The Gestalt heuristic: learning the right level of abstraction to better search the optima

Christophe Philemotte and Hugues Bersini

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2008-021

October 2008

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/2008-021

The Gestalt heuristic: learning the right level of abstraction to better search the optima

Christophe Philemotte cphilemo@iridia.ulb.ac.be
http://iridia.ulb.ac.be/~cphilemo
Hugues Bersini bersini@ulb.ac.be
http://iridia.ulb.ac.be/bersini
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

6 October 2008

Abstract

Nowadays, engineering process often requires the optimization of a cost: improving energetic yield, minimizing the used space, reducing the development effort, . . . Not surprisingly, research in optimization is one of the most active field of computer science. Metaheuristics are among the state-of-the-art techniques for combinatorial optimization problem. In this context, the paper addresses the following question: "Considering computation time and implementation ease constraints, is it then possible to build a system increasing the efficiency of the used metaheuristic by automatically altering the problem representation during the optimization process?". Inspiring by the *Gestalt* psychology, a descriptive theory of complex cognitive processes such as vision, the paper suggests the *Gestalt* heuristic as a valid solution. The heuristic mechanism mainly consists in building a meta-model, an abstraction of the problem. This meta-model is used as a filter or a lens inserted between the metaheuristic and the problem representation. From an engineering point of view, the *Gestalt* heuristic is a promising additional mechanism for rapid prototyping and anytime optimization context. After introducing a formal and unified framework for *Gestalt* heuristic, the paper gives an illustrated guideline for the implementation of this new heuristic. The suggested implementation is then experimentally tested and discussed.

Keywords: optimization, Gestalt, metaheuristic, adaptive representation

1 Introduction

Today, computers are ubiquitous in our modern society and occupy any possible technological and societal niche. They are especially good at helping people to resolve problems and reduce the complexity of daily tasks. While once engineers believed that adequate software would optimally resolve any type of problem, today, with the increasing complexity and size of problems to be tackled and even if the P-vs-NP conjecture has not yet been proven, they admit that optimality is generally beyond any reasonable computation time. More and more engineers tend to resort to approximate approaches such as heuristics and trial-and-error processes. Heuristics are efficient software versions of rules of thumb specific to a problem. However, their development requires knowledge about the problem. Some computer scientists are more interested in saving development efforts, decreasing the need for a detailed understanding of the structure of the problem and proposing generic software mechanisms capable of solving a wider class of problem. And others in improving the efficiency with small computation time resources. In so doing, they are proposing a new kind of problem solving algorithms called metaheuristics that have become state-of-art techniques for Combinatorial Optimization Problems (COPs), especially in anytime context [31].

The most of metaheuristics need to be tuned by adjusting a set of parameters. This tuning is akin to a hand-crafted design of the algorithm for a specific problem. When these algorithms are used, this tuning turns out to be very crucial in its own. In the absence of an optimal and robust setting, any metaheuristic will rather behave such as a Formula One car riding with a tank half of super gasoline and half of inferior gasoline. Rather than a by hand tuning, computer scientists have addressed the question of a possible alternative automatic tuning. This is a very recent research question seeding new families of optimization algorithms such as reactive search [5] or racing algorithms [7]. These algorithms automatically adjust their setting during the optimization process.

Another critical issue is how to design the operators that characterize the metaheuristics. State-of-the-art algorithm design still requires experts to integrate their knowledge about a given family of problems. This expertise is then integrated in the design of operators. Here also, in order to decrease the amount of this prior cognitive load, scientists have addressed the question of systems able to automatically design or select the adequate operators for a given problem. One of the most promising ways to manage that automatization is the incorporation of machine learning techniques, especially a new family of heuristics designated as hyperheuristics [10]. These heuristics mainly explore a search space of heuristics, i.e. the operators. The goal is thus to obtain adaptive metaheuristics needing less development time and especially suited to rapid prototyping.

These recent years, a bridge is being constructed between research in metaheuristics and research in machine learning. Reactive search and hyperheuristic are examples of this cross domain fertilization. Nevertheless, keeping with this same line of thought, other crucial issues remain that still need to find interesting solutions in machine learning or other computer science domains. This paper addresses one of them. As for operators, the representation of a problem requires a particular attention. Good prior knowledge about the problem helps to build adequate representation or coding that makes easier the exploratory process of any metaheuristic. This question of the right encoding has often been addressed in the community of Evolutionary Algorithms (EAs) [27]. Some works dedicated to the adaptation of the representation for EAs exist [4, 57, 24] while for other metaheuristics, the question has never really been touched.

Practically, the adequacy between representation and operators remains a serious topic of concern even in the EA community. Just as parameters tuning or operators selection, an adequate representation of the problem can save money, energy, time and improve the understanding of the problem at hand. Considering computation time and implementation ease constraints, is it then possible to build a system increasing the efficiency of the used metaheuristic by automatically altering the problem representation during the optimization process?

We propose in this paper a new type of algorithm that positively answers this question: the *Gestalt* heuristic. Very much inspired by Gestalt psychology [2], this heuristic aims at improving online how the problem being solved needs to be represented. While Gestalt theory is rather interested in trying to explain and better characterize cognitive and perception processes (see Sec. 2), the new heuristic presented in this paper proposes a mechanism of adaptation and abstraction of the problem representation. The mechanism mainly consists in a meta-model of the inherent structure of the problem that is used as a filter or a lens to be inserted between the metaheuristic and the original representation. In such a way, the operators of the metaheuristics are altered and the sample produced by the metaheuristic is biased. The *Gestalt* heuristic can then be interpreted as a hyperheuristic. Those features of adaptation and abstraction would provide interesting advantages. From a software engineering point of view, adaptation capabilities reduce the software development effort. From a computation time resource point of view, abstraction and adaptive representation mechanisms can reduce the impact of the size or the structure hardness of a problem search space.

In Sec. 3, the heuristic is formalized in a generic framework. The framework defines three elements: an adaptive and flexible structure of representation, an operator that transforms and filters the original search space, and an automated learning system. In Sec. 4, we explain how the framework can be implemented and we detail its integration within a given metaheuristic, a simple genetic algorithm. The framework provides by itself an easy integration whereas the

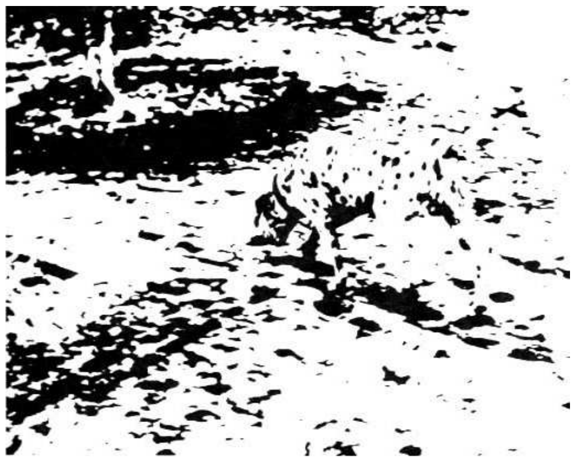

Figure 1: The *Dog Picture* is a famous picture by R.C. James for demonstrating how perception can emerge from the parts of the drawing and illustrating the maxim: "the whole is more than the sum of its parts". The dog is sniffing the ground, in the center right side. It is facing the top, left corner.

implementation does not entail a lot of change of the used metaheuristic. The *Gestalt* heuristic is just akin to a plug-in. The implementation guide is illustrated with the grouping genetic algorithm. In Sec. 5, the validation of assumptions about the *Gestalt* heuristic is experimentally made by means of the optimization of artificial and real-world problems such as hierarchical if-and-only-if problem or traveling salesman problem. The experiments results are then statistically analyzed and discussed. Section 6 discusses related and future works before getting to a general conclusion in Sec. 7.

2 The Gestalt Theory

From ecosystems to bacteria, from populations of species to living organisms such as Human, all biological systems are hierarchically organized like a Matryoshka doll. Each level of this hierarchy is a whole composed of its parts which are the only responsibles for the existence of the whole. Each of these parts consists in another whole with respect to the next lower level and so on. Each higher level provides new properties not found at the level below, giving rise to the famous maxim: "the whole is more than the sum of its parts" (see Fig 1). Beyond characterizing the so called emergent phenomena, this maxim is also supposed to ground the Gestalt theory as established by Max Wertheimer, Wolfgang Kölher and Kurt Koffka [2].

The Gestalt theory defends the idea of systemic grouping. The perception is a combination of individual sensations, in which the environment is perceived in terms of its structure, through related macroscopic properties. Max Wertheimer said [64]:

> When we are presented with a number of stimuli, we do not as a rule experience "a number" of individual things, this one and that and that. Instead larger wholes separated from and related to one another are given in experience; their arrangement and division are concrete and definite.

In psychology, it has effectively been accepted that certain shapes and configurations are favored over others in the vision process. Many theories have been proposed, the Gestalt theory being one of the most influential. Even in computational vision, this theory helps to better discriminate forms, shapes and structures in a picture (for instance see [13, 67]). The primary factors, the laws of organization that determine the grouping are: proximity, similarity, closure and simplicity. Koffka [36] has suggested that these generic laws are coordinated by the law of *Prägnanz*:

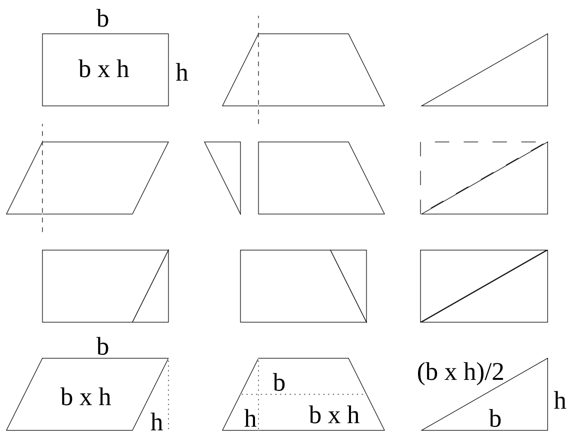

Figure 2: How does a kid figure out the area of a parallelogram, a trapezoid or a triangle, only knowing the rule to compute the rectangle area? According to Wertheimer (59), the kid can understand isomorphisms between different figures. Transforming a parallelogram into a rectangle, the kid figures out how to calculate its area. Wertheimer calls this "productive thinking".

> [. . .] of several geometrical possible organizations that one will actually occur which possesses the best, simplest and most stable shape [. . .]

The adjective "good", "simple" and "stable" are vague as claimed by Bruce, Green and Georgeson [8]:

> The physiological theory of the Gestaltists has fallen by the wayside, leaving us with a set of descriptive principles, but without a model of perceptual processing. Indeed, some of their "laws" of perceptual organisation today sound vague and inadequate. What is meant by a "good" or "simple" shape, for example?

As a matter of fact, Gestalt psychology just provides a descriptive "conceptual" reading in the context of perception and problem-solving. A rigorous mathematical theory is still far from achieved.

According to Max Wertheimer who has studied the problem-solving aspects [65], the essence of successful problem-solving behavior lies in being able to see the overall structure of the problem (see Fig. 2):

> A certain region in the field becomes crucial, is focused; but it does not become isolated. A new, deeper structural view of the situation develops, involving changes in functional meaning, the grouping, etc. of the items. Directed by what is required by the structure of a situation for a crucial region, one is led to a reasonable prediction, which like the other parts of the structure, calls for verification, direct or indirect. Two directions are involved: getting a whole consistent picture, and seeing what the structure of the whole requires for the parts.

And Wertheimer [65] states the following principles:

1. the learner should be encouraged to discover the underlying nature of a topic or problem (i.e., the relationship among the elements) ;
2. gaps, incongruities, or disturbances are an important stimulus for learning ;
3. instruction should be based upon the laws of organization: proximity, closure, similarity and simplicity (see Fig. 3).

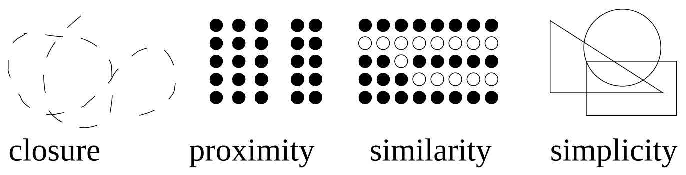

Figure 3: The laws of organizations are generic principles helping the resolution of problems or shaping the perception: closure, proximity, similarity and simplicity. The law of closure: objects grouped together are seen as a whole and discontinuities tend to be ignored. The law of proximity: spatial or temporal proximity may induce a collective classification. The law of similarity: similar items tend to be grouped together. The law of simplicity or *Prägnanz* : simplest and most stable organizations are preferred.

Despite its very qualitative nature, Gestalt theory provides strong clues to approach human perception and problem-solving. In the context of optimization, metaheuristics do not have this kind of systemic mechanism to view the search space in a more abstract way. They have a certain perception of the search space, but do not look at the tackled problem as Gestalters would suggest.

In an optimisation context, Wertheimer's principles can be very meaningful. The first rule explains the main goal of the Gestalt "perspective": an algorithm should spent computation time to discover the underlying structure of the tackled problem. The second rule stresses the key role of stochasticity during the discovering process, a notion quite common to researchers interested in metaheuristics. The last rule emphasizes the Gestalt principles and the benefits to be gained by following them. They are the guiding properties in the search of the solution and underlie this paper that indeed proposes basic algorithmic mechanisms inspired by Gestalt psychology.

3 The Gestalt principles in optimization

Most real-world COPs are at least NP-complete problems. At present, all known algorithms require computation time that is, in the best case, super-polynomial in the input size. Practically, the choice of an algorithmic solution consists in a trade-off between the run-time and the quality of the solution. Thus the design is constrained to approximate approaches such as approximation algorithms[1], heuristics[2] or metaheuristics[3]. As said in Sec. 1, metaheuristics belong to state-of-art techniques for COPs. Some metaheuristics can drive the search as an iterative guided move of one single candidate solution, a precise but very slow process in a complex search space. Other metaheuristics can sample the search space by means of a population of candidate solutions, a more exploratory approach but faster to detect the interesting zones of the search space. Each mechanism offers its own way of diversifying and/or intensifying the search and is more or less adequate depending on the size and the ruggedness of the search space. Regarding the diversification process, the efficiency of the exploration depends on the number of possible local maxima needed to be explored before being confident in the quality of the best of them.

The size, the ruggedness of the search space and the searching run-time are positively correlated. Reducing the size and the complexity of the search space in a way or another is the most efficient way to accelerate the early search. This is an important feature in anytime context. This reduction needs to be seen as a simplification of the space: either some candidate solutions could be skipped or some regions appear not to be promising enough. Nevertheless, a wrong decision and the algorithm would miss the optimum or its region. Moreover at different moments of the run-time, a candidate solution does not have the same interest for the search so that deciding to withdraw a region of the search might lead to a very unsatisfactory final outcome. The quality

[1]An approximate algorithm runs in provable polynomial time in the length and outputs a suboptimal solution.
[2]A heuristic usually, but not always, gives a good solution in a reasonable amount of time.
[3]A metaheuristic is a generic heuristic method combining black-block procedures, usually heuristics themselves.

of a decision will depend on the quality of the knowledge gained so far on the exact underlying structure of the problem. Yet, the structure of the problem is usually not known before launching the optimization process. Gestalt Theory helps us to tackle this specific difficulty.

By analogy, humans tend to perceive the world in a very similar way: some objects are perceived as indivisible entities. When looking at a car, the first object to be perceived and discriminated from the background is the car itself. The fact that a car is composed of wheels, engine, fuel tank, seats, . . . does not really count. Systemic grouping is the key: a "whole", a *gestalt*, is more informative than its parts. A *gestalt* contains the parts plus some macroscopic features. These features describe the interaction between the parts. But in optimization, what are the parts? What is a *gestalt*? The parts could be either the candidate solutions, the variables that describe the problem, or the possible values taken by one or more variables. A *gestalt* could be either a set of candidate solutions, a group of variables, or an encapsulation of several values. Once a *gestalt* is properly defined, the search space can be reduced to this new scale, the scale of *gestalt*. In such a way, the details are not lost. They are just not considered alone. Below, we will describe an algorithmic mechanism inspired by the Gestalt theory. This mechanism is a framework for applying *Gestalt* principles to any metaheuristic, i.e. to render any metaheuristic capable of a Gestalt approach to the problem.

