81, 78] aims to find a mapping $\mathcal{S} : P, D \mapsto A$ such that $\forall d \in D$, $c(a(d))$ is minimized, or, in other words, to find the algorithm in A that obtains the best results on each instance.

ASP is usually tackled by computing a set of proxy problem features that represent the problem instance "well enough" and thus allow to find a mapping $\mathcal{S}$ that works "well enough" in practice. Features can be of various kinds. For continuous optimization problems, usually fitness landscape features are considered [91, 16, 99, 79]. For combinatorial optimization problems, very often features are problem-specific (or even class-specific) characteristics, such as statistics of the cities in TSP-related problems, or of clauses in SAT, thus requiring a certain degree of problem-specific knowledge. Efforts to use fitness landscape features to select algorithms for combinatorial optimization problems include [108] for flowshop scheduling, or [33] for the QAP. The selection of the set of features is crucial for the performance of the mapping. Problem-specific features can provide deep insights on the specific instance, at the cost of sacrifying generality of the mapping, while for more general features it is more difficult to pinpoint the ones that can give a reliable mapping.

Some authors have instead explored the possibility of using the automatic feature extraction of deep neural networks to bypass the manual definition of features in AS, on the bin packing problem and the TSP [2, 123]. The drawback is, of course, the difficulty or impossibility to a posteriori understand and explain the selection.

Given a set of instances that arise from a problem P and data D, an algorithm $a \in A$ with a set of parameters θ_a and a cost function $c(\cdot)$ to measure the quality of a on D, the Algorithm Configuration Problem (ACP) aims to find the set of parameters $\theta_a^\star$ such that $c(a)$ is minimized on P, D. That is, the goal is to find the best configuration of parameters for a on the instances. This problem is also known as *parameter tuning* or, in machine learning, *hyperparameter tuning*. Contrary to ASP, in ACP features are not necessarily considered. The problem is often modeled as a black-box stochastic optimization problem, so the best configuration $\theta^\star$ is the one which obtained the best results on a set of training instances, but no insights are given. However, by comparing different outcomes of ACP across different scenarios, we can infer explanations for the different results. The inclusion of features in the configuration process gives instead the Per-instance Algorithm Configuration Problem [76, 10, 11], which will be the focus of the rest of this section.

Automatic approaches for the ACP can be naturally applied to the task of automatically design algorithms, following the Programming by Optimization paradigm [63]. This makes it possible to tailor an algorithm for a specific scenario, performing extensive experiments and improving the final performance while reducing the burden for the user

The Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [132, 82] is the problem of selecting an algorithm and its parameter

configuration that best solve a problem, or $\arg\min c(a, \theta_a^\star)$, for the $a \in A$. It differs from ASP because in the latter the configuration task is omitted, and from ACP because it consider a set of algorithms rather than a single one, and finally it differs also from separately running a selection phase and a subsequent configuration of the outcome of the selection. It appeared more recently in the literature with respect to ASP and ACP, mostly due to the higher computational cost entailed.

Nonetheless, here we argue that these three problems can be conceptually represented as an inference task on the same set of variables. Informally, this can be understood by considering the portfolio A for the ASP as a categorical parameter of a super-algorithm, whose values are the different algorithms in the portfolio. With (H1) and (H2) we assume that we can instantiate at least one, and possibly more, algorithms from A. The mapping $\mathcal{S}$ searched for in Algorithm Selection as presented above can thus be described as $A|P, D, \mathrm{FA}, \mathrm{FS}, C$. Since FA, FS already depend on observing P, D, and the selection is made by observing the results R for the alternatives evaluated, we can define ASP as the inference task $A|R, \mathrm{FA}, \mathrm{FS}, C$. The emphasis on the mediators FA, FS emphasizes the "white-box" characteristic of ASP.

With (H1) and (H4) we also claim we can configure an algorithm using PA and SA. The usual black box formulation of the Algorithm Configuration Problem is $\mathrm{PA}, \mathrm{SA}|D, C$. Again, the final decision is based on results R observed in the tuning phase. Taking into account that the fact that the algorithm already works for a given problem P is usually assumed, we have $PA, SA|P, D, R, C$. For a proper definition of PA, SA, the same formulation applies to the PbO paradigm for Algorithm Design.

Therefore, we see the relationship between ASP and ACP. They are similar problems, in the sense that they are both inference problems based on problem and instances, and the outcome depends on the results (in fact, the racing algorithm [88] proposed in 1997 for model selection was subsequently used for configuring algorithms [15]). They instead differ in (i) whether we use fully configured algorithms (in A, for ASP) or building blocks (in PA and SA, for ACP), and (ii) if we can observe (and use in the selection or configuration process) some features in F or L.

The separation is, however, neither crisp nor immutable. By observing any feature in F or L in a configuration or PbO task we open the black box and move towards a per-instance configuration approach. By allowing the configuration of numerical parameters of the algorithms in a portfolio for a selection task we have the CASH problem, which can also be instantiated as $\mathrm{PA}, \mathrm{SA}|R, F, L, C$. In practice, therefore, the question is what tools should we use for our application, and the answer depends on the possibility of observing features, the extent to which our algorithm or set of algorithms can be parameterized, and the computational constraints we have.

5. Second application: Per-instance configuration and transfer learning

Transfer learning is the application of knowledge obtained on one domain onto a different domain [133, 107]. It is an important task in machine learning, where

it aims to reuse of past learning tasks as starting point for new tasks. Transfer learning is an inherently causal task, because the conditions that determine whether some information is transferrable between two tasks can be determined by observing the causal structure of the tasks [110, 7]. In particular, we have *direct transportability* between two tasks that share the same set of causal relationships [110].

In this section, we show how our causal framework can be used to transfer algorithm configurations obtained for some problems and instances into new scenarios. This is an open problem of great interest, because one of the limitations of many existing configuration approaches is the difficulty of making use of past experiments for new configuration tasks. Transferring configurations onto new scenarios would make it possible not only to speed up new configuration tasks, but also to exploit the vast amount of data collected in the past. This would also be extremely useful in scenarios where a configuration is not possible or not convenient, such as when the evaluation of the objective function is extremely expensive, when few instances are available, or in case of a problem that have to be tackled only once.

In a typical transfer learning task in machine learning, we typically consider one model and different distributions. The question is therefore how to translate the typical optimization scenario, consisting in algorithms, problems and instances into distributions. The starting point to answer this is to note that transfer learning and per-instance configuration are the same task. This can easily be seen with a derivation analogous to the one between AC and AS in Section 3. From our framework in Figure 1 we see that $\{\mathrm{FA}, \mathrm{FS}\}$ is a set of mediator variables between the problem instances $\{P, D\}$ and the results R (since we assume the landscape L is unobserved, an assumption derived from (H3)). Thus, for a specific problem $p \in P$, the goal is to obtain the best configuration for each instance, that is

$$\{\mathrm{PA}_d, \mathrm{SA}_d\} | \{p, d, \mathrm{FA}_d, \mathrm{FS}_d, R, C\} \ \ \forall d \in D_p \tag{16}$$

where FA_d and $F\mathrm{S}_d$ are the features computed for every instance d of problem p, to distinguish them from the sets of features FA and FS computed over all the instances, for example, the instances we use in the training of the model.

