A Framework for Knowledge Integrated Evolutionary Algorithms

Ahmed Hallawa[1(✉)], Anil Yaman[2,3], Giovanni Iacca[2], and Gerd Ascheid[1]

[1] Chair for Integrated Signal Processing Systems,
RWTH Aachen University, 52056 Aachen, Germany
{hallawa,ascheid}@ice.rwth-aachen.de
[2] INCAS3, P.O. Box 797, 9400 AT Assen, The Netherlands
giovanni.iacca@gmail.com
[3] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.yaman@tue.nl

AQ1

Abstract. One of the main reasons for the success of Evolutionary Algorithms (EAs) is their general-purposeness, i.e. the fact that they can be applied in a straight forward manner to a broad range of optimization problems, without any specific prior knowledge. On the other hand, it has been shown that incorporating a priori knowledge, such as expert knowledge or empirical findings, can significantly improve the performance of an EA. However, integrating knowledge in EAs poses numerous challenges. It is often the case that the features of the search space are unknown, hence any knowledge associated with the search space properties can be hardly used. In addition, a priori knowledge is typically problem-specific and hard to generalize. In this paper, we propose a framework, called Knowledge Integrated Evolutionary Algorithm (KIEA), which facilitates the integration of existing knowledge into EAs. Notably, the KIEA framework is EA-agnostic, i.e. it works with any evolutionary algorithm, problem-independent, i.e. it is not dedicated to a specific type of problems and expandable, i.e. its knowledge base can grow over time. Furthermore, the framework integrates knowledge while the EA is running, thus optimizing the consumption of computational power. In the preliminary experiments shown here, we observe that the KIEA framework produces in the worst case an 80% improvement on the converge time, w.r.t. the corresponding "knowledge-free" EA counterpart.

AQ2

Keywords: Evolutionary algorithms · Knowledge incorporation · Landscape analysis · Evolutionary algorithm fingerprint

1 Introduction

Evolutionary Algorithms (EAs) are considered nowadays a valuable search and optimization tool suitable for many real-world problems characterized by complex multidimensional search spaces. Among the many applications of EAs, some

G. Squillero and K. Sim (Eds.): EvoApplications 2017, Part I, LNCS 10199, pp. 1–17, 2017.
DOI: 10.1007/978-3-319-55849-3_42

notable examples include the optimal design of electronic circuits [1], software [2,3], and even antennas for satellites orbiting outer space [4,5].

Despite the EAs' versatility, the theoretical limitations stated by the "No free lunch" (NFL) [6] pose a limit to their efficiency and applicability. As a possible mitigation for this problem, an EA can be made efficient and effective across a wide range of problems by endowing it with adaptive behavior with respect to the problem structure, thus with problem-specific mechanisms.

Such adaptation typically involves the EA's operators and tunable parameters, which play a pivotal role in the performance of the algorithm. In fact, there are several methods that can be used to optimize the behavior of EA. In a recent survey published by Črepinšek et al. [7], a number of approaches that can be used for this purpose are presented. Traditionally, researchers have used trial-and-error approaches in the attempt to find the best settings of the EA operators that can solve optimization problems most efficiently [8,9]. However, these approaches are typically computationally expensive, because they require numerous iterations (and in many cases it is not feasible to try all possible parametric combinations), and some are problem-specific, thus they can not be generalized to use on problems other than the one for which the tuning was performed. Furthermore, proposed frameworks which offer an adaptive behavior such as in some modulated versions of Differential Evolution (DE) algorithms, as in jDE [10] and JADE [11], does not offer a comprehensive strategy for all tunable parameters. Other approaches uses hyper-heuristics, i.e. they find the optimal EA settings by using an optimization algorithm [7,12].

One of the approaches that have not yet been explored, however, is to use experiences from previous problems with similar population behavior in the EA run [7]. In this work, we propose a framework that is a first attempt in this direction, where we also combine the approaches of "following general guidelines" accumulated in the literature, and "identifying the features of the landscape by a classifier, in order to propose good control parameters" [7].

The framework we propose is dubbed as *Knowledge Integrated Evolutionary Algorithm* (KIEA). Its main component is a *knowledge base* that maintains the knowledge of how various functions, i.e. optimization problems can be efficiently solved. These functions are named as *pilot functions*, and the associated knowledge to optimally tune the EA for solving those functions is named a *strategy*. The framework collects characteristics of the EA population behavior across generations. These characteristics are named *EA fingerprints*, and they are used to classify any unknown function under investigation w.r.t. each of the pilot functions (under the implicit hypothesis that such fingerprints can be used to assess the similarity between different functions). The strategy associated to the classified pilot function is then reused on the unknown function at hand, in the attempt of solving it in the most efficient way possible.

The approach we propose is novel in the following ways: Firstly, it allows the incorporation of various types of knowledge into the EA, allowing the algorithm to adjust its behavior based on the knowledge in the knowledge base. Initially, experts can bootstrap the system with their knowledge on the pilot

functions. Secondly, the experience gained by the EA by solving problems generates valuable empirical knowledge that can also be added to the knowledge base for further use. This can be done by extending the pilot functions or changing the strategies associated with them. As a result, the knowledge base grows by accumulating the experience gained by solving multiple problems. The accumulated knowledge is then "plugged" into the problems that are similar to the ones encountered before. Moreover, the KIEA approach is generalizable as it conducts the problem classification only based on the behavior of the population in the EA run, independently from what this population is representing, i.e. the solution encoding and the genotype/phenotype mapping.