The class of problems tackled by metaheuristics are COPs. They can be defined as a set of variables $X := \{x_1, \ldots, x_n\}$, their respective domains $\mathbb{D}_i, \forall i = 1, \ldots, n$ and possibly constraints $c_j\left(Y_j\right), Y_j \subseteq X, \forall j = 1, \ldots, m$ and an objective function $f : \bigotimes_{i=1}^{n} \mathbb{D}_i \to \mathbb{R}$ to be optimized. The variables are the main constituents of any description of a COP. A metaheuristic will consider them as individual parts. A *gestalt* variable then turns out to be a combination of variables in some given structure. At this new level of *gestalt* variables, the problem is tackled in terms of its structure, not just the simple parts that compose it. The metaheuristic considers *gestalt* variables as a whole and operates on them on account of their integrity. Now, the specific structure of a problem is usually unknown so that the *gestalt* variables cannot be optimally built. However, a parallel process could be dedicated to the discovery of these *gestalt* variables. This process could indeed discover how to shape and organise these *gestalt* variables so as to accelerate the metaheuristic run, i.e. while running, the metaheuristic can simultaneously learn which is the best way to look at the problem. This process is very remindful of the Gestalt principles which are organised around three basic ingredients:

1. a definition of a *gestalt* variable,
2. a way to look at the search space in terms of *gestalt* variables i.e. the definition of *lenses*, and
3. a learning process for evaluating any lens and building a new one on the basis of this evaluation: the *gestalt* learner.

3.1 *Gestalt* variable

Formally, a *gestalt* description is a partition of the set X, which is a set of subsets of X, whose union is X. These subsets are non-empty and pairwise disjoint. Let's define in the following *gestalt* variable and *gestalt* description:

Definition 1. Let's consider a given COP Π without constraints[4] and any instance $\pi \in \Pi$. The problem is defined as triplet $\Pi := (X, \mathcal{D}, f)$ where:

- $X := \{x_1, \ldots, x_n\}$ is a finite set of n variables,
- $\mathcal{D}$ is a function that maps each variable x_i into its domain set $\mathbb{D}_i$,
- $f : S(\pi) \to \mathbb{R}$ is an objective function to be optimized, where $\bigotimes_{i=1}^{n} \mathbb{D}_i$ is the search space $S(\pi)$ of the given instance π.

[4]Constraints do not affect the current definition.

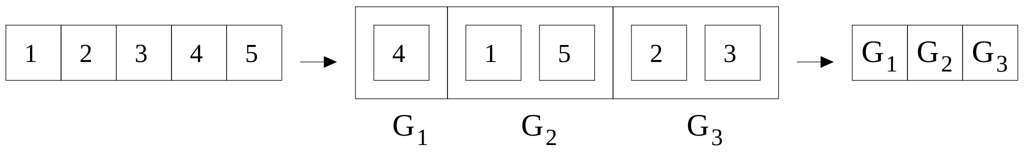

Figure 4: A problem described by five binary variables $\{x_1, x_2, x_3, x_4, x_5\}$ could be redefined on the basis of three *gestalt* variables $G_1 := \{x_4\}, G_2 := \{x_1, x_5\}, G_3 := \{x_2, x_3\}$.

A ***gestalt*** **variable** is defined by an ordered or unordered set $G := \{\ldots, y_j, \ldots\} \in 2^X$ where $y_j \in X$. Its corresponding domain is simply $\bigotimes_j \mathbb{D}_j$ where $\mathbb{D}_j$ is the domain of y_j. A problem can be described by a ***gestalt*** **description** Γ that is a set of $m \leq n$ **gestalt** variables G_k if and only if $\bigcap_k G_k = \{\}$ and $\bigcup_k G_k = X$.

Simply said, a *gestalt* variable is just a grouping of variables. This grouping gives rise to a new indivisible object: a *gestalt* variable. The problem is then described in terms of these indivisible high level *gestalt* variables. Each *gestalt* variable has a domain $\bigotimes_j \mathbb{D}_j$. This domain is the cartesian product of the domains associated with each variable that composes the *gestalt* variable. The cardinality is not modified at all. Given this new description, the search space is not reduced.

The number of partitions of a set is given by the Bell number [49]:

$$B_n = \sum_{k=0}^{n} \binom{n}{k} B_k \tag{1}$$

with $B_0 = 1$, $B_1 = 1$ and $n = |X|$ the number of elements of X. The number of *gestalt* descriptions is known. Before going further and for the sake of clarity, let's provide a simple illustration of a *gestalt* variable.

Example 1. Some COPs are described by binary variables such as the Propositional Satisfiability Problem (SAT). Let us then consider a SAT instance of five variables $\{x_1, x_2, x_3, x_4, x_5\}$ whose domain is $\{0, 1\}$ where 0 and 1, respectively, represent the false and true logic values. The search space has a size of $2^5 = 32$. It exists 52 different possible *gestalt* descriptions. A possible *gestalt* description could be $G_1 := \{x_4\}$, $G_2 := \{x_1, x_5\}$, $G_3 := \{x_2, x_3\}$. The domains of G_1, G_2 and G_3 are respectively $\{0, 1\}$, $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ (see Fig. 4). The size of the whole search space is still 32. No reduction has occurred. So what is the benefit gained by these *gestalt* variables?

The role of the *gestalt* variables is not to reduce the search space. They are objects composed of other smaller ones. They just provide a new level of description. A *gestalt* variable is first of all an encapsulation of variables, somewhat reminiscent of the object-programming paradigm where many aspects are kept private and inaccessible. If an action is applied to a *gestalt* variable G_k, it automatically turns to be applied to all variables of G_k. For instance, if we apply a binary flip to the *gestalt* variable $G_3 := \{x_2, x_3\}$, x_2 and x_3 are flipped together. As explaining in the next section, this property is essentially the result of observing the search space through a given lens.

3.2 Lens

The previous section has insisted on seeing a *gestalt* description not as a simplification of the problem. The cardinality of the search space remains unchanged. A *gestalt* variable is an indivisible object. The metaheuristic has to deal with it. The integrity could be respected by choosing an adequate representation. However if the current *gestalt* variables are not suited to the problem being tackled, the *gestalt* learner will adapt them and so will be the corresponding representation. By changing the representation, the perception of the search space is altered as a consequence of the *gestalt* description. This is indeed the role of the lens: changing the way the metaheuristic looks at the search space. But what do we mean by this "looking at the search space" ?

A metaheuristic can only operate in the search space $S(\pi)$ with respect to a neighborhood relation $N(\pi)$. This relation $N(\pi)$ depends partially on the chosen representation encoding the problem. As Hoos and Stüzle stated [31], the search process of a metaheuristic can be seen as a walk on a neighborhood graph associated with an instance π,

$$\nu(\pi) := (S(\pi), N(\pi)).$$

This neighborhood graph describes what are the possible search moves. Metaheuristics have to respect this underlying structure. Because of $N(\pi)$, departing from a given candidate solution s, the possible next candidate solutions are given as the direct neighbors $N(s)$ of s. We can thus simply say that the neighborhood relation $N(\pi)$ describes how a given metaheuristic looks at the search space.

Definition 2. Let us consider:

- a *gestalt* description Γ is defined by $m \leq n$ *gestalt* variables G_k: $\Gamma := \{G_1, \ldots, G_m\}$,
- $\mathcal{D}_\Gamma$ is a function that maps each *gestalt* variable $G_k = \{y_j\}$ into its domain set $\mathbb{D}_{G_k} = \bigotimes_j \mathbb{D}_j$,
- the search space $S_\Gamma(\pi) = \bigotimes_k \mathbb{D}_{G_k} \equiv S(\pi)$,
- a neighborhood relation on $S(\pi)$, $N(\pi) \subseteq S(\pi) \times S(\pi)$ is specified by the function $N : S(\pi) \rightarrow 2^{S(\pi)} : s \rightarrow N(s) := \{s' \in S | N(s, s')\} \subseteq S$, where the set $N(s)$ is called the neighborhood of s,

A **lens** $\mathcal{G}$ is an operator that:

- maps the neighborhood function N to the new neighborhood function N_Γ: $\mathcal{G}[N] = N_\Gamma : S_\Gamma(\pi) \rightarrow 2^{S_\Gamma(\pi)} : s_\Gamma \rightarrow N_\Gamma(s)$,
- while respecting the condition $|N_\Gamma(s)| \leq |N(s)|$.

The function N_Γ specifies the neighborhood relation $N_\Gamma(\pi) \subseteq S_\Gamma(\pi) \times S_\Gamma(\pi)$.

The lens implicitly defines a bijection from each candidate solution $s \in S$ to its *gestalt* version $s_\Gamma \in S_\Gamma$. The lens supports and produces candidate solutions as Γ describes it. A bijection exists between both sets of lenses and of *gestalt* descriptions (and their size is the same: $B_{|X|}$, see Eq. 1). The function N_Γ respects the given *gestalt* description Γ. It means that for each candidate solution $s \in S_\Gamma$, and for each $s' \in S_\Gamma$ that belongs to $N_\Gamma(s)$, one and only one step walk[5] maps s to s' without breaking the *gestalt* description Γ. So, this new relation $N_\Gamma(\pi)$ induces a new neighborhood graph $\nu_\Gamma(\pi)$ on the underlying search space $S_\Gamma(\pi)$. From a candidate solution, the possible next candidates are not the same anymore. Through the lens, the metaheuristic looks at the problem in a new way (see Fig. 5).

As said in the previous Sec. 3.1, a *gestalt* variable is not simply a grouping of variables. The grouping also affects all operations that could be applied to a part of the candidate solution: applied to a *gestalt* variable G_k, all its variables are simultaneously modified by the operation. We call this the "lens effect". The difference between the size of X and Γ is given by the size of the neighborhood set. According to Defintion 2 and considering any applied operator, the neighborhood size is thus decreased by $|N(s)| - |N_\Gamma(s)|$.

Example 2. Getting back to our previous example of SAT of five variables, let's assume a metaheuristic that uses a one-flip neighborhood. Two candidate solutions are neighbors if and only if they differ in the truth value of exactly one variable. If the five variables are encoded in a binary string, this relation neighborhood corresponds to a Hamming distance of 1. So, the graph neighborhood corresponds to a 5-bit binary hypercube graph. For instance, the candidate solutions

[5]A step walk being one stage of application of the operators of the given metaheuristics.

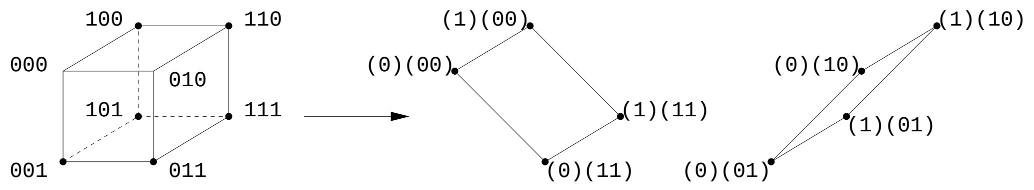

Figure 5: A problem π is encoded with three binary variables. Its neighborhood graph $\nu(\pi)$ is the binary 3-cube that is drawn on the left. The lens $\mathcal{G}$ transforms the neighborhood graph into a new one. It is the union of the two binary 2-cube that are drawn on the right.

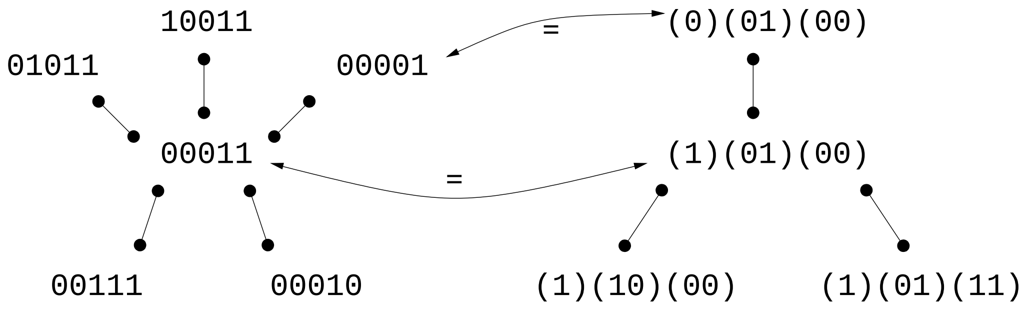

Figure 6: Let us consider a problem that is encoded with five binary variables and a possible candidate solution $(0, 0, 0, 1, 1)$. A one-bit-flip operator (a mutation in EA) gives the five neighbors that are depicted on the left. With the *gestalt* description $G_1 := \{x_4\}, G_2 := \{x_1, x_5\}, G_3 := \{x_2, x_3\}$, the candidate solution $(0, 0, 0, 1, 1) = ([1], [0, 1], [0, 0])$ only has the three possible neighbors that are depicted on the right. One neighbor is the same than one of the original neighbors. The two other neighbors are different from original ones.

$(x_1, x_2, x_3, x_4, x_5) = (0,0,0,1,1)$ and $(x_1, x_2, x_3, x_4, x_5) = (0,0,0,1,0)$ are neighbors. In the context of a given *gestalt* description, the one-flip neighborhood has to be reinterpreted. A *gestalt* variable G does not take a truth value but a l-tuple of truth values where $l = |G|$ is the number of variables that belong to the *gestalt* variable G. All variables belonging to a same *gestalt* variable are considered together. By applying the *gestalt* idea, any manipulation affecting a *gestalt* variable simultaneously affects all the variables belonging to this *gestalt* variable. A flip of one *gestalt* variable provokes a flip of l truth values. The resulting candidate will move by a Hamming distance of l. Depending on the considered *gestalt* variable, a l-exchange neighborhood is obtained. Considering the previously defined *gestalt* description $G_1 := \{x_4\}, G_2 := \{x_1, x_5\}, G_3 := \{x_2, x_3\}$, our candidate solutions $(0,0,0,1,1)$ and $(0,0,0,1,0)$ respectively become $([1],[0,1],[0,0])$ and $([1],[0,0],[0,0]$. They are not neighbors anymore because G_2 is not totally flipped. The only 3 possible neighbors of $([1],[0,1],[0,0])$ are $([1],[1,0],[0,0])$ or $([1],[0,1],[1,1])$ or $([0],[0,1],[0,0])$ (see Fig. 6). In the absence of the *gestalt* description, the neighbors has a size of 5. The condition is well respected.

The lens is thus an operator that modifies the neighborhood relation. This changes the neighborhood graph with an expected reduction in its connectivity. A lens can provide a good or a bad "perception" of the problem for the metaheuristic in charge of this problem. The goodness of a lens depends on three different factors:

1. the problem instance π,
2. the metaheuristic being used, and
3. the current search state.

The problem instance π is characterized by some properties such as the variables dependencies. These properties can be important for the metaheuristic used to find the solution. The lens induces a new representation of π. This new representation can disrupt these properties and make the search harder. Some problem instances could be not well suited to the *gestalt* approach. For instance, the needle in a haystack is the kind of deceptive problem that gains nothing by applying whatever lens. It does not show any structure by definition. The metaheuristic being used to tackle π also searches the space in a very specific way. A neighborhood relation is implicitly defined depending on the operators being applied to generate the new solutions. By applying any lens, the neighborhood relation is being affected and the obtained neighborhood relation could not match the dual neighborhood relation induced by the operators defining the metaheuristic. This specific difficulty is very remindful of the Jones' concept of "one operator, one landscape" [35]. The current search state, the last factor, is being defined by the one or more candidate solutions reached so far. The next possible candidate solutions depend both on this current search state and the neighborhood relation. Provided a given lens, some candidate solutions could be out of reach from a given search state. It is especially the case for the optimum. Without a deep study of the problem at hand and the metaheuristic being used, it is impossible to build a good lens. As a consequence, the lens has to be built online simultaneously to the search. A parallel process has to discover a good lens as a function of the problem at hand and the metaheuristic in use: this is the role of the *gestalt* learner.