Equation 16 explicitly includes instantiation of a problem with its data. Following our argument in Section 3, we can remove this explicit dependency by observing a suitable set of features in $\{\mathrm{FA}, \mathrm{FS}\}$ that are representative enough of the information contained in $\{P, D\}$. We thus obtain

$$\{\mathrm{PA}_d, \mathrm{SA}_d\} | \{\mathrm{FA}_d, \mathrm{FS}_d, R, C\} \ \ \forall d \in D_p. \tag{17}$$

Hence, we see that the observation of this suitably representative set of features makes the starting objective function irrelevant, allowing us to characterize scenarios only at the feature level. Of course, this requires also the features to be problem-independent for the procedure to work across different objective functions, which is why we are going to focus only on landscape features, as they can be more easily defined to suit our goals.

The inference process is reported in Figure 2. The training data is composed of data collected on the training instances, pairing the features computed on each

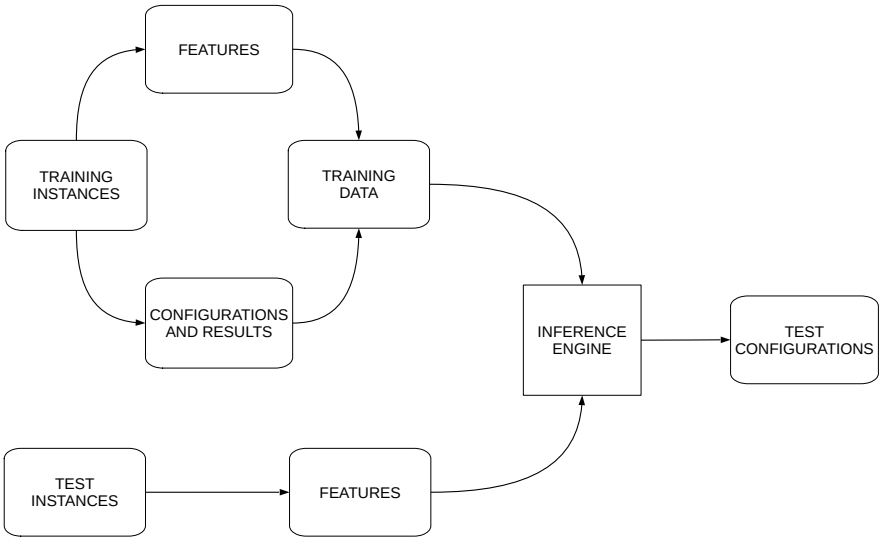

Figure 2: Inference process for the transfer of configurations.

training instance with the configurations tested on it, and the results observed. For each instance, only the configuration that obtains the best results is kept. This data is fed to an inference engine; in principle, any engine can be used for the tasks. The inference engine will then take the features computed on the test instances and compute suitable configurations to use on the test instances.

5.1. Transfer of configurations of a fixed temperature simulated annealing

We follow our procedure to find good configurations for a fixed-temperature variant of simulated annealing (FTA). This is a very simple SLS that always accepts improving moves and accepts worsening solutions probabilistically, weighting the relative worsening in solution quality by a scaling parameter called temperature. Contrarily to the classic simulated annealing, which alters the temperature during the search, FTA always uses the same temperature value. Following our earlier work [54], we define FTA as a two-parameter SLS, the fixed temperature value (real-valued factor in $[0, 1]$ that rescales the average gap between consecutive solutions observed in a preliminary random walk), and the neighbourhood exploration, a binary parameter that determines whether the next solution to be evaluated in the neighbourhood of the incumbent has to be selected randomly or following some ordering.

We consider four objective functions: the Quadratic Assignment Problem (QAP), the Permutation Flowshop Problem under the makespan objective (PFSP-MS), the Permutation Flowshop Problem under the total completion time objective (PFSP-TCT), and the Traveling Salesperson Problem (TSP). For each objective function we have two instance classes. These are all permutation problems where all the possible solutions are feasible, and we can expect a similar algorithmic behaviour. In fact, in [54] we have characterized the behaviour of FTA and simulated annealing with respect to the landscape, analyzed using

Table 1: Set of landscape features used in the transfer learning. Each feature is computed with both a first-improvement and a best-improvement local search.

1	Number of moves to local optimum
2	Number of moves to local optimum, rescaled by instance size
3	Number of moves to local optimum, rescaled by neighbourhood size
4	R^2 of a linear model fit on the solution values
5	R^2 of a linear model fit on the solution values normalized in $[0, 1]$
6	R^2 of an exponential model fit on the solution values
7	R^2 of an exponential model fit on the solution values normalized in $[0, 1]$
8	R^2 of a linear model fit on the number of improving moves in the sequence of neighbourhoods traversed
9	Average proportion of neutral moves in the neighbourhoods traversed
10	Number of neutral last moves
11	Difference of features 4 and 6
12	Difference of features 5 and 7
13	Slope of the sequence of differences between best and average solution in a neighbourhood

problem-independent features. The goal is to make use of collection of experiments, either performed in the past or simulated, for example via a surrogate model, to infer good configurations for each new test instance, both for the same problem and across different problems.

Following the causal framework, problem instances and algorithmic components become conditionally dependent if one or more common descendants are observed. In our case, the observations of features in FA is the set of descendants that makes this possible. In particular, the landscape features we consider, listed in Table 1, are computed with a first improvement and a best improvement. Since the features in FA descend from one specific algorithm $a \in A$ instantiated from $\{\text{PA}, \text{SA}\}$, for which there are virtually countless valid combinations, we cannot expect for two algorithms a_1 and a_2 to observe similar values for the the respective set of features in FA. We choose therefore to use a probing algorithm a_0 that can approximate the behaviour of all or at least many FTA combinations, assuming that the set of features FA_0 observed can be considered an approximation of the set of features observed by the set of actual FTAs tested, and a first improvement can be seen as a FTA with temperature 0. Following [52], the "right" temperature value is the minimal one that guarantees the correct diversification.

The FTA algorithm is implemented in a component-based fashion in the emili framework, and an algorithm is instantiated at runtime by a desired parameter configuration [104]. We find good configurations for the various scenarios using the irace configurator, a state-of-the-art-offline configurator [87]. Each tuning is repeated 15 times, and the best configuration of each tuning is used on the entire test set. The configurations evaluated by irace during the tuning phase, and the relative results observed, are also paired with the features computed on the training instances to be used as training data for the inference task. We use MERCS to infer the test configurations [135]. Also with MERCS we generate 15 configurations for each test instance, for statistical purposes. QAP and TSP

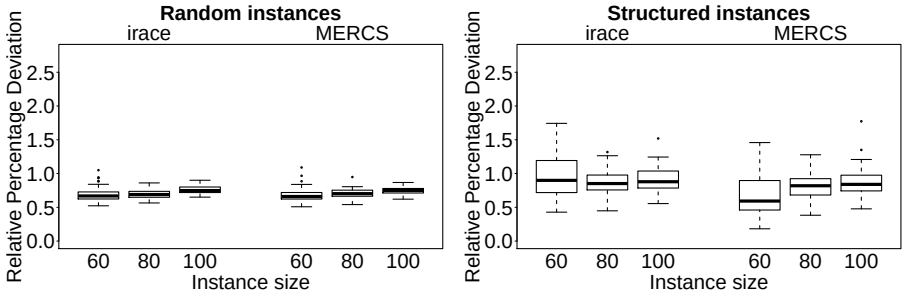

Figure 3: Results obtained in terms of relative percentage deviation from the best known solutions on the random and structured QAP instances by irace (separate tunings for each instance class) and MERCS; lower boxplots represent better results.

experiments run for 10 seconds, while the runtime for the PFSP experiments is $0.015 \times n \times m/2$, where n is the number of jobs and m is the number of machines of an instance. All the experiments are performed on a machine with Intel Xeon E5-2680 v3CPUs running at 2.5GHz, with 16MB cache and 2.4GB of RAM available for each algorithmic run, in single thread mode.

