Information Sciences 424 (2018) 250–272

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Epigenetic algorithms: A New way of building GAs based on epigenetics

Daniel H. Stolfi*, Enrique Alba

Departamento de Lenguajes y Ciencias de la Computación, University of Malaga, Spain

ARTICLE INFO

Article history:
Received 21 June 2017
Revised 14 September 2017
Accepted 3 October 2017
Available online 5 October 2017

Keywords:
Evolutionary algorithm
Epigenetics
MKP
Bio-inspiration

ABSTRACT

This article presents a new set of ideas on how to build bio-inspired algorithms based on the new field of epigenetics. By analyzing this domain and extracting working computational ideas we want to offer a set of tools for the future creation of representations, operators, and search techniques that can competitively solve complex problems. To illustrate this, we describe an epiGenetic Algorithm, analyze its behavior and solve a set of instances of the multidimensional knapsack problem. Since we are in some measure opening a new line of research, we include a description of epigenetics and computational search, show their working principles and show an example algorithm solving a real problem. Our aim is to offer ideas as well as put them to work, to show that they are actually competitive, not just a nice new inspiration.

1. Introduction

In a letter to T.H. Huxley (Dec. 28th 1859), C. Darwin wrote that "Science is so narrow a field, it is clear there ought to be only one cock of the walk!". He was referring to his previous competitor, R. Owen, and was obviously talking about himself as the new leader in biology after the recent publication of *The Origin of Species*. The basis for the (now traditional) conception of inheritance, genomes and the evolution of the fittest were then widely accepted, and swept aside other theories and beliefs like a huge hurricane.

Darwin, and many others before (e.g., Owen) and after him (e.g., Weismann), contributed to dismissing some of the ideas of de Lamark [41]. In fact, several of these ideas were clearly, correctly discredited, such as the linear concept of evolution (known as *recapitulation*). The inheritance of acquired characters from Lamarck was also dismissed at this moment until now [54]. *Epigenetics* (relating heritable traits that respond to the environment and cannot be explained by changes in DNA sequences) is here to stay, after so much evidence of its existence and so many applications in biology, pharmacology and medicine [23,25,53].

It seems that we *have to* consider a second "cock of the walk" for a better understanding of nature, and that is what we do in this article: we combine the ideas of Darwin and Lamarck into one single computational algorithm, obviously within the niche of research into nature-inspired algorithms. In addition to a nice, appealing inspiration for future work, we of course aim for a competitive evaluation of the resulting techniques, to prove that not only are epigenetic algorithms new as a class (not a renaming of an existing algorithm!), but also that they are useful for solving combinatorial and other kinds of problems in modern research.

* Corresponding author.
E-mail addresses: dhstolfi@lcc.uma.es (D.H. Stolfi), eat@lcc.uma.es (E. Alba).

https://doi.org/10.1016/j.ins.2017.10.005
0020-0255/© 2017 Elsevier Inc. All rights reserved.

Metaheuristics for combinatorial optimization problems [12] are frequently inspired by natural processes such as Darwins' theory of evolution: *evolutionary algorithms* are today a classic example [6]. They are used to solve highly complex real-world problems. Usually, they start with a set of initial candidate solutions and iteratively generate new ones in a chain of increasingly fitted populations towards the optimum of the problem. Their non-deterministic guided and intelligent search balances the exploration of the search space and exploits its more promising regions, to hopefully find the optimal solution to the problem being solved.

As mentioned, evolutionary algorithms (EAs) are stochastic search methods which solve a wide range of combinatorial (and continuous) problems very efficiently and effectively, especially those that cannot be solved, using classic optimization tools: guided or efficient exhaustive enumeration like in A*, mathematical programming, branch-and-bound and dynamic programming, to name a few. The multiple search at the same time, the representation of the solutions in any convenient manner for the search, the absence of requirements of continuity/derivability of the function being optimized, and many other advantages (like dealing with any type of constraints, mixed variable domains, large dimensionality, etc.) make EAs a great tool in modern research.

Some algorithms work better with specific types of problems and perform worse over others [62]. Consequently, this first study is targeted to develop a new bio-inspired algorithm family based on epigenetics, that can be later adapted to different problems. This is possible not just in the traditional manner (as with EAs), but also by using a variety of epigenetic mechanisms that we successfully translate from biology to computer optimization.

In summary, this article studies the diverse epigenetic mechanisms controlled by specific DNA methylation as a method of modifying DNA expression, that may be reversible and inheritable. From this study, we define a methodology to generate EAs that efficiently solve many different problems by using epigenetic concepts on information representation, such as histones, nucleosomes, and chromatin, and epigenetic operations such as genomic imprinting, reprogramming, paramutation, position effect, X-inactivation, bookmarking, and gene silencing. As initially stated, we move from the pure inspiration to the traditional optimization analysis, to ensure that this idea is noteworthy to know and usable in the future of bio-inspired computing.

The rest of the paper is organized as follows. In the next section, we review the state of the art related to our work. In Section 3 we introduce the epigenetic theory and in Section 4 we enumerate the Epigenetic Operators. Section 5 presents the epiGenetic Algorithm and its operators. Section 6 contains the description of the Multidimensional Knapsack Problem and Section 7 describes the competitors algorithms. After that, we comment on the parameterization of the algorithms in Section 8, discuss the preliminary evaluation of our algorithm in Section 9, and finally, conclusions and future work are given in Section 10.

2. Background

Epigenetics have inspired several articles in computer science and bioinformatics in the last decade [1], we comment on some of them here. We will show that, even if the term has been mentioned in the past, nothing similar to a methodology for building efficient algorithms has been developed in these few papers.

In [45] the authors describe the optimization strategies that bio-molecules utilize and propose an intragenerational epigenetic algorithm based on them. The authors also present an agent-based cell modeling and simulation environment, called SwarmCell, whose model has been built as an autopoietic system that represents a minimal biological cell. Then, they implement the epigenetic strategies in the model to better clarify the disease mechanisms at the sub cellular level. This strategy's proposed use is to study cancer development patterns in different cell types which have been differentiated by various trans-generational epigenetic mechanisms. The authors state that their epigenetic algorithm can prove to be a fundamental extension to existing evolutionary systems and swarm intelligence models. They discuss improving problem-solving capabilities by implementing epigenetic strategies in their SwarmCell model. Finally, for future work, they intend to develop a trans-generational epigenetic algorithm to demonstrate how the internal organization of a system can pass on its traits to the next generation. Although epigenetic techniques are also proposed in their work, our study focuses on a new set of algorithms based on natural evolution rather than autopoietic systems and swarm models.