3.3 *Gestalt* learner

As explained in the previous section, a learning process needs to be grafted on the algorithm: the *gestalt* learner (see Fig. 7). The learning process builds new lenses. Each lens provides an understanding of the problem in terms of a *gestalt* description, i.e. a model of the underlying structure. This learning process evaluates the quality of a lens and from the current and past evaluations, it builds a new lens. This learning process is run online together with the metaheuristic searching process.

Definition 3. Let us consider:

- the problem instance π,

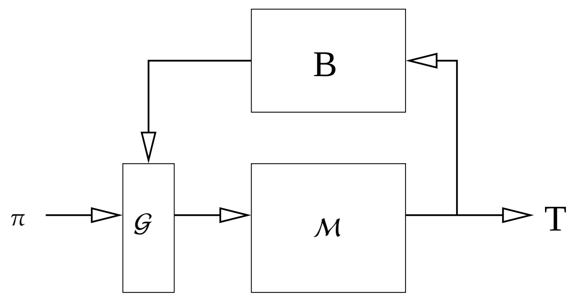

Figure 7: The block diagram of the *gestalt* learner where π is the problem instance, $\mathcal{G}$ is the lens, $\mathcal{M}$ is the metaheuristic, T is the set of candidate solutions and B is the building algorithm. By building new lens $\mathcal{G}$, it aims at maximizing the expected payoff given by the assistance quality criterion C of the lens.

- a *gestalt* description Γ and its corresponding lens $\mathcal{G}$,
- the function step $\mathcal{M}$ of the used metaheuristic with the lens $\mathcal{G}$,
- the objective function f,
- the vector $\bar{T} = (T(0), \ldots, T(t))$, where $T(0)$ is the initial set[6] of candidate solutions and $T(t)$ is the obtained set of candidate solutions after t iterations of $\mathcal{M}$, i.e. $T(t) = \mathcal{M}(T(t-1)) = \mathcal{M}^t(T(0))$.

A ***gestalt* learner** is defined by (see Fig. 7):

- the quality criterion $C : (\pi, \mathcal{G}, \mathcal{M}, \bar{T}, f(\bar{T}), t) \to \mathbb{R}$, and
- the building function $B(\mathcal{G}, C) = \mathcal{G}'$ that maximizes the expected payoff given by the quality criterion.

As explained in the previous section, the lens quality depends on different factors such as the problem instance itself. The definition of an exact objective function is thus impossible due to the lack of knowledge. A criterion of quality has to be defined based on the results obtained by the metaheuristic so far. An obvious criterion should come out as a trade-off between the computation time spent and the candidate solutions found so far. On the basis of this criterion, a learning process could be engaged to resolve the discovery of the good lenses. The *gestalt* learner aims at improving the metaheuristic actions and the results of their operators. The lenses alter these actions by changing the neighborhood graph and could then be viewed as search policies. As a result of any search policy, a certain payoff is obtained after several steps of the metaheuristics. As a matter of fact, the *gestalt* learner could also be considered as a form of reinforcement learning, a process especially meaningful in the *Gestalt* psychology.

On the basis of the measures of the lenses quality, the *gestalt* learner builds one or more new lenses. In some cases, the building process could take into account some structure properties of the problem such as metrics or symmetries. Such a design does not necessary need a lot of *a priori* knowledge. From an engineering point of view, this kind of constructive design can significantly improve the performances but is not always possible. Reinforcement learning provides ways to resolve the kind of situations in which no prior knowledge is available. Metaheuristics or stochastic model based methods are the only alternative to gradually build good lenses. The concept of the *gestalt* learner is high-level enough to be implemented in different ways.

[6]Here, we consider population based metaheuristics. The *gestalt* principles could be adapted without loss of generality to trajectory based metaheuristics. The main principles are focused on how the candidate solution is perceived by the metaheuristic, i.e. the used neighborhood graph.

The lenses must be viewed as search policies guiding the action choices made by the given metaheuristic. Before being tackled, the problem is not known and even less is its underlying structure, its fitness landscape or the position of the optima. The metaheuristic cannot have a complete view of the entire problem. The metaheuristic can only base its decision by considering the current known candidate solutions[7] and possibly the past ones. Provided that the lens quality can be defined by an objective function,on account of the definition of a *gestalt* learner, the search of an optimal lens could be viewed as a COP and will be done directly in the policy space through a general optimization method, such as EA [56, 42, 25].

4 An evolutionary implementation

In previous sections, we have provided a definition of the *Gestalt* heuristic and its mechanisms in the optimization context. These mechanisms do not form a complete metaheuristic but rather need to be seen as beneficial additions to an existing metaheuristic, a plug-in. As explained before, the *gestalt* learner is a kind of reinforcement learning that modifies online the operators heuristics they rely on.

In this section, we will describe one possible implementation of the *Gestalt* approach. This implementation is based on an EA dedicated to grouping problems and related topics such as coevolution or multi-island model. We will also explain how to plug the *Gestalt* heuristic into a given metaheuristic: here an EA, for instance the Simple Genetic Algorithm[8] (SGA). In such a way, an intertwined evolutionary processes will be derived that interacts at different levels. The basic SGA aims at finding an approximate solution to a given problem. The EA implementation of the *Gestalt* approach aims at finding a good search policy and at improving the search of the basic SGA. The complete algorithm merges two different searches: one over the problem search space and one over the lenses space. The search over the definition of the lenses is guided by the evaluations at the basic SGA level computed by using these lenses over the problem search space (a sketch of the complete algorithm can be seen in Alg. 1).

Algorithm 1 A Gestalt enhanced SGA

```
 1: initialize the lenses population
 2: initialize the solutions population
 3: while not Stop-Condition do
 4:   if some lenses are obsolete then
 5:     evaluate the lenses quality
 6:     build new lenses w.r.t. their efficiency
 7:     alter the solutions population
 8:   end if
 9:   operate one step of SGA on the solutions population
10: end while
```

The first step (see line 5 of Alg. 1) evaluates the quality of the lenses (i.e. the policies): according to the candidate solutions found after some generations of the EA. One lens is rewarded according to the way it facilitates the task of the metaheuristic. The second step (see line 6 of Alg. 1) builds new lenses on the basis of the previous ones and their respective quality. The third step (see line 7 of Alg. 1) applies the new lenses to the candidate solutions population. Finally (see line 9 of Alg. 1), through each lens, the metaheuristic perceives the search space in a completely new way and operates. The complete sequence is repeated again. The effect of the lenses is thus to make the metaheuristic operating while adopting a multi-scale perception of the search space, again very reminiscent of *Gestalt* psychology.

[7]In EA terminology, they are called individuals.

[8]Simple Genetic Algorithm is a genetic algorithm that evolves non-overlapping populations of binary encoded individuals. The operators are the one point crossover, the flip mutation and a fitness proportionate selection [39, 27].

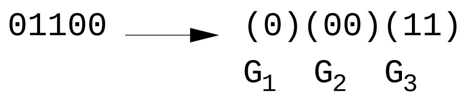

Figure 8: The figure shows the mapping between a bit string $(0, 1, 1, 0, 0)$ and its new representation $([0], [0, 0], [1, 1])$ with respect to the *gestalt* description $G_1 := \{x_4\}$, $G_2 := \{x_1, x_5\}$ and $G_3 := \{x_2, x_3\}$.

In the next subsections, an implementation guideline is given. First, it is explained the effect of a lens on the metaheuristic instance, the SGA. Secondly, the *gestalt* implementation is discussed and described. Finally, we will conclude by detailing the whole algorithm with the suggested *gestalt* implementation.

4.1 Applying lens to the SGA

4.1.1 Encoding

Until now, the definition of a problem Π did not consider the representation issues. In the COP community, this question is mostly raised in the context of EAs. The success of an EA effectively depends on the used representation due to the EA operators: mutation and crossover. EAs and other COP methods generally use a direct representation that exactly corresponds to the problem definition. The problem is then stated by implicitly considering its representation which turns out to be the world as the metaheuristic perceives it. In this case of direct representation, the methods then use problem specific operators. The *gestalt* description is then defined over the representation search space of the problem and by extension can be viewed as an adaptive representation.

Considering Example 1, a natural encoding of SAT is provided by binary strings. A candidate solution is then encoded by a binary vector $x^g = (x_1^g, \ldots, x_l^g)$ where l is the number of variable and $x_i^g \in \{0, 1\}$, as needed by the SGA. In our example, it is a SAT of five variables $(x_1, x_2, x_3, x_4, x_5) \in \{0, 1\}^5$, e.g. a vector of five bits $(0, 1, 1, 0, 0)$. Let's consider 3 *gestalt* bits $G_1 := \{x_4\}$, $G_2 := \{x_1, x_5\}$ and $G_3 := \{x_2, x_3\}$. The chromosome becomes the vector of *gestalt* bits $(G_1, G_2, G_3) \in \{0, 1\} \times \{(0, 0), (0, 1), (1, 0), (1, 1)\}^2$, e.g. $([0], [0, 0], [1, 1])$ (see Fig. 8).

4.1.2 Evaluation

The evaluation of a candidate solution through a lens is the same as without the lens. The computation cost of the bijection[9] can be neglected. The *gestalt* description of a problem does not change the way the score[10] of a candidate solution (respectively an individual) is being computed.

4.1.3 Mutation and crossover

The original operators have to respect the *gestalt* description induced by a lens. As explained in Sec. 3.2, a lens modifies the neighborhood structure exploited by the operators. This modification is due to an atomicity and functionality constraint on the *gestalt* variable, i.e. the "lens effect".

Mutation A Mutation allows slight variations over the selected candidate solution. The mutation assigns a direct neighborhood to a candidate solution (see Example 2). Through a given lens, the mutation operates a slight variation by directly manipulating *gestalt* variables. In the search space that respects a given *gestalt* description, the distance and the metric are being modified by the effect of the lens. In Example 2, we have illustrated this specific modification of the metric: a slight variation of a *gestalt* variable corresponds to a variation proportional to the size of the considered *gestalt* variable.

[9]The lens $\mathcal{G}$ is responsible for mapping the original search space $S(\pi)$ to S_Γ with a one-to-one correspondence.
[10]Or the fitness value, in EA terminology.

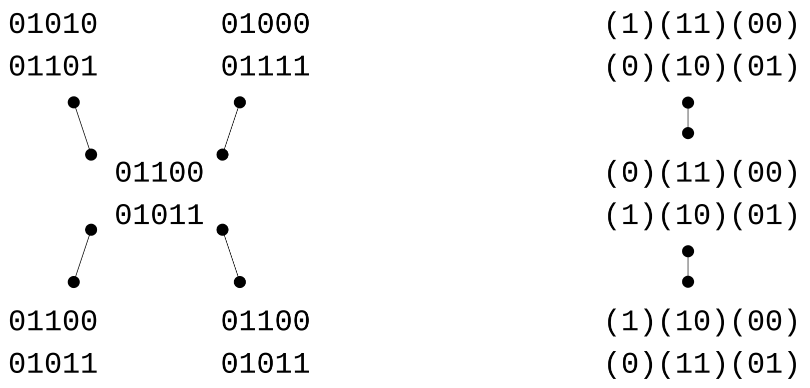

Figure 9: Let us consider a problem that is encoded with five binary variables and two possible candidate solutions $(0,1,0,1,1)$ and $(0,1,1,0,0)$. A one-point crossover can cut at four different positions and gives four possible couples of offsprings, the neighbors that are depicted on the left. With the *gestalt* description $G_1 := \{x_4\}, G_2 := \{x_1, x_5\}, G_3 := \{x_2, x_3\}$, the candidate solutions $(0,1,0,1,1) = ([0],[0,0],[1,1])$ and $(0,1,1,0,0) = ([1],[1,0],[0,1])$ have only two possible couples of offsprings as neighbors. They are depicted on the right.

The one-bit flip mutation of the SGA becomes a one-*gestalt*-variable flip mutation. The flip of a *gestalt* variable causes the flip of all its bit members. For instance, with the candidate solution $([0],[0,0],[1,1])$, the flip of the second *gestalt* variable G_2 gives the candidate solution $([0],[1,1],[1,1])$ (see Fig. 6).

Crossover Sexual reproduction is responsible for the crossover operator. It combines two candidate solutions (the parents) in order to breed one or two new offsprings. The key idea of this operator is inheritance, i.e. the transmission of some chromosomic parts from the parents to the offsprings. Through a given lens, the crossover can only exchange *gestalt* variables as a whole between selected parents.

The SGA use the one-point crossover. With a lens, it cannot cut in a *gestalt* variable since it has to respect its atomicity. The number of cut points is thus less or equal than in the absence of any lens. For instance, with the parents $([0],[0,0],[1,1])$ and $([1],[1,0],[0,1])$, any cut point can be between the second and third *gestalt* variables giving the following offsprings $([0],[0,0],[0,1])$ and $([1],[1,0],[1,1])$ (see Fig. 9).

4.2 Evolution of the lenses

As explained previously, a given lens implies a precise *gestalt* description Γ and modifies the neighborhood relation. The *gestalt* learner builds new lenses. These lenses can be represented as a grouping scheme applied on the encoding of the problem at hand. For the SGA, the alleles or bits are "perceived" through the lens as grouping into *gestalt* bits which become indivisible entities. They implicitly define possible genes - and corresponding phenotypes - that compose the chromosome. With respect to the "lens effect", the SGA cannot operate differently on the bits or on the *gestalt* bits. As illustrated in Example 2 in Sec. 3.2, this "lens effect" is guaranteed by the atomicity of *gestalt* bits: they are handled together. The atomicity implies a transformation of the neighborhood graph. So, the description of the lens can be effectively considered as a grouping scheme. Finding the best lens in a given context can be stated as a grouping problem. One possible EA implementation of the *gestalt* learner in this case turns out to be the Grouping Genetic Algorithm (GGA) [46].

The GGA was originally proposed by Falkenauer [18]. By an appropriate encoding and adequate genetic operators, it was designed to extend the range of Genetic Algorithm (GA)

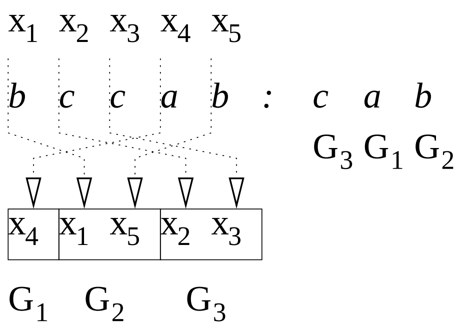

Figure 10: The figure schematically shows how the grouping encoding of a lens defines a *gestalt* description and maps each variable to its corresponding *gestalt* variable.

applicability to grouping applications. Originally, the grouping representation proposed by Falkenauer was composed of two parts: an object part that defines the membership of each object, and a group part that defines the different group labels. The group part is an addition required by the GGA crossover.

As previously explained in Sec. 3.1, a *gestalt* description is formally a set partition: the specific mapping of indices to groups does not change the way the objects are grouped. If there are m groups, each candidate solution has $m!$ representations. Different chromosomes can represent the same solution entailing the Falkenauer's representation to suffer from degeneracy[11]. The group part also adds some redundancy. However, it is used by the crossover. In some cases, degeneracy and redundancy could be beneficial [19, 50]. Recently, several authors [58, 9, 41] have suggested other representations and associated operators with the aim to resolve the redundancy and degeneracy problems. These new works are very promising. However, the original GGA is still an efficient approach that was very successful in many applications [18, 19, 20, 15]. For these reasons, we still use the Falkenauer's original representation and original ideas.

The encoding of a lens consists of two parts: the alleles part and the *gestalt* alleles part. The alleles part is a fixed size vector of the n alleles, i.e. the n objects to be optimally grouped. For the SGA, the objects are the bits. The *gestalt* alleles part is a variable size list of the m labels of each *gestalt* variables, i.e. the m groups. Each *gestalt* allele is then labelled with the symbols that are defined in the group part. With respect to its ordinal place[12], each label is assigned to the allele defining then its membership. For instance, SAT (our Example 1) is encoded by a binary string of 5 bits (the alleles) respectively labelled 1, 2, 3, 4 and 5. The *gestalt* bits G_1, G_2 and G_3 are respectively labelled a, b and c. A lens could be defined as such $bccab : cab$, i.e. $G_1 := \{x_4\}, G_2 := \{x_1, x_5\}, G_3 := \{x_2, x_3\}$ (see Fig. 10).