To assess the performance of KIEA, you see preliminary experiments on a small set of benchmark optimization problems. More specifically, we measure the fingerprint-based classification accuracy on different pilot functions by using different fingerprint properties. In addition to that, we compare the difference in terms of convergence time obtained in the experiments with and without KIEA.

The rest of the paper is organized as follows: Sect. 2 summarizes the previous works on knowledge integration in EAs, Sect. 3 presents the mathematical foundation of the EA fingerprint and the classification process. Section 4 demonstrates and discusses performance evaluations of our approach. Finally, Sect. 5 concludes with the paper.

2 Background

Different knowledge incorporation methodologies available in the literature aim to optimize EAs in order to enhance their performance. One of the key elements where knowledge plays a role is the balance between exploration and exploitation, a crucial aspect for an efficient search [7,13]. If there is, for example, more influence of exploration then the search becomes more like a random search; on the other hand, if exploitation is stronger than exploration then the search space could not be explored, and the behavior of the search becomes similar to the behavior of hill climbing [7].

In EAs, the balance between exploration and exploitation is typically adjusted by the evolutionary operators and their parameters. However, different evolutionary operators and different parameter values influence the process differently; and their combinations may have different, hard-to-analyze effects. One general interpretation considers the mutation and crossover as exploration operators[1], since they make (pseudo-)random changes in the genotype of individuals and cause random jumps in the search space; on the other hand, the selection operator is usually seen as an exploitation operator, because it focuses on specific places by selecting the individuals to reproduce. Moreover, the population size plays an important role in the EA behavior. Generally, re-sizing population

[1] It should be noted, however, that some literature considers the crossover operator as an exploitation mechanism. Generally, mutation and crossover have an effect on both exploration and exploitation, although this effect varies depending on the implementation and the fitness landscape at hand.

can be used to direct evolution process towards exploration or exploitation [14–16], this method is widely used in CMA-ES, DE, and PSO. In that regard, it is also important to highlight that the increase of the population size does not necessarily improve exploration. Conversely, it has shown that there is strong link between population size and structural bias of the algorithm [17]. Consequently, this component has to be taken into consideration when designing self-adapting EA.

The performance of a search process is also closely dependent on the features of the search space and the fitness landscape. There have been many works in the literature that aim to classify the landscapes based on their geometrical and topological features. Most of these works are linked to specific features such as modality, symmetry, etc., such as in [18–21]. There have also been several studies that aim to suggest optimal algorithms, or optimal algorithm parameters, based on the features of the landscape of the search space. For example, Asmus et al. [22] proposed a system for recommending suitable algorithms for a given black-box optimization problem. Muoz et al. [23] introduced a model that links the landscape analysis measures and the algorithm parameters (used CMA-ES) to predict the performance of the algorithm parameters. Picek and Jakobovic [24] performed a thorough study focused on the correlation between fitness landscapes and crossover operators.

Clearly, it would be extremely beneficial to leverage this wide range of knowledge from the literature for tuning the evolutionary operators parameters based on the landscape features. However, applying this knowledge often requires that such features are captured first, in order to choose the proper strategies. Unfortunately though, this is not always possible since either the search space is completely unknown, or hard to characterize.

Moreover, many of the existing works offer knowledge that is problem-specific and therefore hard to generalize and use with other problems. Another difficulty arises from the fact that this knowledge is often scattered over different levels of granularity, from too general to extremely detailed, which makes it hard to have comprehensive strategies.

The objective of the presented work is to propose a way of integrating existing *algorithmic knowledge* into a single, comprehensive evolutionary framework, and test the effect of such knowledge on the optimization performance. Here, with "algorithmic knowledge" we generally refer to the knowledge encompassing the categorizations of problems based on their landscape features, the types of strategies related to the adaptation of evolutionary operators and parameters, and the link between problem types and strategies, i.e. which strategies work best on a specific type. In the next section a detailed description of the framework is presented.

3 Methods

The KIEA methodology is straightforward and its implementation is relatively simple (see Algorithm 1). For completeness, we also report a conceptual scheme of the proposed KIEA in Fig. 1.

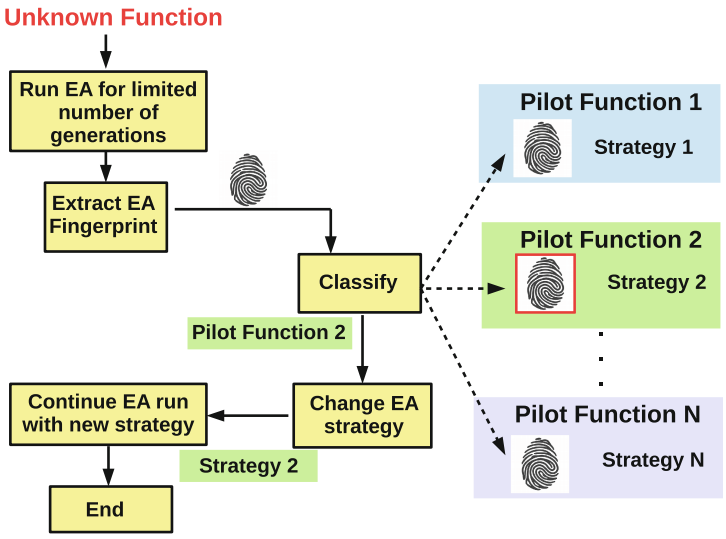

Fig. 1. Conceptual scheme of KIEA

The whole process can be considered as a single run of an EA divided into two stages: in the **first stage**, for a predefined number of generations, G_C, the unknown function under investigation undergoes an EA run with an arbitrary parameter setting (basically, population size, mutation and crossover probability). These settings are set as the initial EA strategy S_0. In each generation evolved in this first stage, a set of properties describing the population behavior is calculated to be used later for classification. These population behavior properties are termed *EA fingerprint* (see Sect. 3.1 for details). The first stage stops when the allotted number of generations G_C is reached.