The structure of the experiments is the same in every scenario presented in the following; they differ only in the set of problems and instance considered. First we perform one tuning with irace on the training set, then we couple the training data with the instance features to infer a new configuration for every test instances. We therefore compare the results obtained on the whole test set considered by the final configuration obtained using irace, with the results obtained by the set of test configurations obtained using MERCS, each one on a single test instance. The goal of this experiment is not to compare the tuning methods, but to show that a general purpose inference engine can be a valid alternative practice for automatic algorithm configuration, using collections of past experiments. This use of problem-independent features makes it possible to apply this process to any collection of problems, with meaningful outcomes.

In the first two scenarios we perform transfer learning between QAP instances. First we consider separately two different instance classes, performing separate tunings and evaluating separately the configurations found on random and structured instances. Inference is also applied separately for the two instance classes, so configurations for the test random instances are inferred only from data obtained during the tuning on random instances, and the same is done on the structured instances. The results are reported in Figure 3, separated by instance size, comparing the solution qualities obtained by the configurations found with both irace and MERCS in terms of percentage deviation from the best known solutions. The results are very similar. Only in the case of structured instances of size 60 there is a statistical difference in favour of the per-instance approach with MERCS, with a p-value $< 3 \times 10^{-5}$, where all the other cases have a p-value greater than 0.05. This happens because for each instance class and size there is a relatively narrow interval of good temperature values that allow FTA to reach those solutions in the given runtime. irace is able to identify it, and MERCS can exploit the information collected during the tuning.

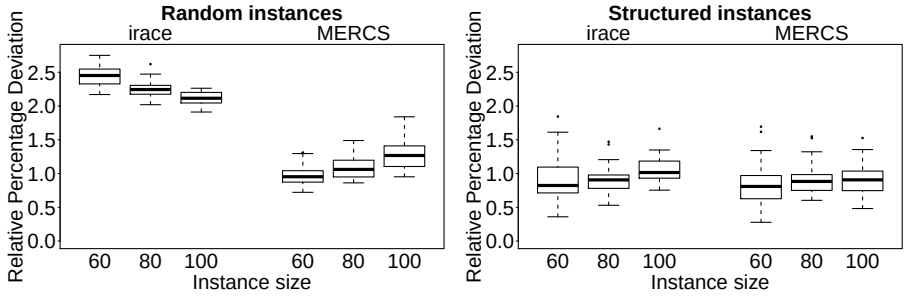

Figure 4: Results obtained in terms of relative percentage deviation from the best known solutions on the random and structured QAP instances by irace (separate tunings for each instance class) and MERCS; lower boxplots represent better results.

In the second experiment we consider the two instance classes together, and tune FTA with irace using a training set of both random and structured instances, obtaining one configuration to test on the entire test set. The results are worse than in the previous case, especially on the random instances, because the two instance classes are better tackled using different configurations. During the training phase, however, irace evaluated also configurations that are good for the random instances, but were eventually discarded because of their poor performance on the structured instances. The feature-based approach with MERCS can exploit those configurations, obtaining better results, albeit not as good as in the previous experiment. In this case, in fact, the configuration space suitable for the random instances was not properly explored during the tuning phase by irace. The results are statistically equivalent only on the structured instances of sizes 60 and 80.

5.2. Transferring configurations across different objective functions

In the third experiment, we use the feature-based approach to infer configurations for the TSP, starting from the tuning data generated by irace on the QAP and the two PFSP objectives. That is, MERCS uses data for these three problems to suggest configurations to test on each TSP instance. We consider two TSP instance classes, uniformly random (RUE) and clustered (RCE), with sizes 100, 150 and 200 cities. We created the instances using the generator from [75], with coordinates on a 1000×1000 matrix.[5] The results obtained by the configurations inferred by MERCS are compared against the results obtained by irace when tuned on a training set of TSP instances, considering the two instance classes separately but including the three instance sizes. In addition, we include the results obtained with a $2-$opt heuristic as a reference. The results are reported in Figure 5. A summary on the temperature values obtained is reported in Table 2.

In both instance classes the results obtained by the configurations found by MERCS are comparable to those found by irace for instance sizes 100 and 150,

[5]On the RCE instances the coordinates may exceed those boundaries, due to the sampling procedure of the generator. This does not cause any numerical instability, so we do not need to correct a posteriori the coordinate list.

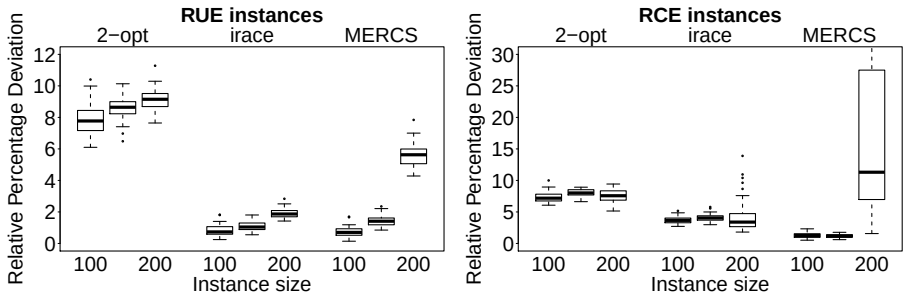

Figure 5: Results obtained in terms of relative percentage deviation from the optimal solutions on the RUE and RCE TSP instances by a 2-opt, irace (tuning on a training set of TSP instances, across all sizes) and MERCS (inference on QAP and PFSP data); lower boxplots represent better results.

Table 2: Average temperature values obtained for each instance class, with the relative standard deviation. For irace the averages are computed over fifteen tunings, for MERCS across all the temperature values for the relative class and size. T_i is the average temperature obtained by irace on the two separate TSP training sets, and is the temperature value used in the experiments reported in Figure 5. T_i' is the temperature obtained by separate tunings on different instance class and size. T_M is the average of the temperatures identified by MERCS for the instances of each size.

Class	Size	T_i	T_i'	T_M
	100		$(0.0489, 3.014 \times 10^{-3})$	$(0.0372, 6.192 \times 10^{-3})$
RUE	150	$(0.0349, 3.615 \times 10^{-3})$	$(0.0373, 2.029 \times 10^{-3})$	$(0.0437, 9.002 \times 10^{-3})$
	200		$(0.0313, 2.156 \times 10^{-3})$	$(0.0569, 6.811 \times 10^{-3})$
	100		$(0.054, 4.439 \times 10^{-3})$	$(0.0337, 6.247 \times 10^{-3})$
RCE	150	$(0.018, 5.738 \times 10^{-3})$	$(0.0443, 3.908 \times 10^{-3})$	$(0.0409, 6.217 \times 10^{-3})$
	200		$(0.0134, 1.603 \times 10^{-3})$	$(0.0399, 8.95 \times 10^{-3})$

Table 3: Distances (rounded to the nearest integer) for selected neighbours in each instance class. For each test instance we compute the average and standard deviation of the distance to the nearest neighbour, the distance to the 50%-th neighbour when ranked by distance, and the distance to the farthest point. The values reported are the average across each instance class and size.