An epigenetic approach in artificial life (ALife) is presented in [57] where the model proposed (EpiAL) uses a dynamic environment to influence the regulation of organisms and the possible inheritance of epigenetic acquired marks. The objective of the EpiAL model is to study the plausibility for the existence of epigenetic phenomena and its relevance to an evolutionary system, from an ALife point of view. Therefore, each agent is able to modify its phenotypic expression due to environment conditions, pass on epigenetic marks between generations enabling the existence of acquired traits which can be transmitted through consecutive generations of agents. The experimentation performed with the EpiAL model in order to study the mechanisms that influence the evolution of the agents shows that the epigenetic populations are able to regulate themselves for dynamic conditions, while the non-epigenetic populations find it hard to prosper in dynamic environments. The authors plan a future development of the model with a focus on both, biological knowledge (developmental biology) and possible problem solving techniques (dynamic environments). This epigenetic approach is focused on the evolution of epigenetic agents to gain more knowledge about this field while we propose an algorithm and solve problems with it.

The authors in [59] incorporate an explicitly controlled gene expression through histone modification in strongly-typed genetic programming (STGP) and call it, epigenetic programming. They propose a double cell representation of the simulated individuals represented by their respective chromatin structures. The authors view their proposed approach of epigenetic programming as a form of epigenetic learning (EL) incorporated into genetic programming via the beneficial modifications of histone code, which take place within the life cycle of evolved simulated organisms. They achieve phenotypic diversity of genotypically similar individuals by using the cumulative effect of polyphenism. They preserve individuals from the destructive effects of crossover by silencing genotypic combinations and explicitly activate them when it is more beneficial. Based on the empirically obtained results, the authors indicate that epigenesis contributes to a 2.1-fold improvement in the computational effort of genetic programming when it is used to evolve the social behavior of predator agents in the predator-prey pursuit problem.

Although these three approaches use concepts related to epigenetic theory, to the best of our knowledge, none of them have proposed an epigenetic algorithm to solve problems related to combinatorial optimization as we do in this article. We believed that the epigenetic model, including problem representation and operators, has not been comprehensively described and exploited to build a working search algorithm, especially in those studies concerned with solving complex optimization problems.

Our proposal consists in a methodology to generate evolutionary algorithms based on the epigenetic model which are able to efficiently solve many different problems by using epigenetic mechanisms. Additionally, we compare our proposal to other well-known models and evaluate their numeric results to see whether our results are competitive or even can improve upon theirs. We first present the natural processes explaining epigenetics.

3. Epigenetics from a representation point of view

A DNA molecule consists of two strands coiled around each other forming a double helix. Each strand is composed of *nucleotides* containing *nucleobases* such as guanine (G), adenine (A), thymine (T), and cytosine (C) [24]. DNA is organized into long structures called chromosomes (23 pairs in humans, consisting of approximately 25,000 genes) which are duplicated during cell division. Inside each chromosome, proteins such as *histones* compact and organize DNA to guide the interactions with other proteins, controlling which genes are expressed. DNA molecule carries genetic information that can be passed from one generation to the next [5]. This is the concept that we can find in current EAs, where the chromosome is a vector of symbols representing DNA genes, usually in a haploid manner (although some diploid representations were once proposed in [30,56]).

In contrast to the classic Mendelian inheritance of phenotypic traits, caused by mutations of the DNA sequence, under the natural selection explained by Darwin's theory of evolution, epigenetic changes are long-term alterations in the transcriptional potential of a cell, due to the activation of certain genes, that are not necessarily heritable [2].

Epigenetics is the study of the biological mechanisms which cause longterm alterations in the transcriptional potential of cells (first step of gene expression, in which a particular segment of DNA is copied into RNA) during their development without changing the DNA sequence, i.e. it does not involve mutations of the DNA itself [11]. These alterations can be heritable and, perhaps, not visible in the next generation but in a generation after. The gene expression process might also be modified by environmental factors [55], diet, personal habits, aging, or random changes, which may contribute to the development of abnormal phenotypes [36]. In addition, epigenetic marks between generations can be reset, and the genome reverted to its original state [63]. Epigenetic processes are essential for development and differentiation, but they can also be present in mature humans as well.

In the nucleus of *eukaryotes* (organisms whose cells contain a nucleus enclosed within membranes), DNA is packaged into a smaller volume so that it can fit in the cell. This combination of DNA and proteins is called *chromatin* (Fig. 1), which also prevents DNA damage, strengthens the DNA to allow mitosis (cell duplication where the cell nucleus is separated into two identical sets of chromosomes), and controls gene expression and DNA replication. During the metaphase (the most condensed and coiled stage) the structure of chromatin is optimized for physical strength and manageability, forming a chromosome structure to prevent shear damage to the DNA when the chromosomes are separated. Epigenetic chemical modification of the structural proteins in chromatin also alters the local chromatin structure.

The primary protein components of chromatin are *histones* [11], in which eukaryotic DNA is wrapped to form *nucleosomes* (Fig. 1). Nucleosomes are the fundamental unit into which DNA and histones are packaged. They are the basic components of a chromosome where the DNA helix is wrapped around to form a series of beads compacting the DNA. Each nucleosome consists of eight histones called the histone octamer. Additionally, histones have long tails protruding from the nucleosome, which can be modified by methylation, acetylation, etc.

DNA methylation is an epigenetic factor which is recognized as the main contributor to the stability of gene expression states through mitotic cell division [33] because it establishes a silent chromatin state that modifies nucleosomes [61]. Epigenetic mechanisms constrain expression by adapting regions of the genome to maintain either gene silencing or gene activity [10]. This is achieved through direct chemical modification of the DNA region itself and by the modification of proteins that are closely associated with the location of each gene [36]. Additionally, DNA methylation and histone modification serve as epigenetic marks for active or inactive chromatin, and such epigenetic marks can be heritable [39].