In the **second stage**, a classification based on the EA fingerprint is performed as follows. The unknown function's fingerprint produced from the first stage is compared with the fingerprint obtained by the same initial strategy S_0 on a set of *pilot functions*, i.e. benchmark functions that are chosen a priori as representative of different categories of problems. Based on fingerprint similarities, the unknown function is then classified as the most similar pilot function. Detailed insights on the classification process are provided in Sect. 3.2.

Associated to each pilot function, the system maintains an EA *strategy*. Here, we refer to "strategy" as a set of changes (adaptation rules) in the EA parameters (e.g. a population size reduction/shrinking, mutation probability update rules, etc.) and when these changes should be implemented within the evolutionary run in order to enhance the optimization performance. These strategies are organized in a *knowledge base*, such that for each pilot function, its associated strategy is the one that according to our empirical experiments (see Sect. 4.1) showed the best performance.

Therefore, after the classification phase, the settings of the algorithm, i.e. mutation rate, population size, etc. are set according to the EA strategy that is associated with the most similar pilot function. For the remaining of the available generations, the EA runs with the chosen strategy. The hypothesis is that similar strategies can enhance the performance of an EA on functions with similar EA fingerprints; in other words, we expect the performance of the optimization process to improve on the unknown function after adopting the strategy associated with the most similar pilot function.

It is important to notice that the ability to classify problems based on the EA fingerprint makes it possible to transfer the knowledge from the pilot functions to any unknown function under investigation, without assuming any previous understanding of its features or properties. Moreover, this mechanism allows generalization of knowledge to a wide range of optimization problems without the need of associating such knowledge to specific fitness landscape properties such as modality, symmetry, etc. Therefore, it avoids the complexity due to landscape analysis, and makes it possible for the system to work with functions with complex, hard-to-analyze landscapes, since all that is required is to capture the population behavior in the EA run.

Algorithm 1. High-level description of the KIEA framework

```
1: procedure KIEA
2:     initialize total no. of generations G
3:     initialize no. of generations for classification G_C
4:     initialize generation counter g = 0
5:     set initial EA strategy S_0
6:     initialize population P
7:     while g < G_C do
8:         F ← evaluate (P)
9:         P ← select (P, F)
10:        P ← reproduce (P, F, S_0)
11:        f ← getFingerprint(P, F)                ▷ Store fingerprint
12:        g = g + 1
13:    end while
14:    Pilot_i ← classify(f)                       ▷ Classification
15:    S_i ← getStrategy(Pilot_i)                  ▷ Retrieve strategy
16:    while g < R_T do
17:        F ← evaluate (P)
18:        P ← select (P, F)
19:        P ← reproduce (P, F, S_i)
20:        g = g + 1
21:    end while
22: end procedure
```

In the following sections, we cover the mathematical details of the EA fingerprints and the classification procedure. In Table 1, we summarize the main symbols used in the text, with the related explanation.

Table 1. Symbols used in the paper

Symbol	Explanation
N	Total number of individuals in population
G	Total number of generations
G_C	Number of generations allocated for classification
$\mathbf{i}_l^g \in \mathbb{R}^n$	Individual belongs to cluster l at generation g
$\mathbf{c}_l^g \in \mathbb{R}^n$	Cluster l center individual at generation g
$c_{\min}$	Minimum number of individuals to form a cluster
$r_{\max}$	Maximum radius of the sphere S^n that a cluster can occupy
C_l^g	Set of individuals in cluster l at generation g
$\|C_l^g\|$	Number of individuals in cluster l at generation g
C_c^g	Set of cluster centers at generation g
$\mathcal{P}_l$	A tuple with the number of individuals in cluster l
d_{ij}^g	Euclidean distance between $\mathbf{c}_i^g$ and $\mathbf{c}_j^g$ at generation g
T_l	Population trend tuple of cluster l
$G_{d_{ij},\epsilon}^g$	Set of points that have equal Euclidean distance $d_{ij} \pm \epsilon$ at generation g
S_i	EA Strategy i

3.1 EA Fingerprint

In the previous section we defined the EA fingerprint as a set of properties that characterize the population behavior. While in principle this could be done at individual level, in practice following all individuals in the population would be extremely computationally expensive, especially in high dimensions. For example, a simple task as finding the pair of points in a set with smallest distance between them (known as *closest pair of points problem*) has time complexity of $O(n^2D/log^2D)$ for D dimensions and n points using a divide and conquer approach [25]. Therefore, we define here EA fingerprints that are based on *clustering* the population and following the resulting clusters properties.

A cluster emerges when a predefined minimum number of individuals c_{min} from the population are grouped in a predefined maximum space r_{max} in the search space. This minimum number of individuals constituting a cluster and the corresponding maximum space used to designate it are set as a percentage of the population size N, e.g. $c_{min} = 5\%$ of N.