	1-NN		50%-NN		Farthest point	
Class	Average	St.Dev.	Average	St.Dev.	Average	St.Dev.
RUE 100	52	29	528	46	987	144
RUE 150	42	23	529	43	1004	144
RUE 200	36	20	530	40	1013	144
RCE 100	23	19	171	34	400	63
RCE 150	15	14	139	27	338	52
RCE 200	16	13	328	27	725	70

while they differ greatly, in both solution quality and variance, for size 200. On the RUE instances the results are not significantly different for size 100 (p-value of 0.2059), with an average solution quality around 0.7% worse than the optimal solutions. For size 150, the solutions found by the configurations inferred by MERCS have an average of 1.42% RPD, compared to a 1.1% of the solutions found by the configurations obtained by irace. For size 200, the results are considerably worse, with an average RPD of 5.6%, but still better than those obtained with a $2-$opt heuristic.

On the RCE instances, for sizes 100 and 150 the solutions found by the configurations obtained with irace are around $3-4\%$ worse than the optimal solutions, while those found by the configurations inferred using MERCS are on average 1.2% worse than the optimal ones. On size 200, instead, the quality of the solutions obtained using MERCS is drastically worse, even with respect to the results obtained with a $2-$opt.

The explanation can be found by observing the instances generated. Representative instances for each class and size are reported in Figure 6. TSP instances, overall, have a landscape that is globally very different from the landscape generated by the QAP and the two PFSP problems of Section 5.1. In particular, a random TSP solution can be expected to have an RPD of 120% from the optimal solution, while on the QAP the initial RPD rarely exceeds 50%. A local search, however, can exploit only local information at the neighbourhood level. A fixed temperature algorithm in particular will work well if the structure of the neighbourhoods is similar in different areas of the solution space [54].

RUE instances exhibit a "regular" behaviour, and a properly configured FTA can traverse a large portion of the search space and find good quality solutions. This is a similar situation to the random QAP landscape of our previous two experiments, so MERCS can exploit these similarities up to a certain extent and find reasonably good configurations. The basic landscape differences, and possibly the set of features considered, make it however not possible to fully distinguish the details between the instance sizes, so the temperature values are suboptimal, and in an opposite trend with respect to the correct one. On our instance sets, good temperature values decrease as the instance size grows, as the increased density of the points makes the average relative difference between neighbouring solutions smaller.

On the RCE instances we observe instead a different situation. The generator we used creates instances with $\lfloor n/100 \rfloor$ clusters, which, in our case, results in 1, 1, 2 clusters for sizes 100, 150 and 200 respectively. These are very different instances. With a single cluster, the distances between each node are actually more uniform than on the RUE instances. As we see in Table 3, on RCE instances of sizes 100 and 150 not only the average distance between nearest neighbours is smaller than on the RUE instances, but also more distant points are anyway comparatively closer. This generates neighbourhoods that are quite uniform, a situation well suited for a fixed temperature algorithm, which in fact obtains better results than on the RUE instances.

Two clusters, on the other hand, will partition the distances into small intra-cluster and much larger inter-cluster distances, as can be inferred from Table 3, making this a challenging scenario for a FTA. A careful inspection of the results on each instance shows how the solution quality found by the FTA worsens as the distance between the clusters increases. This correlation between cluster distance and results explains the huge variability obtained by MERCS. Conversely, overlapping clusters generate a landscape no different from the landscape generated by a single cluster, which results in good final solution qualities. A similar situation arises when the two clusters are adjacent, as also in this case it is possible for a FTA to perform well.

Some visual examples of different instance classes are given in Figure 6. In Figure 7 we show instead three different RCE instances of size 200, with overlapping, adjacent and distant clusters. It is interesting in particular to note the average solution quality found by the 15 configurations found by MERCS in these latter instances. On `rce200-85`, with two overlapping clusters, the average RPD is 1.69%. On `rce200-96` the two clusters are instead adjacent, and the average RPD obtained is 3.19%. Finally, on `rce200-95`, where the two clusters are very distant, the RPD is 50%. In Figure 8 we report the correlation between cluster proximity and final RPD obtained. For MERCS, that relies on features whose value is in some cases dependent on the size of the instance or of the neighbourhood, the correlation is very close to 1. Also for irace, when tuned across all the RCE instance sizes, the correlation is quite strong (0.787). When instead the tuning is done only on the RCE instances of size 200, the correlation on the test set is negligible; in this case the results are better for instances with average distance between clusters, and worse for overlapping or extremely distant clusters.

These results reflect the fact that different point distributions generate different landscapes, and in some cases they may resemble very closely landscapes observed in different situations, making it possible to obtain meaningful configurations from past experiments. On the other hand, on a completely new landscape our method fails to produce a valid configuration, and the too high temperature value prevents the FTA to converge to any decent solution.

5.3. Discussion

In this section, we have described the per-instance configuration task in terms of our causal framework, and have shown how to relate it to the transfer learning more commonly encountered in the machine learning literature. We have also reported a proof-of-concept that shows how existing tools can be used to

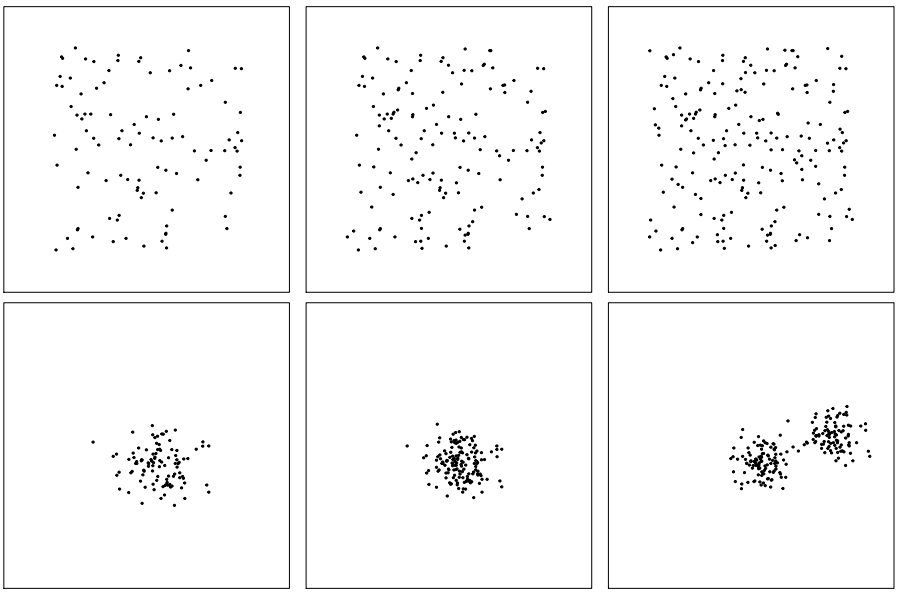

Figure 6: Sample TSP instances from our test set. Top row, from left to right: RUE 100, RUE 150, RUE 200. Bottom row, from left to right: RCE 100, RCE 150, RCE 200. These instances have been generated using the same random seed.

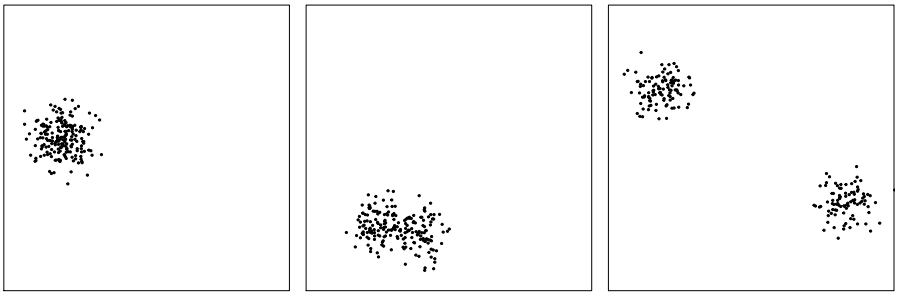

Figure 7: Different cases from the RCE TSP test instances with two clusters. From left to right: `rce200-85` with overlapping clusters, `rce200-96` with adjacent clusters, `rce200-95` with distant clusters.