Epigenetic regulation of gene expression can occur when DNA methylation is lost to allow active or inactive genetic states to be potentially reversible. If methylation fails to be maintained during multiple rounds of DNA replication, a passive loss occurs. On the other hand, active demethylation takes place in non-dividing cells and requires enzymatic activities [36].

We include a schema of the epigenetic mechanisms in Fig. 2, and describe them in the following section.

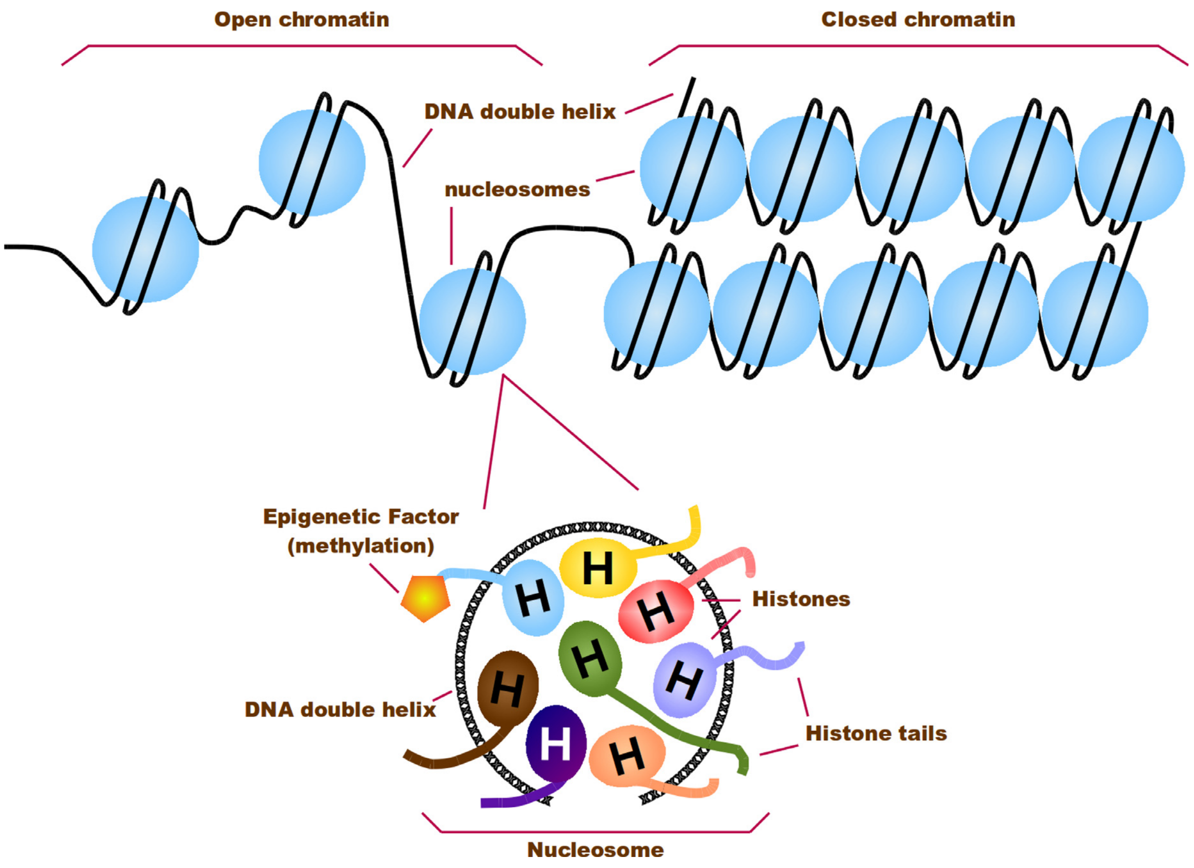

Fig. 1. DNA packaged by the chromatin in eukaryotic cells (top) and an epigenetic factor, i.e. methylation, bound to a histone in a nucleosome (bottom).

4. Epigenetic operators

Epigenetic Mechanisms [2] are the temporal and spatial controllers of gene activity during the development of complex organisms [34]. DNA methylation and histone modification are clear examples of epigenetic mechanisms [51], all of which can affect long-term gene expression, which constitutes the basis for the accurate execution of developmental programs and the maintenance of the cell types over subsequent cell divisions [37].

Epigenetic Mechanisms can be used as operators to modify the solution of a problem represented as a chromosome following the epigenetic methylation rules. Most of the mechanisms use references to the cell's parents to calculate new values for the chromosome as depicted in Fig. 2.

The seven epigenetic mechanisms are analyzed in the following sections.

4.1. Genomic imprinting

Genome Imprinting [16] is a non-Mendelian phenomenon by which a gene expression depends on whether its origin is paternal or maternal [44,61]. Mammals are diploid organisms whose cells have two matched sets of chromosomes, one inherited from the mother and one from the father. Therefore, mammals have two copies of each gene with the same potential to be active in any cell. Genome Imprinting changes this potential by restricting the expression of a gene to one of the parental chromosomes. If an allele inherited from the father is imprinted, only the allele inherited from the mother is expressed, and vice versa.

4.2. Reprogramming

Epigenetic reprogramming [49] is an important aspect of normal mammalian development. Several changes to DNA methylation and histones are imposed on the two parental genomes during cell division, differentiation and other stages of vertebrate development. Many environmental factors, stochastic events, diet, and early experiences may contribute to the variations in the epigenome [4].