An EA fingerprint is grouped into two main groups: Clusters Emergence Characteristics (CEC) and Clusters Constellation Characteristics (CCC). CEC is designed to capture 5 features: number of clusters, number of individuals in each cluster, population trend in each cluster, fitness value of the fittest individual of each cluster, and its position. On the other hand, CEC captures the geometric properties of clusters, which include equidistant cluster topology and the corresponding distances between equidistant clusters in the search space.

In CCC, the first step is recognizing emerging clusters. The procedures for that are described as follows:

1. All individuals $\mathbf{i}_l^g \in \mathbb{R}^n \; \forall \; l = 1 \ldots N$ at generation g are sorted in descending order with respect to fitness.
2. The highest value is designated as a potential cluster center $\mathbf{c}_l^g$ forming the first point in the potential cluster set C_l^g.
3. Going through all population, each individual $\mathbf{i}_l^g$ is assigned to C_l^g with center point $\mathbf{c}_i^g$ if and only if:

$$\|\mathbf{i}_i^g - \mathbf{c}_j^g\| < r_{max}, \quad \forall j \neq i \; \wedge \; j = 1 \ldots N \tag{1}$$

   where r_{max} is the maximum radius of the sphere S^n that a cluster can occupy, $S^n = \{x \in \mathbb{R}^{n+1} : \|x\| = r_{max}\}$. r_{max} is chosen adequately, e.g. 5% of the smallest search domain across all search variables.
4. A cluster C_l^g is designated with the center $\mathbf{c}_i^g$ if and only if the number of individuals assigned to it, $|C_l^g|$, is bigger than or equal to the minimum cluster size $c_{\min}$:

$$|C_l^g| \geq c_{\min} \tag{2}$$

   where $c_{\min}$ is chosen as a percentage of the total population size N, e.g. 5%.
5. All individuals that were previous assigned to a cluster or picked as a potential cluster center are then discarded, steps (3) to (5) are repeated again until all N individuals in the population are considered.

These procedures are executed for each generation, until G_C is exhausted. For each generation g, all cluster centers $\mathbf{c}_l^g$, $l = 1, 2, \ldots$, are kept in a set C_c^g. Furthermore, due to sorting population in the first step in the procedures, $\mathbf{c}_l^g$ are also the fittest points in cluster l. This will be used later for comparing clusters with similar highest fitness points. In addition, for each generation the corresponding fitness value of each cluster center $\mathbf{c}_l^g$ are stored in a fitness set F_c^g. A tuple $\mathcal{P}_l$ with the number of individuals in cluster l through out all generations until a given generation m is defined as follows:

$$\mathcal{P}_l^m = < |C_l^1|, |C_l^2| \ldots, |C_l^{\mathrm{m}-1}|, |C_l^{\mathrm{m}}| > \tag{3}$$

where m can take any value from 1 to total number of generations G. In order to capture the changes in each cluster throughout different generations until a given generation m, for each $\mathcal{P}_l^m$, there exists a population trend tuple T_l^m defined as:

$$T_l^m = \begin{cases} 1, & \text{for } |C_l^k| - |C_l^{k+1}| < 0 \\ 0, & \text{for } |C_l^k| - |C_l^{k+1}| > 0 \end{cases} \qquad \forall k = 1 \ldots m-1 \tag{4}$$

Now, all the aforementioned CEC properties can be defined: the number of clusters for each generation g in $|C_l^g|$, the number of individuals in each cluster l in $\mathcal{P}_l$, the population trend across all generations in each cluster in T_l, the fitness value of the fittest individual (which is also the center) of each cluster in F_c^g, and its position in C_c^g for each generation g.

The second fingerprint component's, CCC, is meant to capture the geometric properties of clusters. Firstly, we define the Euclidean distance between all cluster centers in each generation g as:

$$\mathrm{d}_{ij}^{g} = \|\mathbf{c}_i^g - \mathbf{c}_j^g\| \tag{5}$$

Then, we group equidistant cluster center points as follows:

$$\mathrm{G}_{d_{kl,\epsilon}}^{g} = \{(\mathbf{c}_i^g, \mathbf{c}_j^g) \in C_c^{g^2} \ \forall i \neq j,\ d_{kl} - \epsilon < \|\mathbf{c}_i^g - \mathbf{c}_j^g\| < d_{kl} + \epsilon\} \tag{6}$$

where ϵ is a margin of tolerance adequately chosen as a percentage of the domains of each search variable.

Both d_{ij}^{g} and $\mathrm{G}_{d_{kl,\epsilon}}$ constitute the CCC properties. Now that we have defined all the elements of the EA fingerprint, we can show how we use them for classification.

3.2 Classification

The objective of the classification process is to find the closest pilot function to the unknown function under investigation. This is conducted by comparing the EA fingerprints with the fingerprint of each pilot function. In the following description, ψ_k^g indicates the comparison of feature k at generation g.

The first comparison ψ_1^g is the difference in cluster numbers at each generation. This difference is multiplied by the ratio between the smallest cluster number over the biggest, as follows:

$$\psi_1 = \| |C_l^g|_u - |C_l^g|_p \| \times \frac{\min(|C_l^g|_u, |C_l^g|_p)}{\max(|C_l^g|_u, |C_l^g|_p)} \tag{7}$$

where $|C_l^g|_u$ and $|C_l^g|_p$ are the numbers of clusters on the unknown function and the pilot function, respectively.