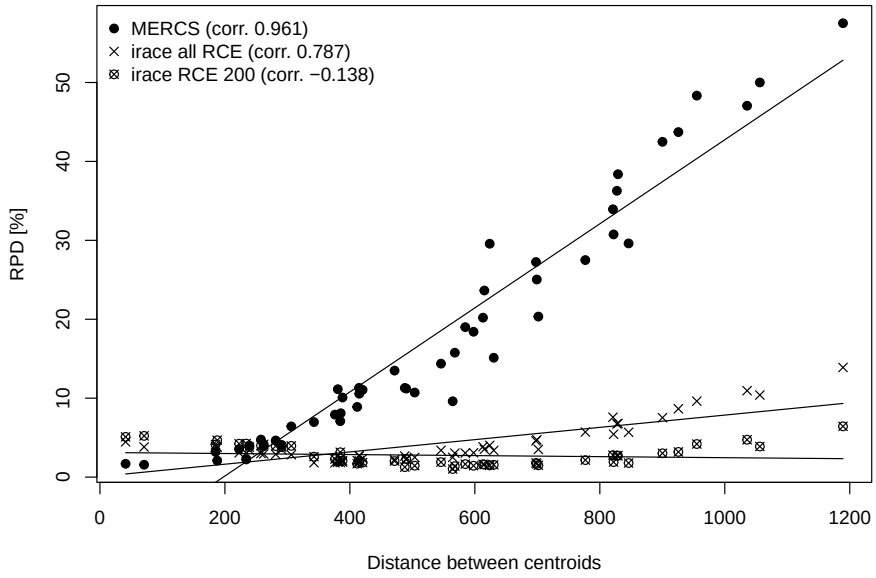

Figure 8: Correlation between cluster distance and RPD obtained on the RCE 200 test instances by MERCS, irace tuned across all the RCE instance sizes, and irace tuned only on RCE 200 instances.

successfully perform this task on various scenarios, including transferring configurations across different objective functions. In particular, this is made possible by the the observation of problem-independent features that represent the conditions encountered by an algorithm during the search. This is of particular interest especially in situations where past experiments are available, but it is not possible or not convenient to perform a focused configuration task. This may be in case of extremely expensive function evaluations, a limited budget, or the impossibility to obtain a homogeneous or representative training set of instances.

In the QAP experiments we have seen how our approach can make us of past experiments to match or even outperform a state-of-the-art offline batch configurator, when our starting data is sufficiently representative of the testing scenario.

We have then obtained configurations for TSP instances, starting only from QAP and PFSP observations. The results obtained clearly demonstrate the viability of our approach. On the RUE instances, and even more so on the RCE ones with one cluster, the results can be considered very good. In these cases, MERCS was able to make use of data from similar landscapes, that were generated by different objective functions. This strenghtens the validity of our working hypothesis (H3), in that only the landscape suffices to characterize the behaviour of an algorithm, and the observation of enough information about it can "cut off" the information flow from the variables in P and D. On the clus-

tered instances of size 200, instead, the results were drastically worse, because either the landscape generated was too different from anything observed in the training data, or the features measured were not sufficient to properly represent the information needed.

Our results can of course be improved in several ways. First of all, a larger collection of problems, instances, features and algorithms increases the likelihood of observing new situations to use in future tasks. It is in fact unlikely that the current training base would allow for the same quality of results on other kinds of problems, in particular problems where the solution is not, in principle, a complete permutation. A more focused choice of features, achieved by feature selection or by the inclusion of additional relevant ones, can also improve the results. We also note how the inference engine we used, MERCS, is not a causal tool. It is possible that an alternative procedure that considers several configurations for every test instance, for example by means of a surrogate model, could obtain better results. The use of a proper causal inference engine could also be better suited for this task. Finally, the failure of our configurations on the TSP RCE instances of size 200 strongly indicates how our procedure needs to be augmented with some mechanism capable of evaluating the suitability of the configuration for the test scenario. The starting point to build such a mechanism would be to analyze the features computed and to relate them to the results obtained, to define a distance metric to not trespass in order for the procedure to output a test configuration. Another possible direction is to determine whether, based on the features computed, the test instance can be considered an anomaly with respect to the starting data, in which case the procedure should alert the user, rather than returning a configuration almost guaranteed to fail.

6. Conclusions

Understanding the behaviour of a stochastic local search (SLS) algorithm is an open question of fundamental importance in operations research and artificial intelligence. There is a huge corpus of literature aimed at this goal, but it is difficult to navigate it because of the different perspectives offered by each work, making it more difficult to narrow bridge the gap between theory and practice. In this work we have proposed a causal framework to model the relationships between the elements involved in the solution of an optimization problem, in order to relate all these approaches in a unified perspective. We have also demonstrated how our approach could be useful to automatically configure an algorithm for an unseen problem, using tools that are already available. Albeit a proof-of-concept, our results prove that the systematic organization of knowledge that already exists is useful to both understand how algorithms work, and to advance the development of algorithms.

The framework is a formalization of our current understanding of the behaviour of optimization algorithms. It is therefore a representation of our subjective interpretation, and thus open for debate in case of disagreements. However, one of the main advantages of using such a model is the possibility of pointing out precisely what the subject of the disagreement is, making it more likely to have a productive discussion about the subject. This applies both at the general

level, that is, on our general understanding of how algorithms work, and when considering specific research works.

While we focused on unconstrained combinatorial problems and SLS algorithms, the framework can be extended to include other classes of optimization problems, such as constrained or continuous ones, and other classes of exact and heuristic algorithms. The conceptual structure of the causal framework makes it possible to relate different works, and include information from different sources. This makes it possible to design a knowledge-based system, an extension of the currently available algorithmic frameworks that can include more and more algorithmic components and problem data, to eventually be able to automatically instantiate a valid algorithm for any unseen scenario. We have already discussed several possible directions to improve the performance of our transfer learning procedure. Additionally, the framework can also be used in a dynamic fashion to perform online configuration, selection and design based on local information, to tailor the resulting behaviour in an optimal way for different shapes of the landscape.

Having explicitly outlined the relationships between the elements involved in the solution of an optimization problem, we can easily define new research questions. A natural one is quantifying how well the set of observed features represents the landscape. While in Section 5 we ignored this issue, it is a fundamental step for applying our transfer learning procedure in a reliable way. This questions can be answered by performing *mediator analysis*, to estimate the proportion of the information from P and D to R that is effectively mediated by FA and FS. In fact, our working hypothesis (H3) that says that a search algorithm traverses a landscape, regardless of how it was generated, in practice can be applied only if the information we collect about the landscape is truly representative of the real one.

Another possible direction is the estimation of the equivalence between algorithms, in order to establish how an algorithm a has to be composed for it to obtain statistically equivalent results to another algorithm b, or to understand when it cannot reach the same level of performance. The theoretical equivalence or non-equivalence of SLS algorithms is a long-standing research direction in the theory of optimization algorithms and the systematic collection of data would make this direction both grounded on real experiments and scalable, thanks to the use of automated tools.

A systematic comparison of artificial and natural landscapes based on experiments with automatically generated state-of-the-art algorithms can advance our understanding about the applicability of theoretical results to the development of algorithms for real-world problems. Similarly, it is possible to define an experimental protocol to answer the question of whether and how different classes of problems generate different landscape. The collection of data from different NP-hard problems may also be used in conjunction with the polynomial-time reduction between them, to understand how this transformation alters the landscape, and consequently, how this transformation reflects onto the development of algorithms.