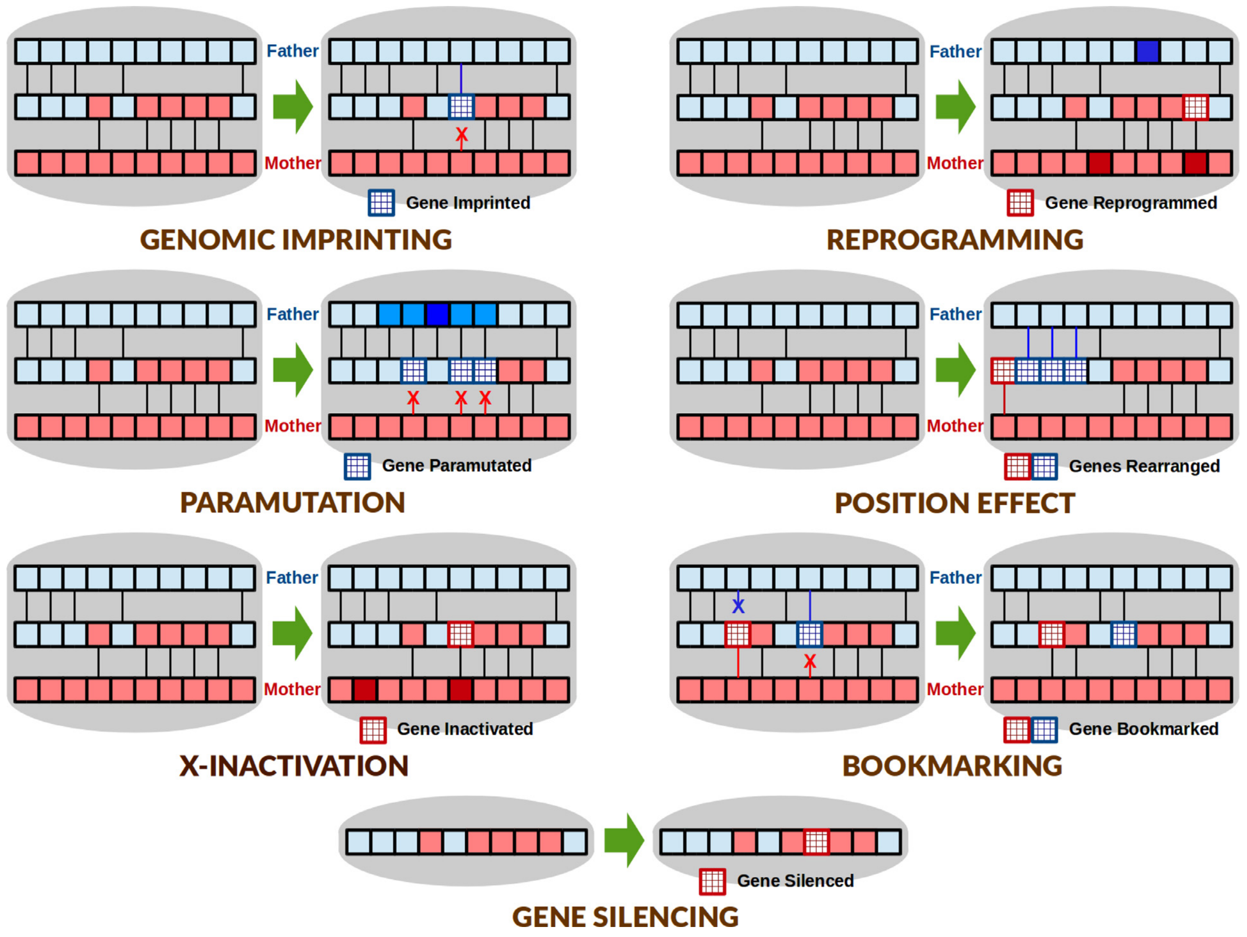

Fig. 2. Schema of each epigenetic mechanism and the modifications made to the cell's DNA thru methylation.

4.3. Paramutation

Paramutation [15] is the epigenetic alteration of one allele induced by the other one in the same location. It occurs when certain alleles impose an epigenetic imprint on the susceptible ones. The paramutated allele could be inherited in traits of later generations even if the gene behind those traits is absent. Paramutation violates Mendel's first law as alleles do not remain unchanged, which differs from the expected classical Mendelian inheritance patterns [21].

4.4. Position effect

Position Effect [9] consists in the juxtaposition of genes with *heterochromatin* (a tightly packed form of DNA) either by rearrangement or by transposition, resulting in a variation of the phenotype to indicate that the gene has been silenced in cells where it is usually active.

4.5. X-Inactivation

Human DNA is packed into 23 pairs of chromosomes (22 pairs of autosomes and one pair of sex chromosomes) of varying size. One chromosome of each pair in inherited from the individual's father and the other from his mother. The sex chromosomes differ between the sexes so that females have two copies of the X chromosome (XX) and males have one X and one copy of the Y chromosome (XY). One of the mechanisms to compensate this difference between members of the same species is switching off genes on one of the female Xs [50]. In some cells it is the paternal X, in others it is the maternal X, but once inactive, all of the clonal descendants of the cell have the same inactive X [35].

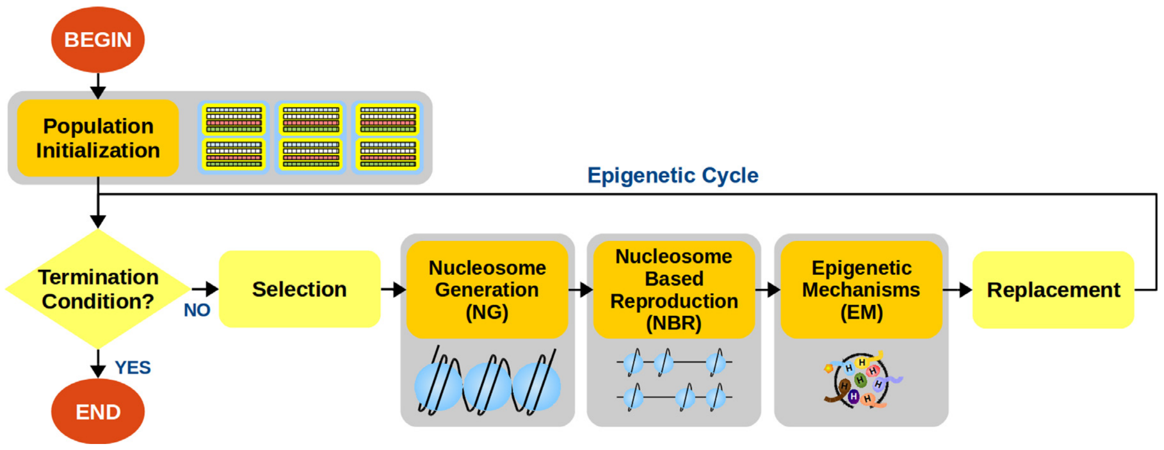
BEGIN
Population Initialization
Epigenetic Cycle
Termination Condition?
NO
YES
END
Selection
Nucleosome Generation (NG)
Nucleosome Based Reproduction (NBR)
Epigenetic Mechanisms (EM)
Replacement

Fig. 3. epiGenetic Algorithm (epiGA). Note that the specially built epigenetic blocks have been highlighted.