4.2.1 Evolutionary Operators

The Falkenauer's GGA introduces different grouping operators specifically designed to respect the structure of a grouping problem. Let's remember here the core concept of each grouping operators: mutation, inversion and crossover [20].

Mutation Three kinds of mutation are used:

- a re-assignment of a variable to a *gestalt* variable,
- a split of a *gestalt* variable, and
- a merge of two *gestalt* variables.

[11] As defined by Radcliffe and Surry [48].
[12] In EA, it corresponds to loci.

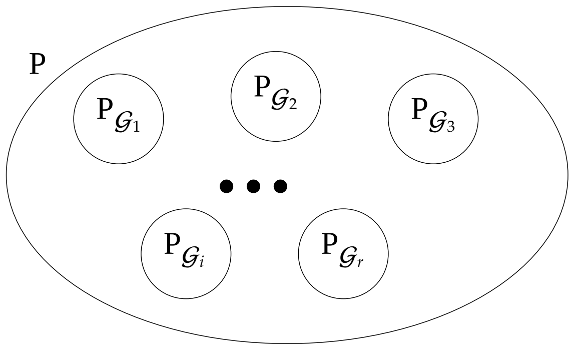

Figure 11: A population of r lenses $\mathcal{G}_i$ is considered. Each lens is associated with an island $P_{\mathcal{G}_i}$ of size ρ. All candidate solutions that belongs to a $P_{\mathcal{G}_i}$ are described and manipulated according to the lens $\mathcal{G}_i$.

Inversion With a probability P_I, the inversion only operates on the *gestalt* allele part. Two points are picked at random and between them, the order of labels are simply reversed. This operator does not change the lens definition. It is important for the crossover operator.

Crossover The crossover is a complex operator that needs several successive computation steps. The main idea is the exchange of genetic materials at the group level (in our case, *gestalt* variables). The algorithm[13] mainly consists in applying a two-points crossover to the group part of parents and exchanging objects with respect to the exchanged groups [20].

4.2.2 Population structure

As implied by Definition 3, a lens is rewarded on the basis of its assistance to the task of the SGA. The SGA is a population based metaheuristic. The lens is applied to all candidate solutions composing the population. The quality of its assistance depends on the evolution of the population. Associating different lenses to different candidate solutions makes impossible the measure of the quality of each lens. By applying a lens on a population, the population will be strongly associated to this lens. The *gestalt* learner is implemented by the GGA. A population of lenses is then evolved. A lens is necessarily associated with a population of candidate solutions. Two possible designs meet these constraints:

- lenses can be sequentially applied to a same population of candidate solutions,
- or lenses can be applied in parallel to different populations of candidate solutions.

The second design is very reminiscent of a multi-islands type of design. Multi-islands structure offers many interesting properties such as diversity, compositional evolution, or easy computation parallelization [55]. The two levels dynamics that characterizes multiple islands evolution is highly compatible with the *gestalt* philosophy while the multi-representation island model [54, 55] brings another perspective for our *Gestalt* approach. Indeed, each lens implies a modification of the original representation. Their corresponding populations are heterogeneous for the representation making each island in charge of one subpopulation.

Let us consider a population of r lenses $\mathcal{G}_i$. The candidate solutions population P is then divided into r islands $P_{\mathcal{G}_i}$, $i = 1, \ldots, r$, of fixed size ρ that are uniquely associated to each lens. Each lens is then applied to its corresponding island. Each candidate solutions of the island is being perceived through the lens, i.e. it is described with respect to the *gestalt* description that is induced by the lens (see Fig. 11).

[13]For a formal definition, see [58].

4.2.3 Evaluation

Paul Wiegand [66] defined a *coevolutionary algorithm* as

> An EA that employs a subjective internal measure for fitness assessment.

The evaluation of the lenses is of this kind. As introduced in Sec. 4.2.2, lenses are rewarded on the basis of the performance of their associated island. The evolution of lenses is a coevolution of different islands.

But what could be a good measure for the lenses quality? This quality depends on different criteria:

- fitness values of candidate solutions, i.e. a form of static quality,
- and gradients in fitness value, i.e. the quality in progress.

With a given lens, the measures based on the fitness values of the candidate solutions indicate how good the obtained results are thanks to this lens. The gradient measures a signed variation indicating how well the lens is improving the search. A good lens is one that helps to quickly discover good candidate solutions. We can define several fitness functions $g \geq 0$ of a lens $\mathcal{G}_i$ such as those based on the criterion of quality of search:

- the best found candidate solution

$$g_{best}(\mathcal{G}_i, t) = \text{best}\left(f(x) | x \in I_{i,t}\right)$$

  where best is the max (min) function value for a maximization (resp. minimization) problem and $I_{i,t}$ is the t^{th} generation of the island I_i after the application of the lens $\mathcal{G}_i$,

- the average over the island

$$g_{avg}(\mathcal{G}_i, t) = \frac{1}{\rho} \sum_{x \in I_{i,t}} f(x)$$

- the average weighted by the best (AWB)

$$g_{AWB}(\mathcal{G}_i, t) = \sqrt{g_{best}(\mathcal{G}_i)\, g_{avg}(\mathcal{G}_i)}$$

4.2.4 Selection

Our EA implementation of the *gestalt* learner uses a specific population structure with two levels of evolution: the candidate solutions and the islands attached to each lens. This two-staged selection process is compatible with the multi-level selection model [38]. The candidate solutions of a same island cooperate during the evaluation of their attached lens. This cooperation increases the probability of selection of their lens. If the lens is selected, the associated island is promoted into the global population. It has been shown that this explicit group selection increases the cooperative behavior within each subpopulation [38].

For GGA, Falkenauer suggests the use of a noisy variant of tournament selection [20]. It is obviously possible to use other selection operators. In a previous work [46], a tournament selection was used. For the SGA and with respect to respect the multi-level selection model, the stochastic universal sampling (SUS) selection has been chosen to be used at the lenses level.

4.2.5 Island operator

As we explained in Section 4.2.2, each lens is applied to its corresponding island. When lenses are evolved, new lenses are created through different operations that were previously described. New questions naturally arise. What do the islands become? Are the islands completely re-initialized? Are the islands kept without modification?

Another possible operation at the level of the islands was proposed in the Multi-Islands Model [55]: migration. Migration is an operator responsible for exchanging information between the islands. There is another operator that is based on the exchange of information: the crossover. We could say that the migration is the crossover operator but applied straightforwardly on the content of the islands. In our case, this analogy is especially salient on account of the strong link between islands and lenses: cross over lenses and bidirectionally migrate their islands.

The migration operation is characterized by the size ρ_m, the frequency t_m and the policy [55]. Skolicki concludes in his dissertation [55]:

> To take advantage of both levels [local and global evolution], inter-island diversity should be maintained by setting migration size low, interval long, topology sparse and policy weak.

The migration is implemented so as to comply with such Skolicki's statement. The topology is implied by the selection of lenses. The size ρ_m will be a small percentage of the island size ρ. The frequency t_m will be low. The specific bidirectional policy will be implemented as an instantiation of multi-level selection model philosophy. The emigrants are selected by SUS and the immigration is made by replacing random individuals but the elites.

4.3 The whole algorithm

Algorithm 2 GGA Implementation of the *Gestalt* approach

```
1:  $t = 0$
2:  initialize $\mathcal{G}_i$ population
3:  for all lenses do
4:     initialize its islands $P_{\mathcal{G}_i}$
5:  end for
6:  while not Stop-Condition do
7:     if $t_m = t$ then
8:        evaluate each $\mathcal{G}_i$
9:        copy elites in next generation
10:       select couples $(i, j)$ of parents with SUS
11:       for all selected couples of $\mathcal{G}_{i,j}$ do
12:          crossover and produce two offsprings
13:          mutations and inversion
14:          associate $P_{\mathcal{G}_{i,j}}$ to offspring
15:          migrate solutions between $P_{\mathcal{G}_{i,j}}$
16:       end for
17:    end if
18:    operate a step of EA on each $P_{\mathcal{G}_i}$
19:    $t = t + 1$
20: end while
```

In the previous sections, we have defended and described the implementation of the *Gestalt* approach by applying the GGA as an instance of EA, and its application to a specific kind of population based metaheuristic, again EAs. The whole algorithm is summarized in Alg. 2.

5 Experiments

The paper addresses the general question of possible online improvement and adaptation of a generic metaheuristic with some constraints. The suggested and discussed solution has been designed as the *gestalt* approach. Building by conceptual analogy with the *gestalt* psychology, it is defined as a general framework to improve any type of problem solving algorithm. As one possible implementation of this framework, we have illustrated how to implement the approach and how to transfer it to a given metaheuristic by making use of the Falkenauer's GGA superposed on an elementary evolutionary algorithms such as the SGA.

This section will experimentally show how the *gestalt* approach is a promising addition to any lower level problem solving algorithms. More specifically, it will practically answer the following question:

> Can an algorithm improve a generic metaheuristic online in an anytime optimization and rapid prototyping context?

The *gestalt* approach implies an alteration of the operators by adapting online the meaning of the variables coding the problem. The adaptation is made through a lens. As explained previously, a lens is an operator that maps the search neighborhood graph to another one. This new neighborhood graph is the new sandbox of the metaheuristic operators. The metaheuristic core idea is kept and the implementation modifications are minimal. The variables need to be redefined in a *gestalt* fashion. The lens transformation consists mainly in a bijective mapping between variables and *gestalt* variables. The constraints of minimal modifications and online operation is then respected. The question can be then stated as a comparison of performance:

Question 1. Following a same computation time cost, does a generic metaheuristic provide a better solution with the *gestalt* learner than without?

This is a very classic question in an optimization context. Experiments should be designed to compare a metaheuristic with and without the *gestalt* learner "plug-in". Departing from this main question about performance comparison, we can then ask the following subquestions:

Question 2. Is GGA an efficient implementation for the optimization of lenses?

Question 3. Is there an optimal lens that improves the performance, especially for deceptive problems or large instances?

Question 4. Are specific problem operators in conflict with a *gestalt* learner?

The last subquestion is a particularly interesting one. If a metaheuristic uses an operator that is very specific to a problem, what could be the role of a *gestalt* learner? The operator is designed with prior knowledge of the problem: it implicitly defines a neighborhood graph with the true structure of the problem. A *gestalt* learner mechanism has no a priori about the problem instance, the metaheuristics and their operators. It does not know that these operators are very likely to be specific to the problem. In an early phase, the *gestalt* learner might probably build lenses that are based on structure in contradiction with the one assumed by the operators tuned to the problem. This subquestion may argue the role of *gestalt* learner in a contradictory way.

Before tuning, designing and precisely describing the experiments, we will define several components such as the problem instances, the used metaheuristics, their parameter settings, the constraints and the efficiency measures. For each experiment, the collected results will then be described, statically analyzed and discussed. Finally, we will conclude the experimental study.

5.1 Experiment components

5.1.1 Problem instances

OneMax The OneMax problem (OM) [53] is a simple binary problem consisting in maximizing the number of ones composing the bit strings. Formally, this problem can be described as finding a string $x = (x_1, \ldots, x_N)$ with $x_i \in \{0, 1\}$ that maximizes the following equation:

$$f_{\mathrm{OM}}(x) = \sum_{i=1}^{N} x_i \, .$$

The optimum solution is a string with N ones getting a score of N. We will only use the instance of sizes $N = 1000$ and 2000 of OM.

Royal Road Challenge The Royal Road Challenge (RRC) is a difficult problem presented by Holland at ICGA'93 [30, 12, 34]. RRC was designed to test the Building Block Hypothesis [29, 27]. RRC shows a hierarchical structure and is typically described as a highly deceptive binary problem. Formally, the problem can be described as finding a string $x = (x_1, \dots, x_N)$ with $x_i \in \{0, 1\}$ that maximizes the following equation:

$$f_{\mathrm{RRC}}(x) = \sum_{j=1}^{k+1} B(j) + \sum_{i=1}^{1+2^k} P(i) ,$$

where $j = 1, \dots, k+1$ indexes the levels in the hierarchy, i indexes the target schemata, B the bonus function and P the part function. There are $n(j) = 2^{k-j+1}$ target schemata at level j (compounded of adjacent pairs of schemata from the next lower level). Each target schema is defined over b loci and at a distance of g from adjacent schemata. The string length N is then equal to $(b + g)2^k$. The bonus function B gives bonuses to correct blocks in x at each level:

$$B(j) = \begin{cases} u^* + (n(j) - 1)u \text{ , if } n(j) > 0 \\ 0 \text{ , if } n(j) = 0, \end{cases}$$

where u^* is the bonus for the first target at each level, and u is the bonus for the remaining targets. The part function P is the contribution to the overall score from $m(i)$ correct alleles[14] in target schema i at the lowest level:

$$P(i) = \begin{cases} m(i)v \text{ , if } m(i) < m^* + 1 \\ -(m(i) - m^*)v \text{ , if } m* < m(i) < b \\ 0 \text{ , otherwise,} \end{cases}$$

where m^* is the minimal required bits to get reward, and v is the reward/penalty by bits. We will use the original Holland's parameter setting: $N = 240$, $b = 8$, $k = 4$, $g = 7$, $u^* = 1$, $u = 0.3$, $v = 0.02$ and $m^* = 4$. The maximum is at 12.8.

Hierarchical problems This is a family of optimization problems proposed by Watson and Pollack [63], which is hierarchically decomposable and possesses non-separable building-blocks. It is called Hierarchical IF-and-only-iF (HIFF) and is a canonical version of a specific class of deceptive problems modelling the interdependency between building blocks [22, 61, 14]. In the line of HIFF, there are some other versions such as the shuffled HIFF (SHIFF) [63] that are unbiased in terms of genetic linkage order, or the HIFF2 [60] in terms of optima complementarity. Formally, all theses problems are defined on bit strings $x = (x_1, \dots, x_N)$ where $x_i \in \{0, 1\}$, the length of x is $N = k^p$, k is the number of sub-blocks in a block, and p is the number of hierarchical levels. We will always use a $k = 2$ and different p to determine the size of the considered problem.

The HIFF and SHIFF problems can be characterized as finding the string x that maximizes the following recurrent function:

$$f_{\mathrm{HIFF}}(B) = \begin{cases} 1 & \text{if } |B| = 1 \\ |B| + f_{\mathrm{HIFF}}(B_L) + f_{\mathrm{HIFF}}(B_R) & \text{if } |B| > 1 \\ & \text{and } \forall i \; b_i = 0 \text{ or } b_i = 1 \\ f_{\mathrm{HIFF}}(B_L) + f_{\mathrm{HIFF}}(B_R) & \text{otherwise,} \end{cases}$$

where B is a block of bits $\{b_1, \dots, b_n\}$, $|B|$ is the size of the block B i.e. n, b_i is the i^{th} element of B, $B_L = \{b_1, \dots, b_{n/2}\}$ and $B_R = \{b_{1+n/2}, \dots, b_n\}$ are the left and right halves of B. There are two global optima that are strings of ones or zeros. For SHIFF, the position of bits in x is randomly reordered.

The HIFF2 problem can be described as finding the string x that maximizes the following equation:

$$f_{\mathrm{HIFF2}}(x) = f_{\mathrm{HIFF}}(x \oplus g_1) + f_{\mathrm{HIFF}}(x \oplus g_2) ,$$

[14] A correct allele is commonly a one and $m(i)$ is simply the sum of bits in schema i.

where g_1 and g_2 are different bit strings of length N that are randomly generated.

We will only consider one set of 100 generated SHIFF2 instances with $(p = 8, N = 256)$ and another one with $(p = 9, N = 512)$. The optima and the gene order are picked at random.

Traveling salesman problem In combinatorial optimization, the Traveling Salesman Problem (TSP) is certainly one of the most studied, popular and representative problems. It is easy to state, to comprehend and typical of many real-world applications: finding the shortest tour passing through a set of cities which have to be visited exactly once. Since it is a NP-hard problem [23], most of the efforts have been focused on the design of algorithms that can find satisfactory solutions in a decent time. The current best performing algorithms somewhat hybridize global and local search methods [32, 59, 33].