The second comparison ψ_2^g is the difference in the position between each center point $\mathbf{c}_l^g$ in the unknown function and the closest cluster center point position in the pilot functions. Only points whose distance from $\mathbf{c}_l^g$ is at most r_{max} are considered, to ensure that the closest point found in the pilot function cannot be chosen more than once, since there cannot exist two center cluster points within the same cluster sphere S^n which has maximum radius r_{max}. Consequently, ψ_2^g is defined as the sum of all the minimum distances between $\mathbf{c}_l^g$ of unknown function and pilot function, as follows:

$$\psi_2^g = \sum_{\mathbf{c}_i^g \in {}^u\mathrm{C}_l^g} \min_{\mathbf{c}_j^g \in {}^p\mathrm{C}_l^g} \|\mathbf{c}_i^g - \mathbf{c}_j^g\| \text{ with } d_{ij} < r_{max} \tag{8}$$

where ${}^u\mathrm{C}_l^g$ and ${}^p\mathrm{C}_l^g$ are sets with center points $\mathbf{c}_l^g$ at generation g for the unknown function and pilot function, respectively. The inequality $d_{ij} < r_{max}$ is the minimum distance condition explained earlier.

The third comparison is defined to capture the difference in population trends between the unknown function and the pilot function for clusters with similar fittest point values, i.e. clusters which have close center fitness values across all generations. Center cluster points with similar fitness values are identified as follows:

$$ {}^{u}T^{g}_{f_0+k\epsilon} = \{f_l \in F^g_u | f_l - \epsilon \leq f_0 + k\epsilon \leq f_l + \epsilon\} \; \forall k \in \mathbb{N} \tag{9} $$

where f_0 is the least fit cluster center $\mathbf{c}^g_l$ in the unknown function at generation g, and F^g_u is the fitness set of the unknown function at generation g. ${}^{u}T^{g}_{f_0+k\epsilon}$ is a set of all points within range $f_0 + k\epsilon$, $\forall k \in \mathbb{N}$. The analogous value for the pilot functions, ${}^{p}T^{g}_{f_0+k\epsilon}$ is calculated similarly:

$$ {}^{p}T^{g}_{f_0+k\epsilon} = \{f_l \in F^g_p | f_l - \epsilon \leq f_0 + k\epsilon \leq f_l + \epsilon\} \; \forall k \in \mathbb{N} \tag{10} $$

where f_0 is least fit cluster center $\mathbf{c}^g_l$ in the unknown function at generation g (same as in Eq. 9), and F^g_p is the fitness set of the pilot function at generation g. Then ψ^g_3 is defined as:

$$ \psi^g_3 = \sum_{k \in \mathbb{N}} \frac{\min(|{}^{u}T^{g}_{f_0+k\epsilon}|, |{}^{p}T^{g}_{f_0+k\epsilon}|)}{\max(|{}^{u}T^{g}_{f_0+k\epsilon}|, |{}^{p}T^{g}_{f_0+k\epsilon}|)} \times \frac{\min(|C^g_l|_u, |C^g_l|_p)}{\max(|C^g_l|_u, |C^g_l|_p)} \tag{11} $$

where ψ^g_3 is the summation of the ratio of number of points in range $f_o + k\epsilon$ in the unknown function and pilot function $\forall k$, normalized w.r.t. the ratio of cluster sizes, exactly like in ψ^g_1. The reason why the min and max operators are used instead of simply having $|{}^{u}T^{g}_{f_0+k\epsilon}|$ in the numerator and $|{}^{p}T^{g}_{f_0+k\epsilon}|$ in the denominator, is to ensure that the value of the fraction is always less than one and therefore ψ^g_3 is comparable with other cases, regardless the fact that $|{}^{u}T^{g}_{f_0+k\epsilon}|$ is greater than $|{}^{p}T^{g}_{f_0+k\epsilon}|$ or vice versa.

The fourth comparison is used to capture the difference between the equidistant cluster center points, as follows:

$$ \psi^g_4 = \frac{\min(|\,\mathrm{G}^g_{d_{ij,\epsilon}}|_u, |\,\mathrm{G}^g_{d_{kl,\epsilon}}|_p)}{\max(|\,\mathrm{G}^g_{d_{ij,\epsilon}}|_u, |\,\mathrm{G}^g_{d_{kl,\epsilon}}|_p)} \text{ with } d_{kl} - \epsilon \leq d_{ij} \leq d_{kl} + \epsilon \tag{12} $$

where $\mathrm{G}^g_{d_{ij,\epsilon}}$ is the set of equidistant cluster centers within the margin of tolerance ϵ at generation g.