4.6. *Bookmarking*

Gene bookmarking is a epigenetic mechanism that controls cell fate and lineage commitment as cells must propagate the gene pattern through mitosis, to daughter cells [52]. It is believed that this pattern of gen activity is somehow marked before mitosis to avoid suffering modifications and let daughter cells know how to reassemble the transcription machinery on the promoters of these genes once mitosis is completed. Bookmarking describes the retention of phenotype-specific transcription factors mitotic chromosomes, allowing the necessary information to be conveyed to progeny cells by inheritable histone marks and DNA methylation [64].

4.7. *Gene silencing*

Gene silencing describes the epigenetic mechanism of gene regulation by “switching off” a gene without using genetic mutation. Transcriptional gene silencing is the result of histone modifications and post-transcriptional gene silencing is the result of the messenger RNA (mRNA) destruction of a particular gene [48]. This epigenetic mechanism plays a central role in the regulation of gene expression, genome stability and is involved in defense against changes in position of a DNA sequence and RNA viruses [46].

5. The epiGenetic Algorithm (epiGA)

Now we have briefly presented the basis of epigenetic chromosomes and operations on them, we start the pure, computational part where we show different ideas to build algorithms based on the previous concepts.

Our novel proposal, the epiGenetic Algorithm (epiGA), consists of a set of strategies, based on evolutionary computation, inspired in nature, especially in epigenetics, with the aim of solving complex combinatorial problems.

The foundation of epiGA is epigenesis. We are interested in how the DNA and histones are collapsed to form nucleosomes, how this affects the gene replication during reproduction, and how the epigenetic mechanisms modify the gene expression thru methylation, all of them in order to build the bio-inspired operators of our algorithm. We think this is a way of building our algorithm that, while different from known models, remains close to a standard GA which will make it easier for other authors to adopt it.

In Fig. 3 we present the block diagram of the epiGA where the epigenetic operators are highlighted. During the *Population Initialization*, new individuals containing cells are created. The *Nucleosome Generation* creates the nucleosome structure where the DNA is collapsed and made inaccessible during reproduction. The *Nucleosome Based Reproduction* operator is where the most promising cells combine with each other following epigenetic rules. Finally, the block called *Epigenetic Mechanisms* is the place in which those mechanisms are applied according to DNA methylation and the surrounding environment.

The pseudocode of the epiGA is described in Algorithm 1. There are a few necessary parameters such as these: N_i: number of individuals; N_c: number of cells; P_e: epigenetic probability; P_n: nucleosome probability; R: nucleosome radius; *Mechanisms*: set of epigenetic mechanisms to be applied; *Environment*: environment rules. Most of these parameters depend on the problem to be solved, consequently their values will be studied in Section 8.

First, the number of steps t, the population $P(0)$, and the auxiliary population $Q(0)$ are initialized. Then, the main loop is executed until the termination condition holds. In the main loop, the auxiliary population Q is filled with individuals from the population P by using the *Selection* operator, as in standard GA. Next, the nucleosome chain is generated for each cell

Algorithm 1 epiGenetic Algorithm.

```
procedure EPIGA($N_i$, $N_c$, $P_e$, $P_n$, $R$, $Mechanisms$, $Environment$)    ▷ $O_{epiGA} = O(n^2)$
    $t \leftarrow 0$
    $P(0) \leftarrow PopulationInitialization(N_i, N_c)$    ▷ P = population
    $Q(0) \leftarrow \emptyset$    ▷ Q = auxiliary population
    while not $TerminationCondition()$ do
        $Q(t) \leftarrow Selection(P(t))$
        $Q(t) \leftarrow NucleosomeGeneration(Q(t), P_n, R)$    ▷ NG
        $Q(t) \leftarrow NucleosomeBasedReproduction(Q(t))$    ▷ NBR
        $Q(t) \leftarrow EpigeneticMechanisms(Q(t), P_e, Mechanisms, Environment)$    ▷ EM
        $P(t+1) \leftarrow Replacement(P(t), Q(t))$
        $t \leftarrow t + 1$
```

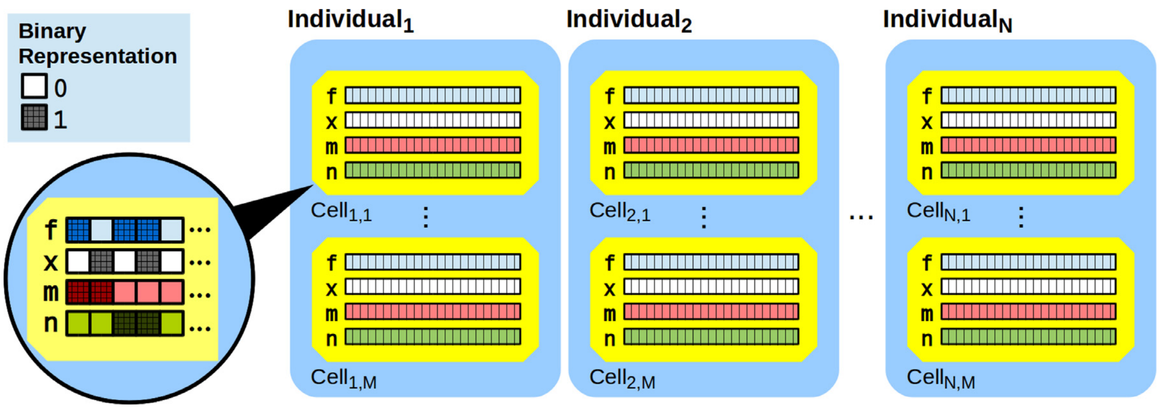

Fig. 4. Population of the epiGenetic Algorithm.

belonging to the individuals in Q. Then, the offspring is obtained after reproduction taking into account the nucleosomes previously generated.