The symmetric TSP can be defined as follows: let us consider an edge-weighted, directed graph $G := (V, E, w)$, where $V := \{v_i\}$ is a set of N vertices, $E \subseteq V \times V$ is the set of directed edges and $w : E \to \mathbb{R}^+$ a symmetric function assigning each edge $e := (v, v') \in E$ to a weight $w(e) = w((v, v')) = w((v', v))$. The TSP consists in finding the optimum Hamiltonian cycle[15] $p := (u_1, \ldots, u_N, u_1)$, where $u_i \in V$, that minimizes the path weight:

$$f_{\text{TSP}}(p) = w((u_N, u_1)) + \sum_{i=1}^{N-1} w((u_i, u_{i+1}))$$

We will use either randomly generated with TSPLIB benchmark generators[16] or real-world instances of different size from TSPLIB[17]:

- 30 clustered instances (CL) of 100 towns generated with `portcgen`,
- 30 uniform instances (UN) of 100 towns generated with `portgen`, and
- 39 instances from TSPLIB (RW): `a280`, `ali535`, `ch130`, `ch150`, `d1291`, `d198`, `d493`, `d657`, `eil101`, `fl417`, `gil262`, `gr137`, `gr229`, `kroA100`, `kroA150`, `kroA200`, `kroB100`, `kroB150`, `kroB200`, `kroC100`, `kroD100`, `kroE100`, `lin105`, `lin318`, `linhp318`, `pcb442`, `pr107`, `pr124`, `pr136`, `pr144`, `pr152`, `pr226`, `pr264`, `pr299`, `pr439`, `ts225`, `tsp225`, `u159` and `u574`.

5.1.2 Metaheuristics and parameter setting

Simple Genetic Algorithm To tackle the binary type of problems, the core metaheuristic being exploited is a SGA [39, 27] with non-overlapping populations, SUS selection, One Point crossover (1PX), flip mutation. The population is initialized uniformly at random. Its parameters are the size of population N_{pop}, the probability of crossover P_X and the probability of mutation P_μ.

Ordering genetic algorithm To tackle the TSP, the metaheuristic being exploited is an ordering Genetic Algorithm (oGA) [27] with overlapping populations, tournament selection of size 2, Partially Match (PMX) or Edge Recombination (ERX) crossover, swap adjacent loci mutation and the elitist heuristic. The population is uniformly initialized at random. Its parameters are the size of population N_{pop}, the probability of crossover P_X, the probability of mutation P_μ and the number of elites N_{eli}.

***Gestalt* plug-in** As explained in Sec. 4, the *gestalt* learner is mainly implemented as a GGA with non-overlapping populations divided into multiple islands, SUS or tournament selection, grouping operators, the elitist heuristic and migration operator. Each island consists in a population that is evolved by the given metaheuristic, either SGA or oGA[18] depending on the problem at

[15] A Hamiltonian cycle is a cyclic path that visits each and every vertex exactly once except for its starting point.
[16] http://www.research.att.com/~dsj/chtsp/download.html
[17] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
[18] See Appendix A for considerations about the *gestalt* description of a permutation.

hand. The island evolution is stopped after the generation that corresponds to the frequency of migration. Its parameters are the number of islands r, the island size ρ, the probability of crossover P_X, the probability of inversion P_I, the probability of mutations $P_{R\mu}$, $P_{S\mu}$, and $P_{M\mu}$, the size of migration ρ_m and its frequency t_m, and the used fitness function (see Sec. 4.2.3). The lenses population is uniformly initialized at random with balance constraint or not, i.e. the same number of variables as the number of *gestalt* variables.

5.1.3 Constraints

The main question 1 is asked in the anytime optimization context with the maximum time being the key constraint. Commonly, the time is measured as the number of function evaluations. The *gestalt* learner implementation, may, however become very costly in real computation time. For the sake of fairness, the maximum time allowed is measured as the real computation time[19] t_{max} in seconds. This time constraint will equally be the termination criterion of the algorithms.

5.1.4 Efficiency measures

The efficiency of an algorithm is simply measured by the non-parametric statistics of the relative objective value of the best found candidate solutions over R runs: minimum, first quartile, median, third quartile and maximum. These statistics are plotted together by relying on boxplots. The relative objective value $f(x)$ is the relative error with the optimum objective value $f(x^*)$: $\Delta f(x) = \frac{|f(x)-f(x^*)|}{f(x^*)}$. With relative objective value, the optimum is always zero and all problems are like minimization problems.

5.2 Experimental design

Three experiments have been designed and performed in order to tackle the four questions enunciated in Sec. 5. Let us remember the questions:

Question 1: Following a same computation time cost, does a generic metaheuristic provide a better solution with the *gestalt* learner than without?

Question 2: Is GGA an efficient implementation for the optimization of lenses?

Question 3: Is there an optimal lens that improve the performance, especially for deceptive problems or large instances?

Question 4: Are specific problem operators in conflict with a *gestalt* learner?

The first experiment is especially designed for Question 2. The second experiment for Question 3. The third experiment for Questions 2 and 4. After describing each experiment in turn, we will conclude by explaining the method we used for tuning the settings of metaheuristics (i.e. GAs) and the *gestalt* plug-in (i.e. the GGA).

5.2.1 Experiment 1

We answer to Questions 1 and 2 by comparing the only metaheuristic and its *gestalt* enhanced version while obeying the computation time constraint $t_{max} = 150$s. We choose, as metaheuristic, the SGA for its simplicity and generality. The tackled problems are easy such as the OM ($N = 1000$), but can also be very deceptive like the SHIFF2 ($N = 256$). OM does not show any structure while SHIFF2 has a strong hierarchical structure. The SGA with and without the *gestalt* plug-in are run 100 times. For OM, each run is processed on the same instance. For SHIFF2, each run is processed on a different instance. We compare the efficiency as a function of the running time

[19]The computer is an AMD Athlon XP 2400+ 2GHz, with 1.5GB RAM. The operating system is a Gentoo GNU/Linux (kernel 2.6.23).

until t_{max} = 120s. Then, we run again all SGA versions by keeping the same settings but on double sized instances of OM (2000) and SHIFF2 (N = 512). In this way, we evaluate the effect of the instance size on the efficiency and adaptive capability of the *gestalt* plug-in.

5.2.2 Experiment 2

In order to address Question 3, a specific lens has been built based on a priori knowledge of a given problem, the RRC. A simple SGA is run 500 times with different lenses applied on the RRC instance. Each run lasts t_{max} = 10s. We measure the benefits in terms of efficiency that this lens set up a priori brings to the optimization process.

The used RRC instance[20] optimum consists in sixteen 8-ones blocks separated by 7-bits blocks. 4 different lenses are built, each leading to a specific *gestalt* description. One of the lenses gives all the details, i.e. the *gestalt* variables simply corresponds to the variables of the problem. The three others consist in grouping all useless sixteen 7-bits blocks in one *gestalt variable*, and composing the remaining *gestalt* variables of one, two or four sequential genes coming from a same useful 8-bits block.

5.2.3 Experiment 3

As for Experiment 1, a comparison is made between a simple metaheuristic and its *gestalt* enhanced version, always subject to the time constraint t_{max} = 150s. For this specific comparison, the metaheuristic is the oGA, and the tackled problem is the TSP. Different TSP instances are considered (see Sec. 5.1.1).

With respect to Question 4, the oGA can use either PMX or ERX as crossover operator. PMX, a generic crossover, has been conceived as a trade-off between the transmission of relative order, absolute order and adjacency relations present in the parent permutations to offspring permutations. PMX is more flexible but performs worse than a crossover suited to a specific problem. ERX is specifically well suited to tackle the TSP. It is one of the best known genetic crossover operators for tackling TSP. ERX considers a solution as a tour of cities and is defined over circular permutations. It aims at transmitting adjacency relations from parents to offspring.

We compare efficiency results obtained after t_{max} for oGA with ERX or PMX and with or without the *gestalt* plug-in tackling TSP instances (see Sec. 5.1.1) 10 times.

5.2.4 Tuning of GA settings

The GAs parameters set has been set up by exploiting Birattari's Race algorithm[21] [7]. Table 1 and 2 summarize tested configurations and best configuration of GAs for each problem. After well tuning each metaheuristic for each problem, the *gestalt* plug-in itself is set up with the same Race algorithm. Tables 3 and 4 summarize the tested configurations and best configuration of *gestalt* plug-in for each problem (except RRC) and each GA.

5.3 Results and discussions

Here, we describe and discuss the results from experiments 1, 2 and 3 (see Sec. 5.2.1, 5.2.2 and 5.2.3) and give statistics about them. The discussion of results is done in next Sec. 5.4.

[20]We remember its parameter setting: $N = 240$, $b = 8$, $k = 4$, $g = 7$, $u^* = 1$, $u = 0.3$, $v = 0.02$ and $m^* = 4$.

[21]This racing algorithm tests different configurations, i.e. parameter sets, over different instances. For each instance, the given metaheuristic is run for each configuration. From the obtained results, the configurations are ranked with a Friedman test. Considering a given confidence level, the Friedman test gives the significantly bad configurations (i.e. with a corresponding p-value). These significantly worse configurations are deleted from the configurations pool and not considered in the next steps of the racing algorithm. An R package of Race is available on `http://cran.r-project.org/src/contrib/Descriptions/race.html`

oGA with 10% of N_{pop} as elites					
with	Instance (Nbr.)	N_{pop}	P_X	P_μ	t_{max} s
ERX	TSP CL (30)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300
	TSP UN (30)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300
	TSP RW (39)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300
PMX	TSP CL (30)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300
	TSP UN (30)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300
	TSP RW (39)	$\{500, 1000, 1500, 2000\}$	$\{1\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$	300

SGA					
with	Instance (Nbr./size)	N_{pop}	P_X	P_μ	t_{max} s
1PX	OM (1/1000)	$\{10, 50, 100, 500, 1000\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$	$\{0, 0.001, 0.01, 0.1\}$	60
	RRC (1/240)	$\{100, 250, 500, 750, 1000\}$	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$	$\{0, \frac{1}{480}, \frac{1}{240}, \frac{1}{120}, \frac{1}{60}\}$	60
	SHIFF2 (100/256)	$\{200, 250, 300, 350, 400\ 500, 750, 1000, 2000, 4000\}$	$\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6\ 0.7, 0.8, 0.9, 1.0\}$	$\{0, \frac{1}{2048}, \frac{1}{1024}, \frac{1}{512}, \frac{1}{256}\ \frac{1}{128}, \frac{1}{64}, \frac{1}{32}, \frac{1}{16}\}$	120

Table 1: Tested configurations of GAs for each problem. Theses configurations are tested with the Race algorithm.

oGA, ERX or PMX 10% of N_{pop} as elites			
Instance	N_{pop}	P_X	P_μ
TSP UN	2000	1	1
TSP CL	2000	1	1
TSP RW	2000	1	1
SGA, 1PX			
OM	100	0.5	0.001
RRC	1000	0.4	$\frac{1}{240}$
SHIFF2	250	0.15	$\frac{1}{1024}$

Table 2: Best configurations of GAs selected by the Race algorithm for each instances set.

best tuned oGA with ERX or PMX	
For each Instances set (Nbr.)	{TSP UN (30), TSP CL (30), TSP RW (39)}
Selection	tournament
Initialisation	{balanced, not balanced}
g	g_{AWB}
ρ	$\{50, 100, 200, 300\}$
r	$\{8, 10, 20, 50\}$
Elites (Nbr.)	0
P_X	1.0
P_I	1.0
$P_{R\mu}$	$\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
$P_{S\mu}$	0.0
$P_{M\mu}$	0.0
ρ_m (in % of ρ)	$\{0, 50\}$
t_m	$\{10, 20, 30, 40, 50\}$
Migration policy (emigrated-replaced)	random-random

best tuned SGA with 1PX for OM (N = 1000)	
Selection	SUS
Initialisation	not balanced
g	g_{best}
ρ	$\{20, 25, 50\}$
r	$\{5, 7, 9, 11\}$
Elites (Nbr.)	1
P_X	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
P_I	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
$P_{R\mu}$	$\{0.0, 0.001, 0.01, 0.1\}$
$P_{S\mu}$	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
$P_{M\mu}$	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
ρ_m (in % of ρ)	$\{10, 20\}$
t_m	$\{5, 7, 10, 20\}$
Migration policy (emigrated-replaced)	SUS-random

best tuned SGA with 1PX for 100 SHIFF2 (N = 256)	
Selection	SUS
Initialisation	not balanced
g	g_{best}
ρ	$\{10, 12, 25, 50\}$
r	$\{5, 10, 25, 30\}$
Elites (Nbr.)	1
P_X	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
P_I	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
$P_{R\mu}$	$\{0.0, \frac{1}{512}, \frac{1}{256}, \frac{1}{128}, \frac{1}{64}\}$
$P_{S\mu}$	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
$P_{M\mu}$	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
ρ_m (in % of ρ)	$\{10, 20, 30, 40, 50, 60\}$
t_m	$\{1, 2, 3, 4, 5, 7, 10\}$
Migration policy (emigrated-replaced)	SUS-random

Table 3: Tested configurations of *gestalt* plug-in for each best tuned GA (see Tab. 2) and problem.

GA	oGA + ERX	oGA + PMX	SGA + 1PX	
Problem	TSP UN, CL, RW		OM	SHIFF2
Selection	tournament		SUS	
Initialisation	balanced		not balanced	
g	g_{AWB}		g_{best}	
ρ	200		20	10
r	10		5	25
Elites (Nbr.)	0		1	
P_X	1.0		1.0	
P_I	1.0		0.75	0.5
$P_{R\mu}$	1.0		0.001	$\frac{1}{128}$
$P_{S\mu}$	0.0		0.25	1.0
$P_{M\mu}$	0.0		0.5	0.25
ρ_m (in % of ρ)	50		10	20
t_m	50	20	5	2
Migration policy (emigrated-replaced)	random-random		SUS-random	

Table 4: Best configurations selected by the Race algorithm for each GA kind and instances problem.

5.3.1 Experiment 1[22]

Figure 12 compares efficiency of SGA without and with *gestalt* plug-in as a function of computation time $t \leq 150$s for OM instances of size $N = 1000$ and 2000 (see Sec. 5.1.1). The sample size of boxplots is 100.

For OM instance of size 1000, the SGA with *gestalt* plug-in gives always a better median and converges faster than the SGA without the *gestalt* plug-in. For instance, at $t = 150$s, the median of SGA efficiency is improved by about 1.6% of the optimum value (Mann-Whitney U test, p-value $7.486\,10^{-12}$). The SGA with *gestalt* plug-in reaches 35% of the optimum value with a median computation time of 50s. The SGA after 67.5s.

For OM instance of size 2000, the SGA with *gestalt* plug-in gives always a better median and converges faster than without. For instance, at $t = 150$s, the median of SGA efficiency is improved by about 3.5% of the optimum value (Mann-Whitney U test, p-value $< 2.2\,10{-}16$). The SGA with *gestalt* plug-in reaches 40% of the optimum value with a median computation time of 30s. The SGA after 105s.

Figure 13 compares efficiency of SGA without and with *gestalt* plug-in as a function of computation time $t \leq 120$s for SHIFF2 instances of size $N = 256$ and 512 (see Sec. 5.1.1). The sample size of boxplots is 100.

For SHIFF2 instances of size 256, until $t = 30$s, the SGA with *gestalt* plug-in gives a better median than without (at $t = 30s$, Mann-Whitney U test, p-value 0.007324, the statistical significance decreases with time). The SGA with *gestalt* plug-in reaches 65% of the optimum value with a median computation time of 15s. The SGA after 20s. After $t = 40$s, the only SGA becomes better than with *gestalt* plug-in (at $t = 40$s, Mann-Whitney U test, p-value 0.006434). At $t = 120$s, the median of SGA efficiency is worsened by about 4.4% of the optimum value (Mann-Whitney U test, p-value $< 2.2\,10^{-16}$). The only SGA reaches 55% of the optimum value with a median computation time of 70s. The SGA with *gestalt* plug-in with 90s.