The fifth comparison captures the trend of the population within clusters. Its definition reflects similarities in the change of population within clusters with similar fitness values. f^g_l is the highest fitness point in each cluster l at generation g, and there might exist a f^g_l in the pilot function at the same generation g with fitness value that is within ϵ range from it, a set that includes all these pairs of points is defined as:

$$ M^g = \{(i,j) \in \mathbb{N}^2 | \; \forall f_i \in F^g_u, f_j \in F^g_p, \; \|f^g_i - f^g_j\| \leq \epsilon\} \tag{13} $$

where F^g_u and F^g_p are the fitness sets of the unknown function and pilot function respectively. M^g includes all cluster id pairs unknown function and pilot function

that have highest fitness values difference less than ϵ. From Eq. 17, each item at position k in the tuple reflects the change in population between generation k and $k+1$, its value be either zero for no change in population or one for increase in population or negative one for a decrease in population. For each pair identified in M^g, a comparison in the population number history up to generation g is defined as follows:

$$\psi_5^g = \sum_{(i,j)\in M} T_i^g \circledcirc T_j^g \tag{14}$$

where T_i^g and T_j^g are the trend tuples of the unknown function and pilot function respectively and $\circledcirc$ is defined here as an XNOR logic operator which produces 1 if operands are equal and 0 otherwise.

The sixth comparison captures the difference in the number of clusters per generation, as follows:

$$\psi_6^g = \sum_{g=1}^{G} \frac{\min(|C_l^g|_u, |C_l^g|_p)}{\max(|C_l^g|_u, |C_l^g|_p)} \tag{15}$$

Finally, the classification process is concluded based on ψ_1 to ψ_6, using a weighted sum as follows:

$$\psi_{Total} = \sum_{i=1}^{6} w_i \psi_i \tag{16}$$

where w_i is the weight of property i, which is set as follows:

$$w_i = \begin{cases} 1, & \text{for } i = 3, \\ -1, & otherwise \end{cases} \qquad \forall i = 1 \ldots 6 \tag{17}$$

as with exception to ψ_3, the lesser the value the better the match with the pilot function. Finally, the total value is then compared with each pilot function, and the highest value leads to the winning pilot function, which then leads to the unknown function adapting its strategy.

4 Results

As a proof-of-concept, we now present the numerical results obtained by KIEA on a small set of benchmark functions. First, we explain the algorithmic setup and the strategy initialization in Sect. 4.1. Then, in Sect. 4.2 we illustrate the effect of different fingerprint properties on classification. Finally, Sect. 4.3 evaluates the performance of KIEA in terms of convergence time.

4.1 System Setup and Strategy Initialization

Although the KIEA framework is algorithmically agnostic and can be used with any EA, for testing purposes we use a classic Genetic Algorithm (GA). In the

prototype we implemented, we focused on strategies expressed in terms of population size and mutation rates. Figures 2 and 3 show a preliminary performance analysis for two simple benchmark functions, namely the Ackley and the Gaussian function. It can be seen that for the Ackley function, population size 60 and mutation rate 0.01 is the most suitable strategy, while for the Gaussian function population size 40 and mutation rate 0.1 offer a more suitable strategy.

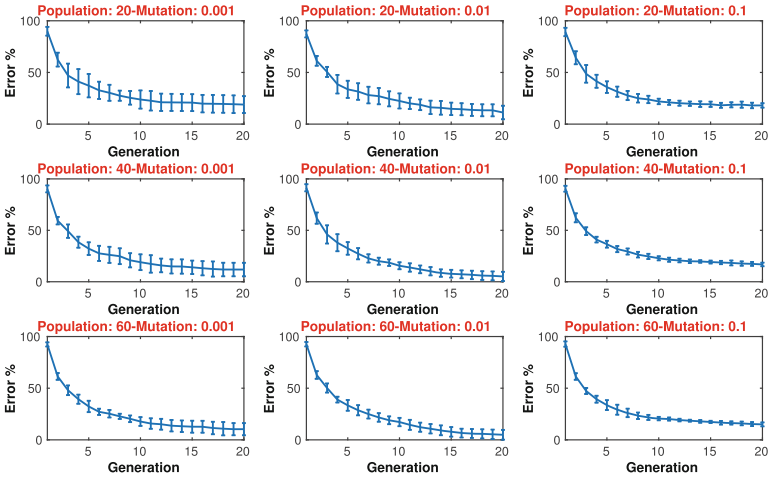

Fig. 2. Strategy analysis on the Ackley function

4.2 Effect of Fingerprints

To test the ability to classify a function based on its fingerprint, we conducted a four tests in total. In the first two, the objective was to classify the two pilot functions (Ackley and Gaussian) as if they were unknown. Moreover, we considered two additional unknown functions (Rastrigin and Rosenbrock), different from the pilot functions. Figures 4 and 5 show the classification of the Ackley, Gaussian, Rastrigin, and Rosenbrock functions, respectively. Each classification test was done by first extracting 50 different fingerprints for each pilot function, and then comparing each of the 50 fingerprints from the unknown function, thus with a total of 2500 comparisons (50 pilot functions fingerprints $\times$ 50 unknown function fingerprints). It is important to highlight that each property in the fingerprint contributes to the classification process, i.e. they are all needed and their contribution varies depending on the function at hand, while keeping the overall successful classification rate 90% to 98%. Moreover, their variance is relatively small (3% on average), which shows the good reliability of the proposed classification process (Fig. 6).