The cells of the offspring in Q are then exposed to the environment while the different epigenetic mechanisms are applied. Then the modified chromosome of each cell is evaluated to obtain their new fitness values. Finally, the new population $P(t+1)$ replaces the current one ($P(t)$) with the individuals of the auxiliary population $Q(t)$ by using the *Replacement* operator. This implies that the individuals of $P(t)$ with the worst fitness values are replaced by the individuals of $Q(t)$ if and only if the new ones have better fitness values (lower values if we are minimizing and vice versa). Each part of the algorithm, including its operators, are explained as follows:

5.1. Population initialization

The initial population could be randomly initialized or could be created by seeding according to the problem knowledge or by a greedy algorithm. In this article we address, without loss of generality, the solution of combinatorial optimization problems represented as a binary vector. Consequently, we present in Fig. 4 the structure of the population for epiGA. Each individual in the population made of N individuals contains M cells which can represent different solutions of the problem. Inside each cell we find four binary vectors of the same size of the chromosome (problem representation). They are the x vector (chromosome) where the solution is encoded, the f and m vectors which contain the chromosomes of the parents of the cell (f stands for father and m for mother), and finally the n vector where the binary mask (nucleosome mask) representing the nucleosome structure is stored.

The pseudocode of the *Population Initialization* is presented in Algorithm 2. The parameters N_i and N_c are the number of individuals and cells that will be in the population. Each new cell is generated by the constructor function, *Cell*, which fills the x vector with a new solution that can be either randomly generated or made especially, depending on the problem's characteristics. After that, the cell is evaluated to obtain its fitness value according to its chromosome and the problem being solved. The result of this operator is the set of N_i individuals, I, containing N_c cells each, which will become the initial population for the algorithm.

Algorithm 2 Population Initialization.

```
function PopulationInitialization($N_i$, $N_c$)                 ▷ $O_{PI} = O_{EV}$
    for all $i \in N_i$ do                                        ▷ $N_i$ individuals
        for all $c \in N_c$ do                                    ▷ $N_c$ cells
            $C(c) \leftarrow Cell()$                              ▷ Constructs a new cell
            $Evaluate(C(c))$
        $I(i) \leftarrow Individual(C)$                           ▷ set of individuals
    return $I$                ▷ set of $N_i$ individuals, each one containing $N_c$ cells
```

5.2. Selection

If the termination condition is not satisfied, the algorithm enters a new iteration whose first step is the *Selection* operator. In this article we have used the well-known binary tournament [29] as the selection operator although a different selection strategy could be used here. As a result, the auxiliary population P' is obtained and the epigenetic cycle of epiGA continues.

5.3. Nucleosome Generation (NG)

The next operation is the *Nucleosome Generation* (NG) which generates a new nucleosome vector as a mask for each cell in the individuals of the population as shown in Algorithm 3.

Algorithm 3 Nucleosome Generation.

```
function NucleosomeGeneration($P$, $P_n$, $R$)                   ▷ $O_{NG} = O(n)$
    for all $i \in P$ do                                          ▷ Each individual i
        for all $c \in i$ do                                      ▷ Each cell c
            $n \leftarrow getNucleosome(c)$
            for all $j \in n$ do
                if $rnd() < P_n$ then                 ▷ Chromosome is probabilistically collapsed
                    $Collapse(n, R)$                              ▷ n[j-R˜–˜j+R] = 1
    return $P$
```

Each position of the nucleosome vector n in each cell is likely to be a center of a nucleosome in the vector. Depending on the nucleosome probability P_n new nucleosomes will be generated, centered in the chosen position j, with a radius R, representing a collapsed DNA region. At the end of this operation all the cells in the population have a new nucleosome mask in n.

5.4. Nucleosome Based Reproduction (NBR)

The *Nucleosome Based Reproduction* (NBR) operator (Algorithm 4 and Fig. 5) uses the nucleosome mask generated by the previous operator in the epigenetic cycle to guide the recombination of the solutions stored in each cell.

First, an empty population P' is initialized. Second, two individuals (i_1 and i_2) are taken from the current working population P. Third, the best cells (c_1 and c_2) from i_1 and i_2 are extracted as well as their chromosomes x_1 and x_2 and nucleosomes n_1 and n_2. Next, a new nucleosome mask N is calculated by carrying out the boolean operation OR between the n_1 and n_2. After that, the contents of the chromosomes are swapped only where the DNA is not collapsed, i.e. where the nucleosome mask value is zero, and stored in two new chromosomes x_1' and x_2'. At the end of the loop, two new cells c_1' and c_2' are created by using the new chromosomes x_1' and x_2', the former ones x_1 and x_2 and the new nucleosome mask N. Note that the former chromosomes are needed to store the father f and mother m in the new cell (Fig. 5), as they could be needed when applying Epigenetic Mechanisms later. Finally, two new individuals i_1' and i_2' are created by replacing the worst cell in them by the new cells c_1' and c_2', respectively. Then i_1' and i_2' are added to the population P'. This operation is especially useful to preserve the diversity of the cells in the individual and thus in the population. At the very end of the code, the resulting population P' is returned to be used by the next operator in the epigenetic cycle of the epiGA.

5.5. Epigenetic Mechanisms (EM)

This operator applies to each cell of the population P (received as a parameter), the epigenetic mechanism specified in the parameter m with a probability Pe and under the effects of the environment rules contained in *Environment*, as shown in Algorithm 5. After that, the new content of the cell, i.e. the x chromosome, is evaluated.

Although we discussed seven different epigenetic mechanisms in Section 4 and showed them in Fig. 2, in this first approach we are only studying the Gene Silencing mechanism. It is very common in the biological domain and allowed us a first computational development.