For SHIFF2 instances of size 512, the SGA with *gestalt* plug-in gives always a better median and converges faster than without. For instance, at $t = 120$s, the median of SGA efficiency is improved by about 2.2% of the optimum value (Mann-Whitney U test, p-value $< 2.2\,10^{-16}$). The SGA with *gestalt* plug-in reaches 70% of the optimum value with a median computation time of 30s. The only SGA with 60s.

Experiment 1 is designed for Question 2 and focused on the study of the EA implementation of *gestalt* plug-in. The efficiency of its assistance is analyzed as a function of the computation

[22]See Sec. 5.2.1.

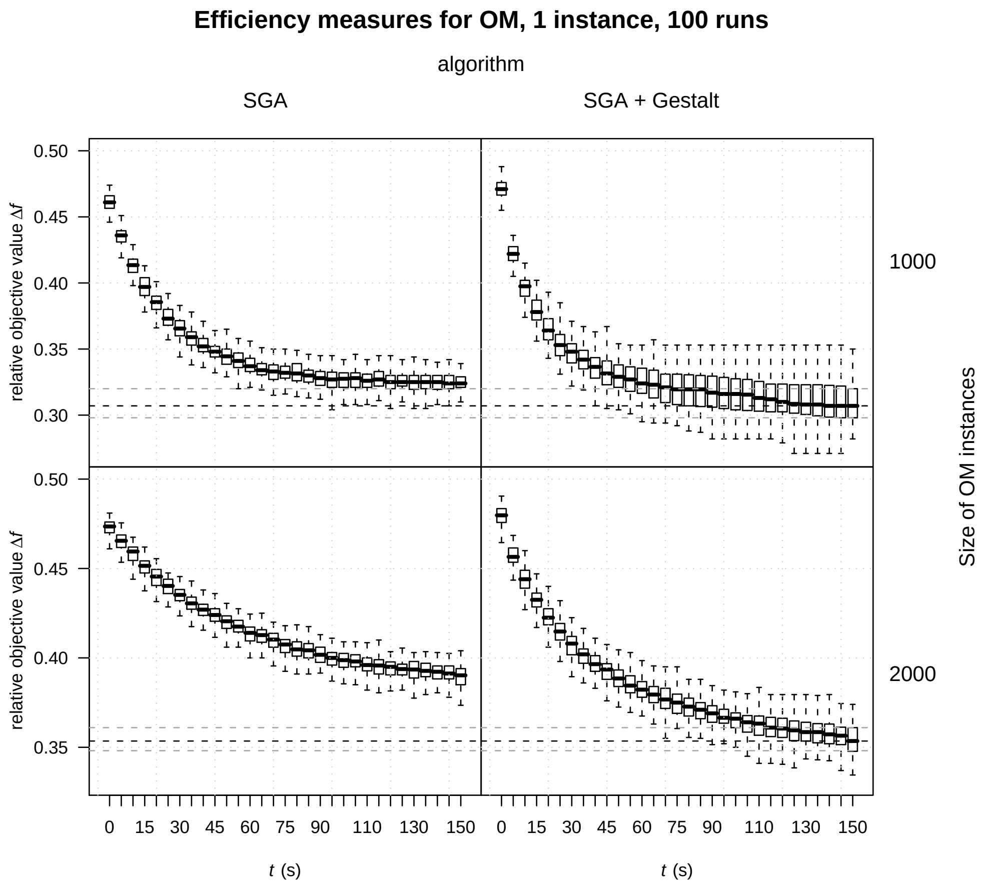

Figure 12: Experiment 1 (see Sec. 5.2.2). The efficiency of SGA is compared with its *gestalt* enhanced version..Two different size of OM instances are considered: $N = 1000$ bits (plotted at the top) and 2000 bits (plotted at the bottom). For each size, one instance is tackled one hundred times by each algorithm version. The relative objective value Δf, i.e. the measure of efficiency, is plotted on the y axis as a function of computation time t (in seconds), plotted on the x axis.

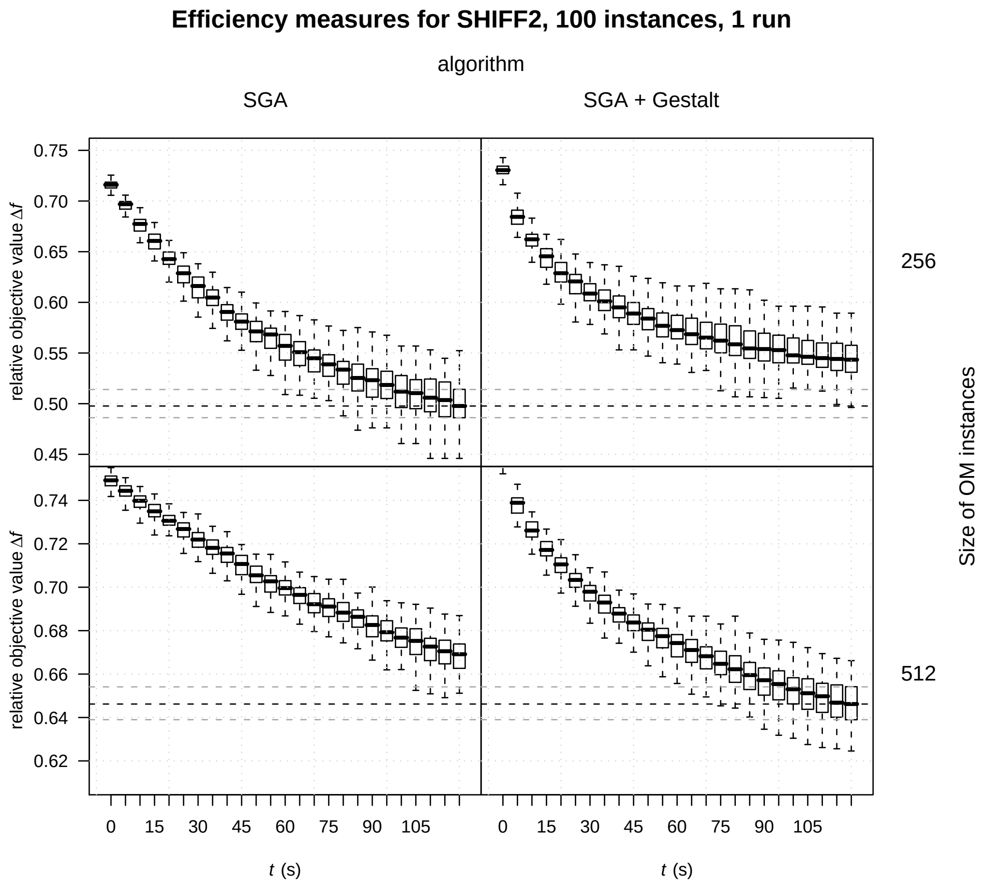

Figure 13: Experiment 1 (see Sec. 5.2.2). The efficiency of SGA is compared with its *gestalt* enhanced version..Two different size of SHIFF2 instances are considered: $N = 256$ bits (plotted at the top) and 512 bits (plotted at the bottom). For each size, one hundred different instances are tackled once time by each algorithm version. The relative objective value Δf, i.e. the measure of efficiency, is plotted on the y axis as a function of computation time t (in seconds), plotted on the x axis.

time. Two different problems, OM and SHIFF2, and two instance sizes of them are considered. The SGA and *gestalt* plug-in are well tuned for each problem and the smaller instance size. We can notice that the *gestalt* plug-in effectively improves the SGA efficiency. The improvements are in terms of quality of solution but also of time of convergence, especially in early optimisation phase. The EA implementation, especially the GGA, can not refine a structure when necessary. As explained for the Experiment 1, the probability presence of *gestalt* values in a population depends on the size of its *gestalt* variable. Sometimes, it is useful to break the structure to find new region in search space. When the instance size is doubled and despite of a specific tuning, the *gestalt* plug-in also improves the adaptation of SGA.

5.3.2 Experiment 2[23]

Figure 14 compare SGA efficiency as a function of computation time $t \leq 5$s with different lenses for RRC instance (see Sec. 5.1.1). The sample size of boxplots is 500. The four lenses describe different structure corresponding to the RRC definition. Induced *gestalt* variables group bits as a function of their membership, i.e. if the bit belongs to a 8-ones block or a 7-bits intron. The first lens ($b = 1$, $g = 1$) gives all the details, i.e. each bit of 8-ones block and 7-bits intron corresponds to one *gestalt* variable. The second lens ($b = 1$, $g = 112$) groups all bits of all sixteen introns in one *gestalt* variable and separately considers each bit of 8-ones block. The third ($b = 2$, $g = 112$) and fourth ($b = 4$, $g = 112$) lenses are like the second one but they respectively group the bits of 8-ones block by two and four.

From $t = 0$s until $t = 2$s, the lenses ($b = \{1, 2\}$, $g = 112$) are better than the first lens ($b = 1$, $g = 1$) (Mann-Whitney U test, p-values $< 2.2\,10^{-16}$). From $t = 3$s until $t = 4$s, the second lens ($b = 1$, $g = 112$) is better than the first lens ($b = 1$, $g = 1$) (Mann-Whitney U test, p-values are respectively $< 2.2\,10^{-16}$ and $3.348\,10^{-5}$. For $t = 5$s, the second lens gives a better median, first quartile and minima than the first lens. But the third quartile and the maxima are worse, and the difference between distributions are not statistically significant (Mann-Whitney U test, p-value 0.7601).

The fourth lens gives better results than the first lens for $t = 1$s (Mann-Whitney U test, p-value 0.001554). With this lens, the SGA converges very fast to a worse value than with the other lenses. This fact can be observed for the third lens after $t = 3$s. For the second and third lenses (and fourth one to a lesser extent), the increase in efficiency is the greatest for the first second of computation time.

Designed for the Question 3, Experiment 2 investigates the effect of lenses built with a priori knowledge on the efficiency of a given metaheuristic, here a SGA. According to the definition of RRC, the tackled problem, several lenses are built. We can notice the benefit of grouping the introns together in a same *gestalt* variable. When grouping bits from 8-ones block, the begin of optimization is improved. But for too big *gestalt* variables, this benefit becomes a barrier for the optimization: SGA quickly falls into local optimum. When the population is initialized, different values of *gestalt* variables are initially present or not. The probability of their presence in the population decreases as a function of the size of *gestalt* variables. Because of the *gestalt* description, the recombination can not exchange bits of a same *gestalt* variable between two different candidate solutions. In Experiment 2, a given applied lens is static along the optimization. Some regions of the search space are thus not accessible. Reaching these unaccessible regions needs to change the lens. This gives explanations for the observed early convergence and the necessity of dynamic mechanism as the *gestalt* learner.

5.3.3 Experiment 3[24]

Figures 15, 16 and 17 compare oGAs with or without *gestalt* plug-in for UN, CL and RW TSP instances (see Sec. 5.1.1). The sample sizes of boxplots are respectively 300, 300 and 390. For each TSP instance kind (UN, CL and RW), the distributions of efficiency measures are different (Kruskal-Wallis test, p-values $< 2.210^{-16}$).

[23]See Sec. 5.2.2.
[24]See Sec. 5.2.3.

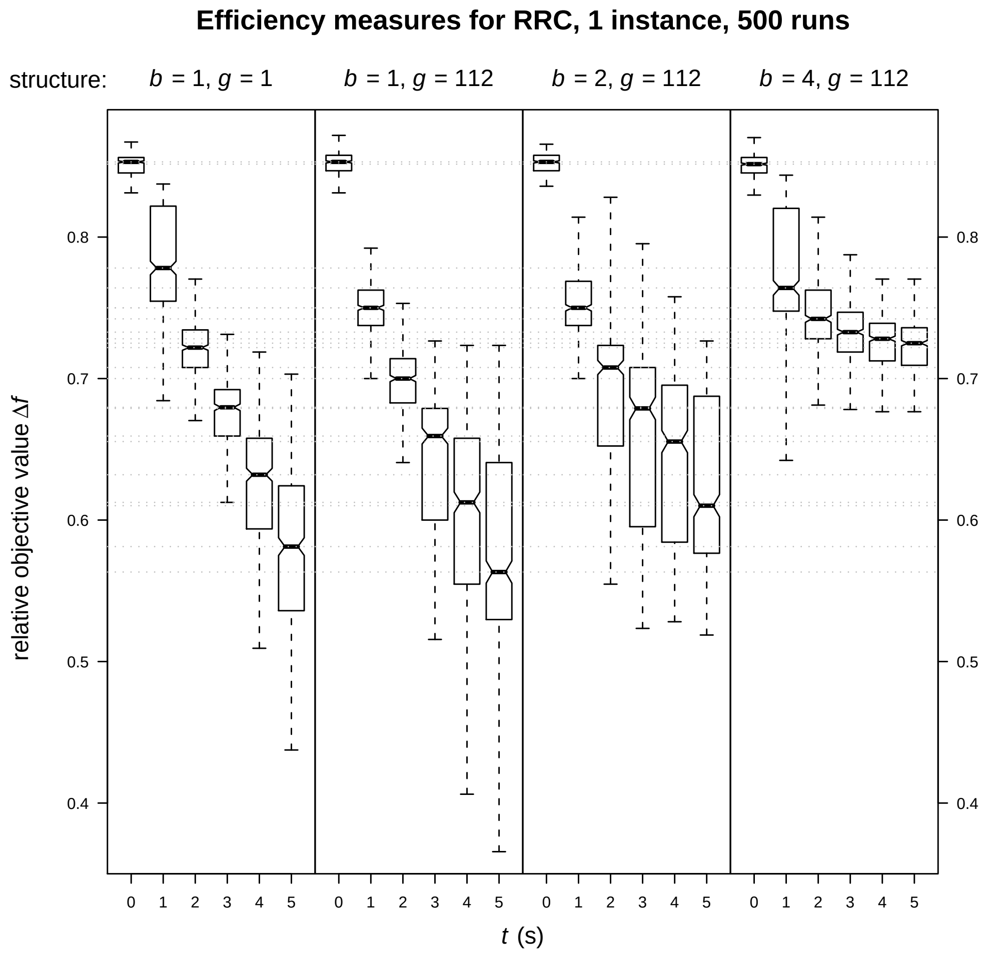

Figure 14: Experiment 2 (see Sec. 5.2.2). An SGA is run with four different lenses. From left to right. First lens ($b = 1$, $g = 1$) gives all details. The three next lenses ($g = 112$) consist in grouping all useless sixteen 7-bits introns in one *gestalt* variable and considering *gestalt* variables of one ($b = 1$), two ($b = 2$) or four ($b = 4$) sequential genes coming from a same 8-ones block. We compare the assistance of each lens in term of efficiency, i.e. the relative objective value Δf, as a function of time ($t \leq 5$s). The SGA is run 500 times for each lens.

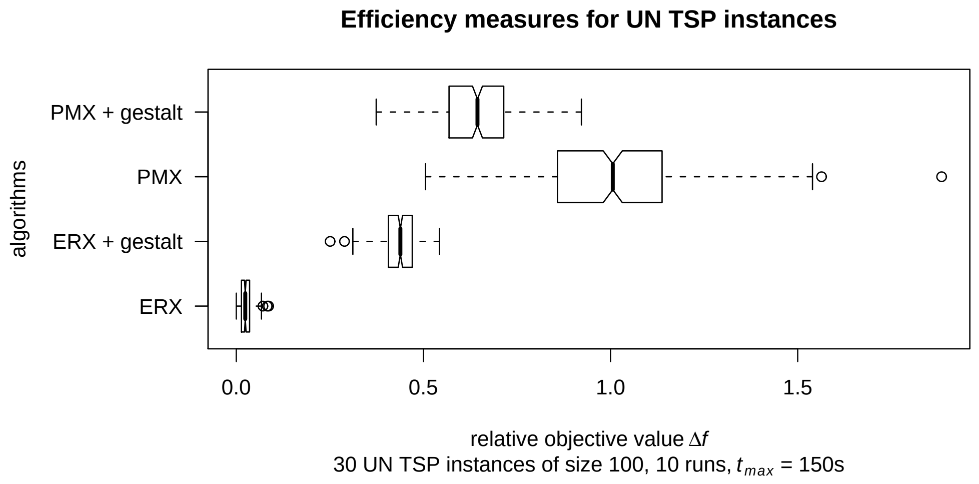

Figure 15: Experiment 3 (see Sec. 5.2.3. The oGA uses either PMX or ERX. Their efficiency is compared with their *gestalt* enhanced version. 30 different UN TSP instances are tackled 10 times by each algorithm version. The efficiency is measured at $t_{max} = 150$s and plotted on the x axis as relative objective value Δf.