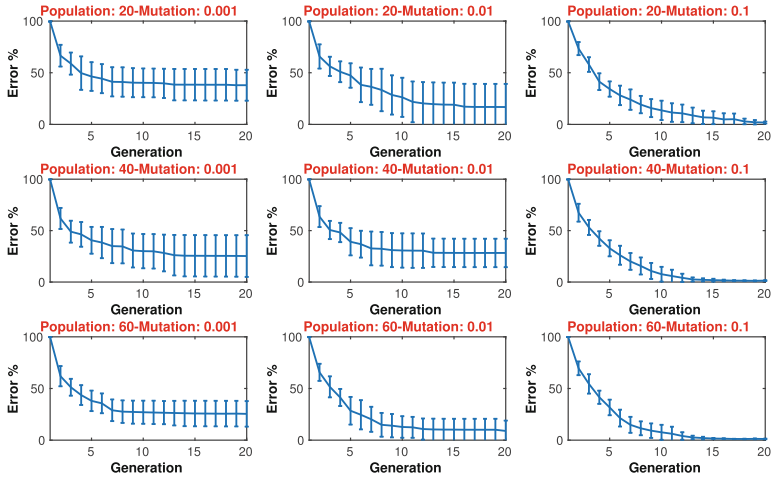

Fig. 3. Strategy analysis on the Gaussian function

4.3 KIEA Performance

We conclude the experimental validation of KIEA by measuring the performance gain in terms of convergence time. Table 2 summarizes the convergence time on the Ackley and Gaussian function in 10 dimensions, with and without KIEA. Tests are done 25 times per function and the 1st, 7th, 13th, 19th and 25th best convergence times out of the 25 runs are captured. All runs are done with FES $= 10^3$ and termination error $= 10^{-6}$. It is clear that there a decrease across all the best times when using KIEA, which reaches a 80% decrease in the worst case. Moreover, there is a significant decrease in standard deviation across all the runs, which suggests a more reliable performance when using KIEA (Fig. 7).

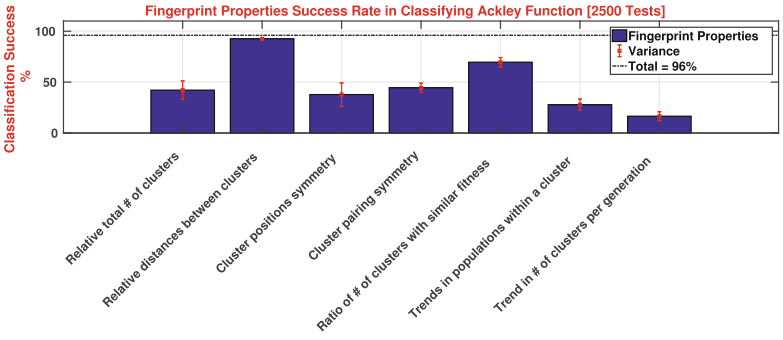

Fig. 4. Classification of the Ackley function

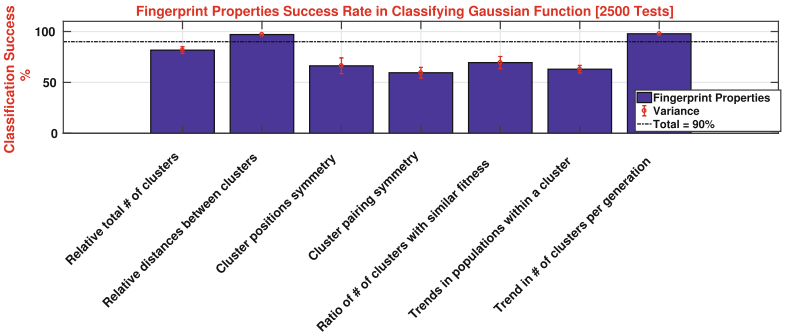

Fig. 5. Classification of the Gaussian function

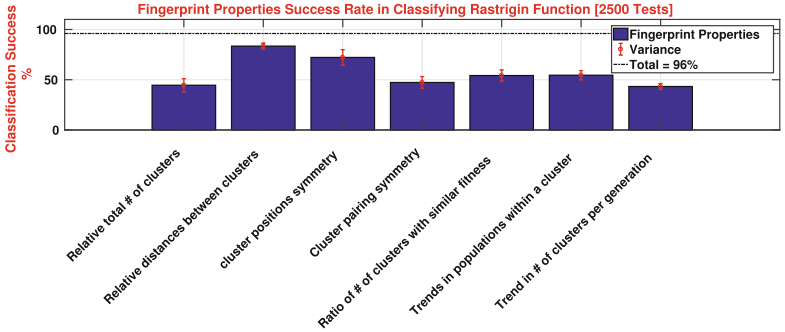

Fig. 6. Classification of the Rastrigin function

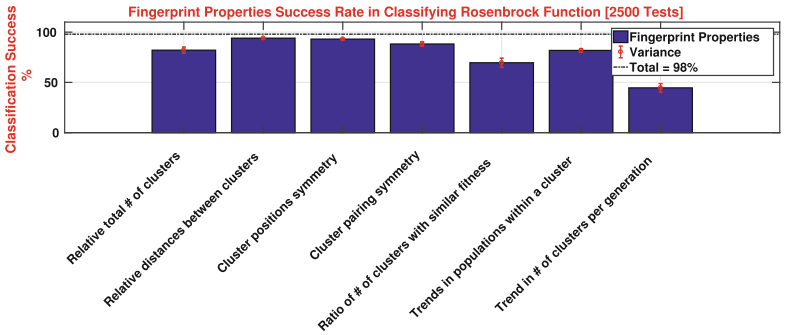

Fig. 7. Classification of the Rosenbrock Function

Table 2. Convergence time T [s] with and without KIEA

	Ackley (w/o KIEA)	Ackley (w/ KIEA)	Gaussian (w/o KIEA)	Gaussian (w/ KIEA)
1st	1.439	1.261	0.232	0.1179
7th	2.994	2.890	0.131	0.1194
13th	3.014	2.905	0.137	0.1200
19th	3.053	2.929	0.14	0.1283
25th	10.056	2.990	0.191	0.1658
Mean	3.677	2.791	0.1662	0.1303
Std	2.324	0.388	0.0439	0.0203

5 Conclusions

In this work, we have introduced a framework for knowledge integration in evolutionary algorithms. The framework, named KIEA, is based on the concept of *EA fingerprint*, i.e. a set of a properties that capture the population behavior in the solution space while the EA is running. In addition to the algorithmic description of the framework, we presented a mathematical formalization of the properties constituting the fingerprint.