Algorithm 4 Nucleosome Based Reproduction (NBR).

```
function NucleosomeBasedReproduction($P$)                                  ▷ $O_{NBR} = O(n)$
    $P\prime \leftarrow \emptyset$                                         ▷ Auxiliary population for NBR
    for all $\{i_1, i_2\} \in P$ do                                        ▷ Each two individuals
        $c_1 \leftarrow getBestCell(i_1)$
        Let $c_1, c_2$ be the best cells from $i_1$ and $i_2$             ▷ According to fitness value
        Let $x_1, x_2$ be the solution vectors from $c_1$ and $c_2$
        Let $n_1, n_2$ be the nucleosome vectors from $c_1$ and $c_2$
        $N \leftarrow n_1$ ~OR~ $n_2$                                      ▷ New nucleosome mask
        while $j < size(N)$ do
            if $N(j)$ then                                                 ▷ Collapsed DNA ($N(j) = 1$) zones do not change
                $x_1\prime(j) \leftarrow x_1(j)$
                $x_2\prime(j) \leftarrow x_2(j)$
            else                                                           ▷ Non-collapsed DNA zones do change
                $x_1\prime(j) \leftarrow x_2(j)$
                $x_2\prime(j) \leftarrow x_1(j)$
            $j \leftarrow j + 1$
        $c_1\prime \leftarrow Cell(x_1\prime, x_1, x_2, N)$                ▷ New cells with new parents and nucleosomes
        $c_2\prime \leftarrow Cell(x_2\prime, x_1, x_2, N)$
        $i_1\prime \leftarrow replaceWorst(i_1, c_1\prime)$               ▷ New individuals based on the former ones
        $i_2\prime \leftarrow replaceWorst(i_2, c_2\prime)$               ▷ $O(Nc)$
        $P\prime \leftarrow P\prime \cup \{i_1\prime, i_2\prime\}$
    return $P\prime$
```

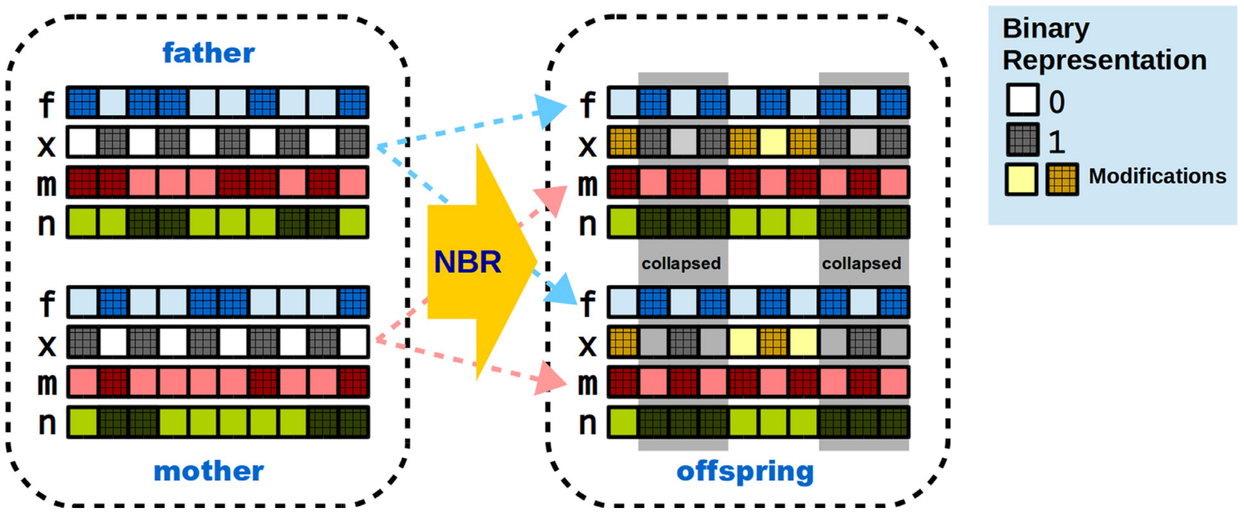

Fig. 5. Nucleosome Based Reproduction (NBR). Both vectors x change only where the DNA is not collapsed (positions of vector n with zeros). New vectors f and m are taken from the parents of the new cell. Offspring's vectors n are calculated by applying the boolean OR to the parents' vector n.

Despite that, possible implementations of the rest of mechanisms are focused on the parents' values for each position of the chromosome. These values will change from one parent to the other in Genomic Imprinting, if the parental ones have changed in Reprograming, or if they are in the neighborhood of some selected parental genes in Paramutation. Additionally, the Position Effect mechanism switches the dependency of the selected genes from one parent to the other, the X-inactivation mechanism inactivates some genes in the mother's chromosome and these changes affect the chromosome of the individual, and finally, Bookmarking keeps some selected genes invariant.

5.5.1. Gene Silencing (GS)

The pseudocode of *Gene Silencing* is presented in Algorithm 6. It receives cell c, epigenetic probability P_e, and the *Environment*, as parameters. As we have mentioned, only collapsed DNA is likely to be changed by methylation. In GS, the probability of methylation is provided by the environment (a vector of probabilities) for each gene (position of vector x).

First, the chromosome x and the nucleosome mask n is obtained from the cell c. Second, each position of the chromosome x is selected and, if the corresponding position in the nucleosome mask n indicates that the DNA is collapsed, the value of

Algorithm 5 Epigenetic Mechanisms (EM).

```
function EPIGENETICMECHANISMS($P$, $Pe$, $m$, $Environment$)        ▷ $O_{EM} = O_{GS}$
    for all $i \in P$ do                                             ▷ Each individual i
        for all $c \in i$ do                                         ▷ Each cell c
            $ApplyMechanisms(m, c, Pe, Environment)$
            $Evaluate(c)$
    return $P$
```

Algorithm 6 Gene Silencing (GS).

```
procedure GENESILENCING($c$, $Pe$, $Environment$)                   ▷ $O_{GS} = O(n)$
    $x \leftarrow getSolution(c)$
    $n \leftarrow getNucleosome(c)$
    while $j < size(x)$ do
        if $n(j)$ then                                               ▷ Only Collapsed DNA methylates
            if $rnd() < P_e$ then                                    ▷ Epigenetic probability
                $x(j) \leftarrow rnd() < Environment(j)$              ▷ Influence of the environment
        $j \leftarrow j + 1$
```