The oGA with PMX gives always the worst median. With the *gestalt* plug-in, the efficiency median of oGA is improved by about 36% of the optimum value for UN, 15% for CL and 118% for RW (Mann-Whitney U test, p-values $< 2.2\,10^{-16}$). The range of the distribution is decreased by about 49% of the optimum value for UN, 29% for CL and 360% for RW.

The oGA with ERX gives always the best median. With the *gestalt* plug-in, the efficiency median is worsened by about 41% of the optimum value for UN, 35% for CL and 48% for RW (Mann-Whitney U test, p-values $< 2.2\,10^{-16}$). The range of the distribution for the ERX is increased by about 15% of the optimum value for the 3 instance cases with the *gestalt* plug-in.

Figures 18 and 19 compare oGAs with or without *gestalt* plug-in for instances of size less than 400 and greater than 400 towns. The sample sizes of boxplots are respectively 940 and 50. For each size categories, the distributions of efficiency measures are different (Kruskal-Wallis test, p-values are respectively $< 2.210^{-16}$ and $9.425\,10^{-10}$).

The oGA with PMX gives always the worst median. With the *gestalt* plug-in, the efficiency median of oGA is improved by about 30% of the optimum value for size less than 400 and 488% for size greater than 400 (Mann-Whitney U test, p-values $< 2.2\,10^{-16}$). The range of the distribution is decreased by about 132% of the optimum value for size less than 400 and 467% for size greater than 400.

The oGA with ERX gives the best median for size less than 400. With the *gestalt* plug-in, the efficiency median is worsened by about 40% of the optimum value (Mann-Whitney U test, p-value $< 2.2\,10^{-16}$). The range of the distribution of the ERX is increased by about 42% of the optimum value with the *gestalt* plug-in.

The oGA with ERX and *gestalt* plug-in gives the best median for size greater than 400. The efficiency median of the ERX is improved by about 35% of the optimum value (Mann-Whitney U test, p-value 0.01861). The interquartile range of the distribution for ERX is decreased by about 65% of the optimum value with the *gestalt* plug-in. However, the range of the distribution for the ERX is increased by about 34% of the optimum value with the *gestalt* plug-in.

Questions 2 and 4 have guided the design of Experiment 3. Another GA is considered: oGA with PMX or ERX. The tackled problem is TSP. Several kinds and sizes of TSP instances are considered (UN, CL, RW). The efficiency of an algorithm is measured at a given computation time. Added to oGA with PMX, *gestalt* plug-in always improves the original efficiency. PMX is

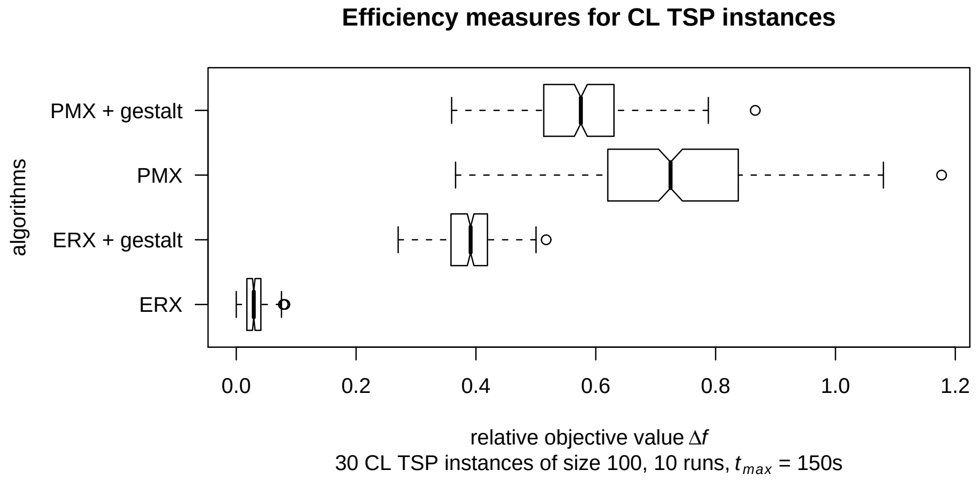

Figure 16: Experiment 3 (see Sec. 5.2.3. The oGA uses either PMX or ERX. Their efficiency is compared with their *gestalt* enhanced version. 30 different CL TSP instances are tackled 10 times by each algorithm version. The efficiency is measured at t_{max} = 150s and plotted on the x axis as relative objective value Δf.

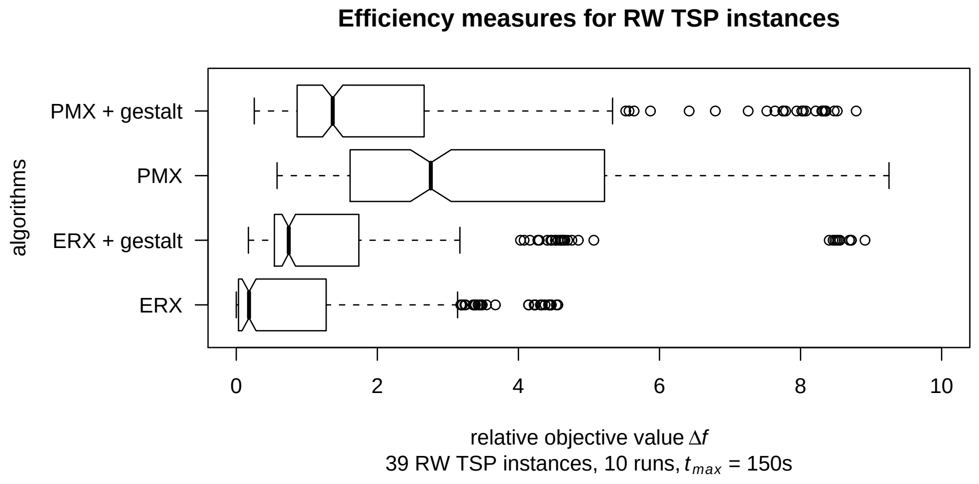

Figure 17: Experiment 3 (see Sec. 5.2.3. The oGA uses either PMX or ERX. Their efficiency is compared with their *gestalt* enhanced version. 39 different RW TSP instances are tackled 10 times by each algorithm version. The efficiency is measured at t_{max} = 150s and plotted on the x axis as relative objective value Δf.

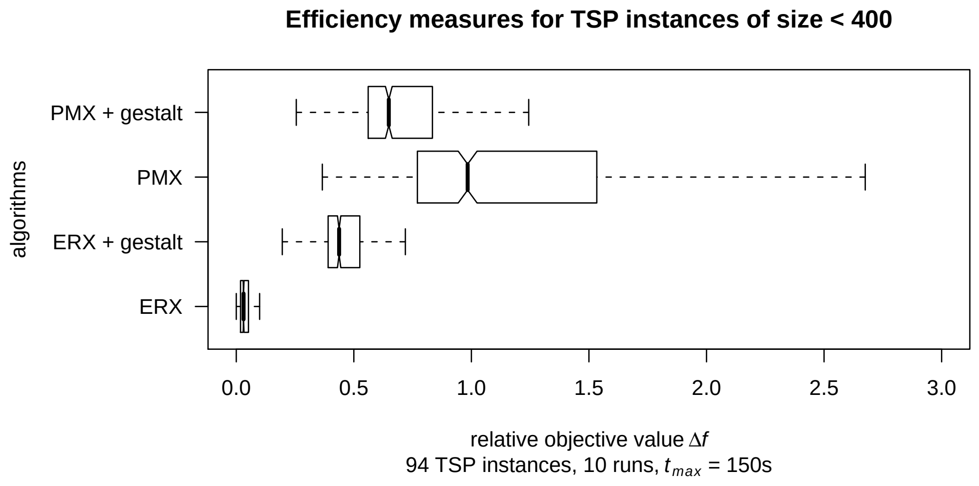

Figure 18: Experiment 3 (see Sec. 5.2.3. The oGA uses either PMX or ERX. Their efficiency is compared with their *gestalt* enhanced version. 94 different CL TSP instances of size less than 400 are tackled 10 times by each algorithm version. The efficiency is measured at t_{max} = 150s and plotted on the x axis as relative objective value Δf.

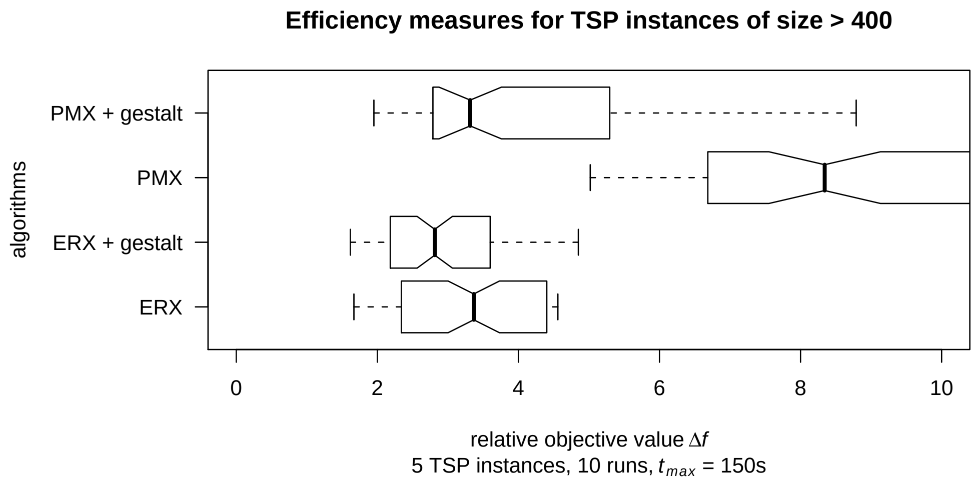

Figure 19: Experiment 3 (see Sec. 5.2.3. The oGA uses either PMX or ERX. Their efficiency is compared with their *gestalt* enhanced version. 5 different TSP instances of size greater than 400 are tackled 10 times by each algorithm version. The efficiency is measured at t_{max} = 150s and plotted on the x axis as relative objective value Δf.

designed to be a trade-off between the transmission of relative order, absolute order and adjacency relations present in the parent permutations to offspring permutations. It is a generic operator, and does not imply strong a priori structure. For small instances, oGA with ERX is still the best. ERX considers a solution as a tour of cities and is defined over circular permutations. It aims at transmitting adjacency relations from parents to offspring. Adding *gestalt* plug-in worsens the original efficiency of oGA with ERX. ERX effectively implies a strong a priori structure based on locality. With *gestalt* plug-in, it has to operates at higher levels. There is a conflict between the representation abstraction and the operator. However, considering the instance size, the *gestalt* plug-in also improves oGA with ERX to handle huge search space.

5.4 Conclusion

As conclusion, lenses can improve the efficiency of a metaheuristic, especially if they are adapted online. The EA implementation nearly succeeds to implement the *gestalt* learner. But GGA can not really refine the structure when necessary. The learning is poor and the benefits mainly comes from lenses. The assistance of the *gestalt* plug-in is growing with the size of the search space. Finally, from results obtained by Experiment 1, 2 and 3, we can say "yes" to the main Question 1.

6 Related and future works

6.1 Intrinsic emergence

Originally [6, 45], the *Gestalt* heuristic has been inspired by the concept of intrinsic emergence due to Crutchfield [11]. Crutchfield provides the following definition of intrinsic emergence [11]:

> [. . .], pattern formation is insufficient to capture the essential aspect of the emergence of coordinated behaviour and global information processing, [. . .] At some basic level, though, pattern formation must play a role. [. . .] What is distinctive about intrinsic emergence is that patterns formed confer additional functionality which supports global information processing, [. . .] During intrinsic emergence there is an increase in intrinsic computational capability, which can be capitalized on and so lends additional functionality.

Bersini [6] argued that intrinsic emergence relaxes the "subjectity" of emergence by avoiding the presence of an external human observer. The "anthropomorphisation" of the phenomenon of emergence is antagonistic to any scientific practice that aims at not leaving subjectivism a leg to stand on. A "functional device" must substitute the human observer that will fine-tune its observation of any macro-phenomena produced by the system. In other words, intrinsic emergence is a mechanism adding useful functionalities to a system, namely a system exhibiting intrinsic emergence should be capable of introspection on itself. It should observe its macroscopic and functional properties and assimilate them, presenting thus some form of internal feedback of the whole, i.e. a whole perceived by something else but somewhat able to act on itself [46].

In the case of *Gestalt* heuristic, the system is a given metaheuristic and the structural patterns are described through the *gestalt* variables. The lenses are the means to reintroduce the discovered functional patterns. They alter the original process of the system and boost its search capacities. The *gestalt* learner is the mechanism that detects these structural and functional patterns and provides this information through the lenses. In such a way, the *Gestalt* heuristic appears to enrich any metaheuristic with this "intrinsic emergent" property.

6.2 Model based search algorithms

Model based search algorithms generate candidate solutions using a parametrized probabilistic model [68]. On the basis of new generated solutions, the probabilistic model is updated favouring highly promising region. There exist different kinds of model based search algorithms. The most popular ones are ant colony optimization [17], cross-entropy methods [51], estimation of distribution algorithms [44, 37] such as population-based incremental learning algorithm [3] or hierarchical bayesian optimization algorithm [43]. Some of them are solely based on the sample of current solutions. Others use additional information collected during the search and stored in an auxiliary memory [68]. As Zochlin et al precised [68], the term "model" does not mean an approximate description of the environment. They however noticed the analogy between the adaptive stochastic mechanism and the modeling of the structure of the "promising solutions".

The *Gestalt* heuristic builds structure in terms of *gestalt* variables. At first glance, it could be classified as a model based search algorithms. But it is not for several reasons. First, the *Gestalt* heuristic is built over a given metaheuristic and builds an approximate and macroscopic structure of the problem by observing the metaheuristic process. The *Gestalt* heuristic adapts and factorizes the problem representation for a specific metaheuristic. Without the metaheuristic, the *gestalt* plug-in just plugs on nothing. Second, model based search algorithms aim at directly building candidate solutions. The *gestalt* approach aims at improving a metaheuristic process by reformulating the problem. Third, the *gestalt* learner is more based on reinforcement learning techniques than on optimization ones. The search space is considered like an environment where the metaheuristic behaves such as one or more agents. If a model is built within the *gestalt* learner, it is an approximate description of the environment, i.e. the search space. The *gestalt* learner does not attempt to explicit everything it can measure, but rather tests what is intrinsically important for the given metaheuristic. The *Gestalt* heuristic processes at systemic levels and rather needs to be construed as a meta-model based algorithm.

Nevertheless, the *gestalt* learner is stated as a learning problem that could be considered as an optimization problem. There are no constraint to implement it in a way or another. Model based search algorithms could be a good implementation solution for it. For instance, we can imagine ants that would deposit pheromone trails behind the metaheuristic actions over a partial subgraph of the search space. The lenses would be produced on the basis of all these trails.

6.3 Evolutionary computation

In this paper, we have suggested an evolutionary implementation of the *Gestalt* heuristic (see Sec. 4). This implementation could present some analogies with important concepts such as "building blocks". We aim at clarifying some of them and avoiding any ambiguity or misunderstanding of the true fundamentals of the *Gestalt* heuristic.

6.3.1 Building block hypothesis and linkage learning

A building block is a group of closely located alleles [29]. The building block hypothesis is a theory of adaptive dynamics of the SGA that was firstly suggested by Holland [29] and inspired by Fisher's theories of sexual evolution [21]. The building block hypothesis describes an adaptive mechanism which basically recombine such building blocks. Nowadays this hypothesis is always being discussed and still the subject of controversies. Some hierarchical problems such as the Royal Road functions [40] were specifically built to test the building block hypothesis, especially the hierarchical assembly of building blocks. As a consequence of the failures of SGA, new algorithms were designed to discover and learn linkage i.e. the relationship between genes or the closeness of genes in building blocks. Linkage is a property of the tackled problem. For instance, there are messy GA [26], compact GA [28], or hierarchical bayesian optimization algorithm [43].