In the preliminary experiments conducted in this study, the framework prototype showed a successful classification probability between 90% and 98%, depending on the function to optimize. Furthermore, the comparison of the convergence time on functions optimized with and without KIEA proved that the presented framework consistently enhances the convergence time, reaching a worst-case improvement of nearly 80%.

In addition to the improved numerical performance, the presented framework has the advantage of being fully expandable, since it is possible to add new pilot functions and strategies in a straightforward manner. Furthermore, KIEA is suitable for any evolutionary algorithm as it is not strictly bound to any specific EA implementation. In future works, we will test this framework on a wider range of optimization problems, and we will expand our knowledge base of pilot functions and corresponding strategies. Finally, we plan to perform tests with different kinds of state-of-the-art EAs, to show the general applicability of KIEA.

Acknowledgments. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 665347. We also gratefully acknowledge the computational resources provided by RWTH Compute Cluster from RWTH Aachen University under project RWTH0118.

References

1. Koza, J.R., Keane, M.A., Streeter, M.J.: What's ai done for me lately? genetic programming's human-competitive results. IEEE Intell. Syst. **3**, 25–31 (2003)
2. Arcuri, A., Yao, X.: Co-evolutionary automatic programming for software development. Inf. Sci. **259**, 412–432 (2014)
3. Squillero, G.: MicroGP - an evolutionary assembly program generator. Program. Evol. Mach. **6**(3), 247–263 (2005)
4. Hornby, G.S., Globus, A., Linden, D.S., Lohn, J.D.: Automated antenna design with evolutionary algorithms. In: AIAA Space, pp. 19–21 (2006)
5. Lohn, J.D., Linden, D.S., Hornby, G.S., Kraus, W.F., Rodriguez-Arroyo, A.: Evolutionary design of an X-band antenna for NASA's space technology 5 mission. In: null, vol. 155. IEEE (2003)
6. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
7. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput. Surv. (CSUR) **45**(3), 35 (2013)
8. Bäck, T.: Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 57–62. IEEE (1994)
9. Gates, G.H., Merkle, L.D., Lamont, G.B., Pachter, R.: Simple genetic algorithm parameter selection for protein structure prediction. In: IEEE International Conference on Evolutionary Computation, vol. 2, pp. 620–624. IEEE (1995)
10. Yang, M., Cai, Z., Li, C., Guan, J.: An improved adaptive differential evolution algorithm with population adaptation. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp. 145–152. ACM (2013)
11. Caorsi, S., Massa, A., Pastorino, M., Randazzo, A.: Optimization of the difference patterns for monopulse antennas by a hybrid real/integer-coded differential evolution method. IEEE Trans. Antenna Propag. **53**(1), 372–376 (2005)
12. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 124–141 (1999)
13. Eiben, A.E., Schippers, C.A.: On evolutionary exploration and exploitation. Fundamenta Informaticae **35**(1–4), 35–50 (1998)
14. Harik, G.R., Lobo, F.G.: A parameter-less genetic algorithm. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1, pp. 258–265. Morgan Kaufmann Publishers Inc. (1999)
15. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**(4), 287–297 (1999)
16. Iacca, G., Mallipeddi, R., Mininno, E., Neri, F., Suganthan, P.N.: Super-fit and population size reduction in compact differential evolution. In: IEEE Workshop on Memetic Computing (MC), pp. 1–8. IEEE (2011)
17. Kononova, A.V., Corne, D.W., Wilde, P., Shneer, V., Caraffini, F.: Structural bias in population-based algorithms. Inf. Sci. **298**, 468–490 (2015)
18. Beyer, H.G., Schwefel, H.P.: Evolution strategies-a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002)
19. Casas, N.: Genetic algorithms for multimodal optimization: a review. arXiv preprint arXiv:1508.05342 (2015)
20. Miller, B.L., Shaw, M.J.: Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 786–791. IEEE (1996)

21. Sareni, B., Krahenbuhl, L.: Fitness sharing and niching methods revisited. IEEE Trans. Evol. Comput. **2**(3), 97–106 (1998)
22. Asmus, J., Borchmann, D., Sbalzarini, I.F., Walther, D.: Towards an FCA-based recommender system for black-box optimization. In: Workshop Notes, p. 35 (2014)
23. Muñoz, M.A., Kirley, M., Halgamuge, S.K.: A meta-learning prediction model of algorithm performance for continuous optimization problems. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 226–235. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32937-1_23
24. Picek, S., Jakobovic, D.: From fitness landscape to crossover operator choice. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 815–822. ACM (2014)
25. Min, K., Kao, M.-Y., Zhu, H.: The closest pair problem under the hamming metric. In: Ngo, H.Q. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 205–214. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02882-3_21