Acknowledgments

Alberto Franzin acknowledges support by the Innoviris project 2018-SHAPE-25a "PHILEAS: smart monitoring par détection de comportements anormaux appliquée aux objets connectés", and from the Service Public de Wallonie Recherche under grant n°2010235 - ARIAC by DIGITALWALLONIA4.AI. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Director. We thank Mauro Birattari for discussions that helped improving the quality of this work.

References

[1] A. A. Albrecht, P. C. R. Lane, and K. Steinhöfel. Analysis of local search landscapes for k-SAT instances. *Mathematics in Computer Science*, 3(4):465–488, 2010.

[2] Mohamad Alissa, Kevin Sim, and Emma Hart. Algorithm selection using deep learning without feature extraction. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *GECCO*, pages 198–206, New York, NY, 2019. ACM Press.

[3] Florian Arnold and Kenneth Sörensen. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research*, 105:32–46, 2019.

[4] Florian Arnold and Kenneth Sörensen. What makes a VRP solution good? the generation of problem-specific knowledge for heuristics. *Computers & Operations Research*, 106:280–288, 2019.

[5] Marvin A. Arostegui Jr, Sukran N. Kadipasaoglu, and Basheer M. Khumawala. An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. *International Journal of Production Economics*, 103(2):742–754, 2006.

[6] Anne Auger and Benjamin Doerr, editors. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, volume 1 of *Series on Theoretical Computer Science*. World Scientific Publishing Co., Singapore, 2011.

[7] Elias Bareinboim and Judea Pearl. Transportability of causal effects: Completeness results. In Jorg Hoffmann and Bart Selman, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 698,704. AAAI Press, 2012.

[8] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and Jr. William R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.

[9] Roberto Battiti and Giampietro Tecchiolli. Simulated annealing and tabu search in the long run: A comparison on QAP tasks. *Computer and Mathematics with Applications*, 28(6):1–8, 1994.

[10] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. Feature based algorithm configuration: A case study with differential evolution. In Julia Handl, Emma Hart, P. R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *PPSN*, volume 9921 of *LNCS*, pages 156–166, Heidelberg, 2016. Springer.

[11] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. Per instance algorithm configuration of CMA-ES with limited budget. In Peter A. N. Bosman, editor, *GECCO*, pages 681–688. ACM Press, New York, NY, 2017.

[12] Hans-Georg Beyer, Hans-Paul Schwefel, and Ingo Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287(1):101–130, 2002.

[13] André Biedenkapp, Marius Thomas Lindauer, Katharina Eggensperger, Frank Hutter, Chris Fawcett, and Holger H. Hoos. Efficient parameter importance analysis via ablation with surrogates. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, February 2017.

[14] Mauro Birattari, Marco Chiarandini, Marco Saerens, and Thomas Stützle. Learning graphical models for algorithm configuration. In T. Berthold, A. M. Gleixner, S. Heinz, and T. Koch, editors, *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimization Problems*, LNCS. Springer, Heidelberg, 2011.

[15] Mauro Birattari, Thomas Stützle, Luís Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, pages 11–18. Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[16] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuss. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In Terence Soule and Jason H. Moore, editors, *GECCO*, pages 313–320. ACM Press, New York, NY, 2012.

[17] Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. mlrMBO: A modular framework for model-based optimization of expensive black-box functions. *Arxiv preprint arXiv:1703.03373 [stat.ML]*, 2017.

[18] Christian Blum and Gabriela Ochoa. A comparative analysis of two matheuristics by means of merged local optima networks. *European Journal of Operational Research*, 290(1):36–56, 2021.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[20] Edmund K. Burke and Yuri Bykov. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2017.

[21] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[22] Christian Leonardo Camacho-Villalón, Marco Dorigo, and Thomas Stützle. Why the intelligent water drops cannot be considered as a novel algorithm. In Marco Dorigo, Mauro Birattari, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *ANTS 2018*, volume 11172 of *LNCS*, pages 302–314, Heidelberg, 2018. Springer.

[23] Christian Leonardo Camacho-Villalón, Marco Dorigo, and Thomas Stützle. The intelligent water drops algorithm: why it cannot be considered a novel algorithm. *Swarm Intelligence*, 13:173–192, 2019.

[24] Christian Leonardo Camacho-Villalón, Thomas Stützle, and Marco Dorigo. Cuckoo search $\equiv$ $(\mu + \lambda)$-evolution strategy – a rigorous analysis of an algorithm that has been misleading the research community for more than 10 years and nobody seems to have noticed. Technical Report TR/IRIDIA/2021-006, IRIDIA, Université Libre de Bruxelles, Belgium, 2021.

[25] Felipe Campelo and Elizabeth F. Wanner. Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances. *Journal of Heuristics*, 26(6):851–883, 2020.

[26] Peter C. Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 331–340. Morgan Kaufmann Publishers, 1995.

[27] Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, 2019.

[28] William J. Cook. Computing in combinatorial optimization. In Bernhard Steffen and Gerhard Woeginger, editors, *Computing and Software Science: State of the Art and Perspectives*, volume 10000 of *LNCS*, pages 27–47. Springer, Cham, Switzerland, 2019.

[29] Jeroen Corstjens, Nguyen Dang, Benoît Depaire, An Caris, and Patrick De Causmaecker. A combined approach for analysing heuristic algorithms. *Journal of Heuristics*, 25(4):591–628, 2019.

[30] Jeroen Corstjens, Benoît Depaire, An Caris, and Kenneth Sörensen. A multilevel evaluation method for heuristics with an application to the VRPTW. *International Transactions in Operational Research*, 27(1):168–196, 2020.

[31] Nguyen Dang and Carola Doerr. Hyper-parameter tuning for the $(1 + (\lambda, \lambda))$ GA. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *GECCO*, pages 889–897. ACM Press, New York, NY, 2019.

[32] Nguyen Dang Thi Thanh, Leslie Pérez Cáceres, Patrick De Causmaecker, and Thomas Stützle. Configuring irace using surrogate configuration benchmarks. In Peter A. N. Bosman, editor, *GECCO*, pages 243–250. ACM Press, New York, NY, 2017.

[33] Augusto Dantas and Aurora Pozo. On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem. *Theoretical Computer Science*, 805:62–75, 2020.

[34] Fabio Daolio, Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. Local optima networks and the performance of iterated local search. In Terence Soule and Jason H. Moore, editors, *GECCO*, pages 369–376. ACM Press, New York, NY, 2012.

[35] Luca Di Gaspero, Marco Chiarandini, and Andrea Schaerf. A study on the short-term prohibition mechanisms in tabu search. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence, ECAI 2006, Riva del Garda, Italy, August29 - September 1, 2006*, pages 83–87. IOS Press, 2006.

[36] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[37] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science*, 801:1–34, 2020.

[38] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. The $(1+\lambda)$ evolutionary algorithm with self-adjusting mutation rate. *Algorithmica*, 81(2):593–631, 2019.

[39] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[40] Benjamin Doerr, Timo Kötzing, Johannes Lengler, and Carola Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[41] Benjamin Doerr, Frank Neumann, Dirk Sudholt, and Carsten Witt. Runtime analysis of the 1-ANT ant colony optimizer. *Theoretical Computer Science*, 412(1):1629–1644, 2011.