In *Gestalt* heuristic, *gestalt* variables are defined as group of variables. In the EA framework, there could be considered as group of genes but there are not to be confused with building blocks. Linkage learning algorithms base their process on the discovering of dependencies between valued variables for a given problem. The *gestalt* learner builds new *gestalt* description on the basis of the efficiency of the metaheuristic search process. Therefore, the *gestalt* description structure depends on the used metaheuristic and the tackled problem. But the goal is different: one aims at describing the problem in terms of dependencies while the other aims at improving the metaheuristic process by altering the problem representation. The *Gestalt* heuristic transforms the problem for a specific optimization algorithm. It is not important to discover true linkage but it is important to efficiently translate the problem for the metaheuristic being used.

6.3.2 Compositional evolution

Watson [62] defines compositional evolution as:

> [. . .] evolutionary processes involving the combination of systems or subsystems of semi-independently preadapted genetic material.

In this biological metaphor, a complex solution then results from the interaction of partial solutions. Parts of solution are evolved through the interaction with another. Some algorithms are based on compositional evolution such as symbiogenic evolutionary adaptation model [61], or evolutionary transition algorithm [16]. Both use two mains ideas behind the composition evolution approach: identifying good collaborations by symbiotic relations and aggregating partial solutions in complex solutions.

The *Gestalt* heuristic does not aim at building solution to a classical problem. It is more a learning algorithm dedicated to optimization algorithms than a true optimization algorithm. The *Gestalt* heuristic transforms the problem representation by fixing a particular structure through the use of lenses. It does not induce a division of solution in parts. However, the used metaheuristic could be a compositional EA.

In the suggested evolutionary implementation, we use islands of solutions that are used for the evaluation of a lens. Skolicki [55] noticed that, under some conditions, Multi-Islands Model induces a compositional evolution:

> In the cooperative mode, islands exchange parts of solutions by recombining migrants and local individuals, which results in a compositional evolution at the inter-island level. [. . .] I proposed a configuration with many small islands exchanging random individuals. I showed that such configuration performs better for problems that may be solved by compositional evolution, in particular for modular, hierarchical problems.

We can conclude that the evolutionary implementation of the *Gestalt* heuristic turns the original EA into a compositional EA.

6.4 Future works

In this paper, we have addressed the problem of online assistance of a given metaheuristic. The suggested solution consists in a new algorithmic approach defined by the *Gestalt* heuristic. The defined framework is totally independent of any algorithmic implementation details, even if we have presented general guidelines to make an evolutionary implementation. A lot of possible future works are then easy to conceive especially in terms of implementation solution.

6.4.1 Implementation of *gestalt* learner

As mentioned in Sec. 6.2, model based search algorithms could be interesting implementations for the *Gestalt* heuristic. But there are plenty optimization or learning algorithms. The *Gestalt* heuristic is akin to an interface for hybridizations where a second algorithm is added to a first one and aims at improving its process. There are a lot of possible combinations. Because of the hugeness of the lenses search space, combinatorial optimization algorithms can effectively be a good choice.

Another possibility is the use of reinforcement learning techniques, especially ones that are developed for stochastic routing of agents in graph. In Sec. 3.2, we have noticed that the search process of a metaheuristic can be seen as a walk on the neighborhood graph. For trajectory based metaheuristics, the process is like an agent walking on the neighborhood graph $\nu(\pi)$. For population based metaheuristics, we can consider a population of agents that can eventually communicate together. In both cases, one or more agents walk on $\nu(\pi)$ and search for the global optimum. Agents do not know if the visited node is the goal, i.e. the global optimum. Each agent step is defined by operators and cost a same amount of time: one iteration time unit. An agent could be positively rewarded or negatively. The reward $r_{ss'}$ depends on the agent progression to the goal. This scenario can be considered as a stochastic routing of agents [1]. The routing policy is nothing else than the tuned operators that the metaheuristic uses. The optimal routing policy is the set $\varpi = \{p_{ss'}\}$ that maximizes the total reward R_{ϖ}, i.e. the measure of the closeness to the goal. This graph perspective could provide an interesting reinforcement learning implementation of the *gestalt* learner. *Gestalt* descriptions would be produced according to the probability transition matrix. However, we notice some constraints such as the impossibility to know the whole graph $\nu(\pi)$ and the probability transition matrix.

6.4.2 A more sophisticated *gestalt* description

As defined in Sec. 3.1, a *gestalt* variable is a set of variables: $G \coloneqq \{\dots, y_j, \dots\}$. In this set G, no relation between variables is defined. In *gestalt* description Γ, *gestalt* variables are not put in relation. Let us roughly imagine some possible extension:

- considering a *gestalt* variable as a variable and allowing this way hierarchical composition
- endowing a *gestalt* variable with a relation between variables
- endowing a *gestalt* description with a relation between *gestalt* variables

These suggestions reinforce the principle of functional composite data structure of the *Gestalt* perspective and need a grammar to be easily defined or programmed. The *gestalt* learner would then reprogram the problem representation. These features need special techniques permitting reprogrammation such as genetic programming [47] or grammar evolution [52].

7 Conclusion

This paper is motivated by the following question: "Considering computation time and implementation ease constraints, is it then possible to build a system increasing the efficiency of the used metaheuristic by automatically altering the problem representation during the optimization process?". We suggest the *Gestalt* heuristic as a valid solution. This heuristic aims at improving online the way the problem being solved needs to be represented. The mechanism mainly consists in a meta-model of the inherent structure of the problem. This meta-model is used as a filter or a lens inserted between the metaheuristic and the problem representation. In such a way, the operators of the metaheuristics are altered and the sample produced by the metaheuristic is biased. From an engineering point of view, we see this *Gestalt* heuristic as a promising additional mechanism for rapid prototyping and anytime optimization context.

The paper introduces a new family of heuristics based on the consideration of the problem representation. Except in the EA community, the representation problem is not really considered in other metaheuristics. By formalizing the *Gestalt* heuristic, we have provided a common and unified framework of the meta-model adaptation to all metaheuristics. Moreover, this perspective could be interpreted as a very generic hyperheuristic. To support the idea, an EA implementation and a guideline of its application to other EAs are explained. The implementation is realized and experimentally tested. Besides, the assistance of *Gestalt* is an increasing function of the size and the computation time constraint. And we can answer the main question: "Yes, the *Gestalt* heuristic is one of this kind of systems.".

As noticed in experimental results, the current EA implementation, the GGA, does not totally realize an efficient implementation of the *gestalt* learner, i.e. the online learning mechanism of the problem structure. More studies are needed to improve the EA implementation. But other implementation ways need to be investigated, especially machine learning techniques.

A *Gestalt* description of permutation

For the TSP, each variable $x_i \in X$ represents a city and a candidate solution describes a tour of cities or a permutation $p = (\dots, x_i, \dots)$. The permutation is defined by the mapping function $\mathcal{D}$ that maps each city onto an integer $1, \dots, |X|$ such that each integer is mapped to an unique city x_i.

With a lens, a *gestalt* description is defined. The variables, i.e. the cities, are then aggregated into *gestalt* variables G_i, or megacities. The permutation is defined over the set of megacities and a candidate solution is reformulated as a tour of megacities $p = (\dots, G_i, \dots)$. In a megacity, the cities are also ordered. This order is determined while applying a new lens and respect the initial order and proximity of the cities. And the last city of a megacity is connected to the first city of the next megacity and so on. A direct bijection exists between a permutation of megacities and permutation of cities. Thus, the same operators can be simply applied by considering the new level of representation. For instance, the PMX will work by matching labels of megacities and ERX by recombining the edges between megacities.

Acknowledgements

The authors would like to thank Thomas Stützle and Utku Salihoglu for their comments, suggestions and useful discussion about this work, and Reinhardt Chamonal for his English corrections.

References

[1] Y. Achbany, F. Fouss, L. Yen, A. Pirotte, and M. Saerens. Tuning continual exploration in reinforcement learning. In S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, editors, *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN 06)*, volume 4132 of *Lecture notes in Computer Science*. Springer, September 2006.

[2] M. Ash. *Gestalt Psychology In German Culture 1890 - 1967*. Cambridge University Press, 1995.

[3] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University, 1994.

[4] L. Barbulescu, J.-P. Watson, and D. L. Whitley. Dynamic representations and escaping local optima: Improving genetic algorithms and local search. In *AAAI/IAAI*, pages 879–884, 2000.

[5] R. Battiti, M. Brunato, and F. Mascia. *Reactive Search and Intelligent Optimization*. Operations research/Computer Science Interfaces. Springer Verlag, 2008. in press.

[6] H. Bersini. Whatever emerges should be intrinsically useful. In *Artificial life 9*, pages 226–231. The MIT Press, 2004.

[7] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[8] V. Bruce, P. R. Green, and M. Georgeson. *Visual Perception: Physiology, Psychology and Ecology*. Psychology Press Ltd, 3d edition, 1996.

[9] J.-S. Chen, Y.-T. Lin, and L.-Y. Chen. A relation-based genetic algorithm for partitioning problems with applications. In H. G. Okuno and M. Ali, editors, *New Trends in Applied Artificial Intelligence, 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2007*, volume 4570 of *Lecture Notes in Computer Science*, pages 217–226. Springer, 2007.

[10] P. I. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *PATAT '00: Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, pages 176–190, London, UK, 2001. Springer-Verlag.

[11] J. Crutchfield. Is anything ever new? considering emergence. In D. P. G. Cowan and D. Melzner, editors, *Integrative Themes*, volume XIX of *Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

[12] J. Culberson. Holland's royal road and giga. GA-List v7n23, August 1993.

[13] M. M. Dastani. *Languages of Perception*. PhD thesis, Institute for Logic, Language and Communication, Universiteit van Amsterdam, 1998.

[14] E. D. de Jong, D. Thierens, and R. A. Watson. Hierarchical genetic algorithms. In X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. T. A. Kabán, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 232–241, Birmingham, UK, 2004. Springer-Verlag.

[15] P. De Lit, E. Falkenauer, and A. Delchambre. Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Math. Comput. Simul.*, 51(3-4):257–271, 2000.

[16] A. Defaweux. *Evolutionary Transitions as a Metaphor for Compositional Search: Definition and Evaluation of a New Optimisation Algorithm*. PhD thesis, Computational Modeling Lab, Computer Science Department, Vrije Universiteit Brussel, 2006.

[17] M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[18] E. Falkenauer. A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, 2(2):123–144, 1994.

[19] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[20] E. Falkenauer. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[21] S. R. A. Fisher. *The Genetical Theory of Natural Selection*. Dover Publications Inc., 2nd - revised edition, 1958.

[22] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning*, 13:285–319, 1993.

[23] M. R. Garey, R. L. Graham, and D. S. Johnson. Some np-complete geometric problems. In *STOC '76: Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 10–22, New York, NY, USA, 1976. ACM Press.

[24] O. O. Garibay, I. I. Garibay, and A. S. Wu. The modular genetic algorithm: Exploiting regularities in the problem space. In *In Proceedings of ISCIS 2003 The International Symposium on Computer and Information Systems, LNCS*, pages 584–591. Springer-Verlag, 2003.

[25] M. R. Glickman and K. P. Sycara. Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 194–201, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[26] D. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, (3):493–530, 1989.

[27] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[28] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. In *IEEE Transactions on Evolutionary Computation*, volume 3, pages 523–528, 1999.

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

[30] J. H. Holland. Royal road functions. GA-List v7n22, August 1993.

[31] H. H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications*. Elsevier, 2004.

[32] D. S. Johnson and L. A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The travelling salesman problem: A case study in local optimization, pages 215–310. John Wiley & Sons, Inc., 1997.

[33] D. S. Johnson and L. A. McGeoch. *The Traveling Salesman Problem and Its Variations*, chapter Experimental analysis of heuristics for the STSP, pages 369–443. Kluwer Academic Publishers, 2002.

[34] T. Jones. A description of holland's royal road function. *Evol. Comput.*, 2(4):409–415, 1994.

[35] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, The University of New Mexico, Albuquerque, NM, 1995.

[36] K. Koffka. *Principles of Gestalt Psychology*. Harcourt (New York), 1935.

[37] P. Larranaga. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Kluwer Academic Publishers, 2002.

[38] T. Lenaerts. *Different Levels of Selection in Artificial Evolutionary Systems: Analysis and Simulation of Selection Dynamics*. PhD thesis, Department of Computer Science, Vrije Universiteit Brussel, 2003.

[39] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[40] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, Cambridge, MA, 1992. MIT Press.

[41] A. Moraglio, Y.-H. Kim, Y. Yoon, and B.-R. Moon. Geometric crossovers for multiway graph partitioning. *Evol. Comput.*, 15(4):445–474, 2007.

[42] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[43] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*. Springer, 2005.

[44] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.*, 21(1):5–20, 2002.

[45] C. Philemotte and H. Bersini. Coevolution of effective observers and observed multi-agents system. In *Advances in Artificial Life, 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005, Proceedings*, volume 3630 of *Lecture Notes in Computer Science*, pages 785–794. Springer, 2005.

[46] C. Philemotte and H. Bersini. A gestalt genetic algorithm: less details for better search. In H. Lipson, editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1328–1334, New York, NY, USA, 2007. ACM.

[47] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[48] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, editors, *Proceedings of the Third Workshop on Foundations of Genetic Algorithms*, pages 51–72. Morgan Kaufmann, 1994.

[49] G.-C. Rota. The number of partitions of a set. *American Mathematical Monthly*, 71(5):498—504, 1964.

[50] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer, 2nd edition, 2006.

[51] R. Y. Rubinstein. Cross-entropy and rare events for maximal cut and partition problems. *ACM Trans. Model. Comput. Simul.*, 12(1):27–53, 2002.

[52] C. Ryan, J. J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming, First European Workshop, EuroGP'98, Paris, France, April 14-15, 1998, Proceedings*, volume 1391 of *Lecture Notes in Computer Science*, pages 83–96. Springer, 1998.

[53] J. Schaffer and L. Eshelman. On Crossover as an Evolutionary Viable Strategy. In R. Belew and L. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991.

[54] Z. Skolicki and K. A. D. Jong. Improving evolutionary algorithms with multi-representation island models. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 420–429. Springer, 2004.

[55] Z. M. Skolicki. *An Analysis of Island Models in Evolutionary Computation*. PhD thesis, Department of Computer Science, George Mason University, 2007.

[56] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[57] M. Toussaint. *The evolution of genetic representations and modular adaptation*. PhD thesis, Institut für Neuroinformatik, Ruhr-Universiät Bochum, 2003.

[58] A. Tucker, J. Crampton, and S. Swift. Rgfga: An efficient representation and crossover for grouping genetic algorithms. *Evol. Comput.*, 13(4):477–499, 2005.

[59] T. Walters. Repair and brood selection in the traveling salesman problem. In A. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands*, volume 1498 of *Lecture Notes in Computer Science*, pages 813–822. Springer-Verlag, 1998.

[60] R. A. Watson. Analysis of recombinative algorithms on a non-separable building-block problem. In W. N. M. William M. Spears, editor, *Proceedings of the Sixth Workshop on Foundations of Genetic Algorithms*, pages 69–90. Morgan Kaufmann, 2001.

[61] R. A. Watson. *Compositional evolution: interdisciplinary investigations in evolvability, modularity, and symbiosis*. PhD thesis, Brandeis University, 2002. Adviser-Jordan B. Pollack.

[62] R. A. Watson. *Compositional Evolution: The Impact of Sex, Symbiosis, and Modularity on the Gradualist Framework of Evolution*. Vienna Series in Theoretical Biology. The MIT Press, 2006.

[63] R. A. Watson, G. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 97–108, London, UK, 1998. Springer-Verlag.

[64] M. Wertheimer. *A Source Book of Gestalt Psychology*, chapter Laws of Organization in Perceptual Forms, pages 71–88. New York: The Humanities Press, 1938. English translation of the original paper published as Untersuchungen zur Lehre von der Gestalt II (1923).

[65] M. Wertheimer. *Productive thinking*. Harper (New York), 1959.

[66] R. P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, Department of Computer Science, George Mason University, 2003.

[67] S.-C. Zhu. Embedding gestalt laws in markov random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11):1170–1187, 1999.

[68] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1–4):373–395, 2004.