[42] Johann Dréo, Carola Doerr, and Yann Semet. Coupling the design of benchmark with algorithm in landscape-aware solver design. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *GECCO Companion*, pages 1419–1420. ACM Press, New York, NY, 2019.

[43] Stefan Droste, Thomas Jansen, and Ingo Wegener. A new framework for the valuation of algorithms for black-box-optimization. In Kenneth A. De Jong, Riccardo Poli, and Jonathan E. Rowe, editors, *Proceedings of the Seventh Workshop on Foundations of Genetic Algorithms (FOGA)*, pages 253–270. Morgan Kaufmann Publishers, 2002.

[44] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.

[45] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. On the empirical scaling behaviour of state-of-the-art local search algorithms for the euclidean TSP. In Sara Silva and Anna I. Esparcia-Alcázar, editors, *GECCO*, pages 377–384, New York, NY, 2015. ACM Press.

[46] Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In Blai Bonet and Sven Koenig, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1114–1120. AAAI Press, 2015.

[47] Agoston E. Eiben and Günther Rudolph. Theory of evolutionary algorithms: A bird's eye view. *Theoretical Computer Science*, 229(1-2):3–9, 1999.

[48] Chris Fawcett and Holger H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458, 2016.

[49] Matteo Fischetti and Michele Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.

[50] Michael Foster, Matthew Hughes, George O'Brien, Pietro S. Oliveto, James Pyle, Dirk Sudholt, and James Williams. Do sophisticated evolutionary algorithms perform better than simple ones? In Carlos A. Coello Coello, editor, *GECCO*, pages 184–192, New York, NY, 2020. ACM Press.

[51] Alberto Franzin and Thomas Stützle. Comparison of acceptance criteria in randomized local searches. In Evelyne Lutton, Pierrick Legrand, Pierre Parrend, Nicolas Monmarché, and Marc Schoenauer, editors, *EA 2017: Artificial Evolution*, volume 10764 of *LNCS*, pages 16–29, Heidelberg, 2017. Springer.

[52] Alberto Franzin and Thomas Stützle. Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104:191–206, 2019.

[53] Alberto Franzin and Thomas Stützle. A causal framework for understanding optimisation algorithms. In Fredrik Heintz, Michela Milano, and Barry O'Sullivan, editors, *Trustworthy AI – Integrating Learning, Optimization and Reasoning. TAILOR 2020*, volume 12641 of *LNCS*, pages 140–145. Springer, Cham, Switzerland, 2021.

[54] Alberto Franzin and Thomas Stützle. A landscape-based analysis of fixed temperature and simulated annealing. *European Journal of Operational Research*, In press, 2022.

[55] Walter J. Gutjahr. A Graph-based Ant System and its convergence. *Future Generation Computer Systems*, 16(8):873–888, 2000.

[56] Walter J. Gutjahr. ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, 82(3):145–153, 2002.

[57] Bruce Hajek and Galen Sasaki. Simulated annealing–to cool or not. *System & Control Letters*, 12(5):443–447, 1989.

[58] Jin-Kao Hao and Jêrome Pannier. Simulated annealing and tabu search for constraint solving. In Martin C. Golumbic et al., editors, *Fifth International Symposium on Artificial Intelligence and Mathematics, AIM 1998, Fort Lauderdale, Florida, USA, January 4-6, 1998*, pages 1–15, 1998.

[59] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[60] Giles Hooker. Generalized functional ANOVA diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2012.

[61] John N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994.

[62] John N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, 1996.

[63] Holger H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, February 2012.

[64] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search—Foundations and Applications.* Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[65] Holger H. Hoos and Thomas Stützle. On the empirical scaling of run-time for finding optimal solutions to the traveling salesman problem. *European Journal of Operational Research*, 238(1):87–94, 2014.

[66] Jérémie Humeau, Arnaud Liefooghe, El-Ghazali Talbi, and Sébastien Verel. ParadisEO-MO: From fitness landscape analysis to efficient local search algorithms. *Journal of Heuristics*, 19(6):881–915, June 2013.

[67] Mohamed Saifullah Hussin and Thomas Stützle. Tabu search vs. simulated annealing for solving large quadratic assignment instances. *Computers & Operations Research*, 43:286–291, 2014.

[68] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *LNCS*, pages 507–523. Springer, Heidelberg, 2011.

[69] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Identifying key algorithm parameters and instance features using forward selection. In Panos M. Pardalos and G. Nicosia, editors, *Learning and Intelligent Optimization, 7th International Conference, LION 7*, volume 7997 of *LNCS*, pages 364–381. Springer, Heidelberg, 2013.

[70] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*, volume 32, pages 754–762, 2014.

[71] Alan W. Johnson and Sheldon H. Jacobson. On the convergence of generalized hill climbing algorithms. *Discrete Applied Mathematics*, 119(1):37–57, 2002.

[72] David S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In Michael H. Goldwasser, David S. Johnson, and Catherine C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 215–250. American Mathematical Society, Providence, RI, 2002.

[73] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation: Part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.

[74] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation: Part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.

[75] David S. Johnson and Lyle A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[76] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC: Instance-specific algorithm configuration. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press, 2010.

[77] Artem Kaznatcheev, David A. Cohen, and Peter Jeavons. Representing fitness landscapes by valued constraints to understand the complexity of local search. *Journal of Artificial Intelligence Research*, 69:1077–1102, 2020.

[78] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, March 2019.

[79] Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation*, 27(1):99–127, 2019.

[80] Pascal Kerschke, Hao Wang, Mike Preuss, Christian Grimme, André H. Deutz, Heike Trautmann, and Michael T. M. Emmerich. Towards analyzing multimodality of continuous multiobjective landscapes. In Julia

Handl, Emma Hart, P. R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *PPSN*, volume 9921 of *LNCS*, pages 962–972. Springer, Heidelberg, 2016.

[81] Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60, 2014.

[82] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 17:1–5, 2016.

[83] Slawomir Koziel, David Echeverría Ciaurri, and Leifur Leifsson. Surrogate-based methods. In Slawomir Koziel and Xin-She Yang, editors, *Computational Optimization, Methods and Algorithms*, volume 356 of *Studies in Computational Intelligence*, pages 33–59. Springer, Berlin/ Heidelberg, 2011.

[84] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642, 2012.

[85] Arnaud Liefooghe, Bilel Derbel, Sébastien Verel, Hernán E. Aguirre, and Kiyoshi Tanaka. Towards landscape-aware automatic algorithm configuration: Preliminary experiments on neutral and rugged landscapes. In Bin Hu and Manuel López-Ibáñez, editors, *Proceedings of EvoCOP 2017 – 17th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 10197 of *LNCS*, pages 215–232. Springer, Heidelberg, 2017.

[86] Andrea Lodi and Andrea Tramontani. Performance variability in mixed-integer programming. In Huseyin Topaluglu, editor, *Theory Driven by Influential Applications*, pages 1–12. INFORMS, 2013.

[87] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[88] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Research*, 11(1–5):193–225, 1997.

[89] Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. *Computers & Operations Research*, 51:190–199, 2014.

[90] Klaus Meer. Simulated annealing versus Metropolis for a TSP instance. *Information Processing Letters*, 104(6):216–219, 2007.

[91] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günther Rudolph. Exploratory landscape analysis. In Natalio Krasnogor and Pier Luca Lanzi, editors, *GECCO*, pages 829–836. ACM Press, New York, NY, 2011.

[92] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[93] Alfonsas Misevičius and Dovilė Kuznecovaitė. Investigating some strategies for construction of initial populations in genetic algorithms. *Computational Science and Techniques*, 5(1):560–573, 2018.

[94] Alfonsas Misevičius, Dovilė Kuznecovaitė, and Jūratė Platužienė. Some further experiments with crossover operators for genetic algorithms. *Informatica*, 29(3):499–516, 2018.

[95] Debasis Mitra, Fabio Romeo, and Alberto Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. In *Decision and Control, 1985 24th IEEE Conference on*, pages 761–767. IEEE, 1985.

[96] A. Moraglio and A. Kattan. Geometric generalisation of surrogate model based optimization to combinatorial spaces. In Peter Merz and Jin-Kao Hao, editors, *EvoCOP*, volume 6622 of *LNCS*, pages 142–154. Springer, Heidelberg, 2011.

[97] A. Moraglio, Yong-Hyuk Kim, and Yourim Yoon. Geometric surrogate-based optimisation for permutation-based problems. In Natalio Krasnogor and Pier Luca Lanzi, editors, *GECCO Companion*, pages 133–134. ACM Press, New York, NY, 2011.

[98] Lucien Mousin, Marie-Eléonore Kessaci, and Clarisse Dhaenens. Exploiting promising sub-sequences of jobs to solve the no-wait flowshop scheduling problem. *Arxiv preprint arXiv:1903.09035*, 2019.

[99] Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[100] Sabrina M. Oliveira, Mohamed Saifullah Hussin, Andrea Roli, Marco Dorigo, and Thomas Stützle. Analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In *Proceedings of the 2017 Congress on Evolutionary Computation (CEC 2017)*, pages 1734–1741. IEEE Press, Piscataway, NJ, 2017.

[101] Pietro S. Oliveto, Jun He, and Xin Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007.

[102] Pietro S. Oliveto and Carsten Witt. Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science*, 605:21–41, 2015.

[103] Ibrahim H. Osman and Chris N. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6):551–557, 1989.

[104] Federico Pagnozzi and Thomas Stützle. Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European Journal of Operational Research*, 276:409–421, 2019.

[105] Federico Pagnozzi and Thomas Stützle. Evaluating the impact of grammar complexity in automatic algorithm design. *International Transactions in Operational Research*, pages 1–26, 2020.

[106] Quan-Ke Pan and Rubén Ruiz. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research*, 40(1):117–128, 2013.

[107] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[108] Lucas Marcondes Pavelski, Myriam Regattieri Delgado, and Marie-Eléonore Kessaci. Meta-learning on flowshop using fitness landscape analysis. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *GECCO*, pages 925–933, New York, NY, 2019. ACM Press.

[109] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

[110] Judea Pearl and Elias Bareinboim. Transportability of causal and statistical relations: A formal approach. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 247–254. AAAI Press, 2011.

[111] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[112] Leslie Pérez Cáceres, Bernd Bischl, and Thomas Stützle. Evaluating random forest models for irace. In Peter A. N. Bosman, editor, *GECCO Companion*, pages 1146–1153, New York, NY, 2017. ACM Press.

[113] Leslie Pérez Cáceres, Federico Pagnozzi, Alberto Franzin, and Thomas Stützle. Automatic configuration of GCC using irace. In Evelyne Lutton, Pierrick Legrand, Pierre Parrend, Nicolas Monmarché, and Marc Schoenauer, editors, *EA 2017: Artificial Evolution*, volume 10764 of *LNCS*, pages 202–216. Springer, Heidelberg, 2017.

[114] Erik Pitzer, Andreas Beham, and Michael Affenzeller. Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis. In Martin Middendorf and Christian Blum, editors, *EvoCOP*, volume 7832 of *LNCS*, pages 109–120, Heidelberg, 2013. Springer.

[115] Colin R. Reeves and A. V. Eremeev. Statistical analysis of local search landscapes. *Journal of the Operational Research Society*, 55(7):687–693, 2004.

[116] John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[117] Fabio Romeo and Alberto Sangiovanni-Vincentelli. A theoretical framework for simulated annealing. *Algorithmica*, 6(1-6):302–345, 1991.

[118] Günther Rudolph. Convergence analysis of canonical genetic algorithms. *tnn*, 5(1):96–101, 1994.

[119] Günther Rudolph. Convergence of non-elitist strategies. In Zbigniew Michalewicz, editor, *Proceedings of the First IEEE International Conference on Evolutionary Computation (ICEC'94)*, pages 63–66, Piscataway, NJ, 1994. IEEE Press.

[120] Rubén Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494, 2005.

[121] Bhupinder Singh Saini, Manuel López-Ibáñez, and Kaisa Miettinen. Automatic surrogate modelling technique selection based on features of optimization problems. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *GECCO Companion*, pages 1765–1772. ACM Press, New York, NY, 2019.

[122] Alberto Santini, Stefan Ropke, and Lars Magnus Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24:783–815, 2018.

[123] Moritz Seiler, Janina Pohl, Jakob Bossek, Pascal Kerschke, and Heike Trautmann. Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael T. M. Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature - PPSN XVI*, volume 12269 of *LNCS*, pages 48–64, Cham, Switzerland, 2020. Springer.

[124] Kate Smith-Miles and Simon Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, 2015.

[125] Kate Smith-Miles, Jano I. van Hemert, and Xin Yu Lim. Understanding TSP difficulty by learning from evolved instances. In Christian Blum and Roberto Battiti, editors, *LION*, volume 6073 of *LNCS*, pages 266–280, Heidelberg, 2010. Springer.

[126] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[127] Kenneth Sörensen, Florian Arnold, and Daniel Palhazi Cuervo. A critical analysis of the "improved clarke and wright savings algorithm". *International Transactions in Operational Research*, 26(1):54–63, 2017.

[128] Kenneth Sörensen, Marc Sevaux, and Fred Glover. A history of metaheuristics. In Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende, editors, *Handbook of Heuristics*, pages 1–27. Springer International Publishing, 2018.

[129] Thomas Stützle and Marco Dorigo. A short convergence proof for a class of ACO algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365, 2002.

[130] Thomas Stützle and Holger H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[131] Thomas Stützle and Manuel López-Ibáñez. Automated design of metaheuristic algorithms. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 541–579. Springer, 2019.

[132] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013*, pages 847–855. ACM Press, New York, NY, 2013.

[133] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer, 1998.

[134] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*, volume 37. Springer, 1987.

[135] Elia Van Wolputte, Evgeniya Korneva, and Hendrik Blockeel. MERCS: multi-directional ensembles of regression and classification trees. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4276–4283. AAAI Press, February 2018.

[136] Jean-Paul Watson, J. C. Beck, A. E. Howe, and Darrell Whitley. Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence*, 143(2):189–217, 2003.

[137] Jean-Paul Watson, Adele E Howe, and Darrell Whitley. Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. *Computers & Operations Research*, 33(9):2623–2644, 2006.

[138] Ingo Wegener. Simulated annealing beats metropolis in combinatorial optimization. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *LNCS*, pages 589–601, Heidelberg, 2005. Springer.

[139] Dennis Weyland. A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing*, 12(2):50–60, 2010.

[140] Dennis Weyland. A critical analysis of the harmony search algorithm: How not to solve Sudoku. *Operations Research Perspectives*, 2:97–105, 2015.

[141] Lin Xu, A. R. KhudaBukhsh, Holger H. Hoos, and Kevin Leyton-Brown. Quantifying the similarity of algorithm configurations. In Paola Festa, Meinolf Sellmann, and Joaquin Vanschoren, editors, *Learning and Intelligent Optimization, 10th International Conference, LION 10*, volume 10079 of *LNCS*, pages 203–217, Cham, Switzerland, 2016. Springer.