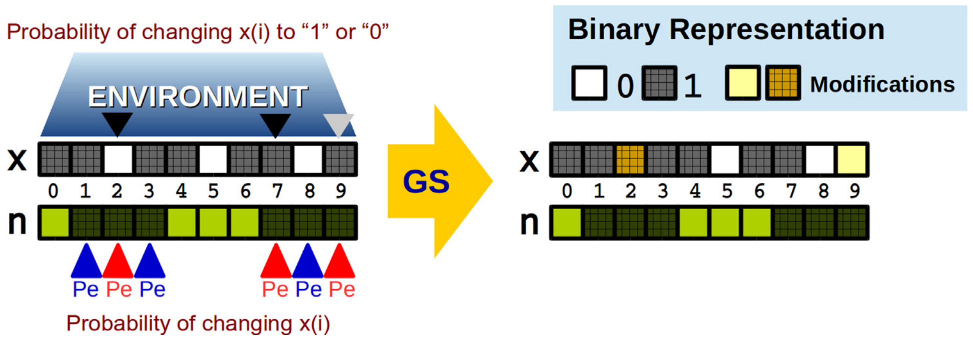

Fig. 6. Gene Silencing (GS). Only collapsed DNA is likely to be methylated. The methylation probability is given by the environment.

$x(j)$ may change according to the probability P_e and the *Environment*. Here we are supposing a binary problem, as a result, each position of x is a binary value, and *Environment* is a vector of probabilities that modifies the likelihood that each value x has of being "1".

By using the nucleosome mask (n), we control which parts of the solution are likely to change, i.e. the ones that were not modified by the reproduction. Moreover, the environment allows us to define a different probability distribution for each position of the solution vector. It could be just a floating point number if we have a binary vector as the solution, or a more complex expression for a solution comprising integers, for example. Note that we can prescind from the environment's influence if we set 0.5 as the probability value for every position of the solution, making them all equiprobable.

In Fig. 6 we present an example of *Gene Silencing* applied to chromosome x, according to the nucleosome mask n. It can be seen that the positions 1, 2, 3, 7, 8, and 9, correspond to a collapsed DNA (vector n). After using the random number generator and comparing the result with P_e, only positions 2, 7, and 9, are candidates for changing their value. Finally, the influence of the environment changes positions 2 and 7 to "1", and position 9 to "0". Note that this is not just a bit flip operator as the value of position 7 has not been changed this time despite having been initially selected.

5.6. Replacement

The last operator in the epigenetic cycle is the *Replacement* operator. Here, we have used an elitist replacement [28] although another replacement operator could be used instead. In doing so, we have selected the new working population in an elitist way, copying the best individuals to it. Note that the concept of best individuals depends on their fitness values and the problem that is being solved, e.g. if we are minimizing, the lower, the better.

Having explained the main structure and operators of the proposed epiGA, we move on to describing the Multidimensional Knapsack Problem which has been selected to test our proposal.

6. Multidimensional Knapsack Problem (MKP)

The Multidimensional Knapsack Problem (MKP) is a well-known NP-Hard combinatorial problem [27] that has been studied for decades since it first appeared in [31,40].

We have chosen this problem because it is a highly complex binary problem and because there are several studies available to be compared to our results. Our goal here is to add experimentation to our proposed algorithm based on epigenesis, test how it performs against the state of the art, and validate the use of the epigenetic operators as a viable way of solving hard combinatorial problems.

Different methods have been used to solve the MKP [47], many of them based on kernel search [3], multi-level search strategy [13], Particle Swam Optimization (PSO) [17], Genetic Algorithms (GA) [19], branch and bound techniques [26], and greedy techniques [42].

The MKP consists of n items and m different knapsacks of capacity $c_i,\ i \in \{1, \ldots, m\}$. Each item $j,\ j \in \{1, \ldots, n\}$, has an associated profit p_j and consumes a quantity w_{ij} from the knapsack i, if the item has been selected through the variable x_j by setting it to 1, otherwise 0. The objective is to maximize the profit of the items in the m knapsacks (Eq. (1)) without exceeding the maximum capacity of each one (c_i), according to the constraints described in Eqs. (2) and (3).

$$\text{Maximize} \quad z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} w_{ij} x_j \le c_i, \quad i = 1, 2, \ldots, m \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n \tag{3}$$

To solve the MKP we have defined the fitness function as presented in Eq. (4).

$$F(\vec{x}) = \begin{cases} \sum_{j=1}^{n} p_j x_j & \text{if } \sum_{j=1}^{n} w_{ij} x_j \le c_i, \forall i \\ -1.0 & \text{otherwise} \end{cases} \tag{4}$$

There the fitness value is the sum of the profits corresponding to the objects included in all the knapsacks ($x_j = 1$) providing they do not exceed the maximum capacity of each one (c_i). If this happens, the result is a negative value, e.g. −1, so that the *Evaluate* function can repair the solution as described in Algorithm 7.

Algorithm 7 Evaluate.

```
function EVALUATE($c$)                                    ▷ $O_{EV} = O(n)$
    $x \leftarrow getSolution(c)$
    $f \leftarrow Fitness(x)$
    while $f < 0$ do
        $i \leftarrow findRndOne(x)$
        $x(i) \leftarrow 0$
        $f \leftarrow Fitness(x)$
    $setSolution(c, x)$
    return $f$
```

First, solution x is obtained from cell c. Second, the fitness value is calculated according to Eq. (4). Third, if the fitness value is less than 0 a random position of x where its value is 1 is changed to 0 (an item is removed from the knapsacks). After that, the fitness value is calculated again and the internal loop is repeated until the fitness value is greater or equal to 0. Finally, the solution in cell c is replaced by the new one and the fitness value is returned. Note that we have preferred to use a simple repairing technique instead of a greedy one [19] or just a penalization term [22].

We have adapted the environment to the MKP so that it represents the probability of including an item in the knapsacks. Instead of being equiprobable, all the available items have a probability bias which depends on the relation between its profit and weight in all the knapsacks (Eq. (5)).

$$Environment(j) = \frac{p_j}{\sum_{i=1}^{m} w_{ij}} \tag{5}$$

We calculate the environment (*Environment*) for all the items at the very beginning of the algorithm, when the characteristics of the available items are known. The values are normalized so that the probabilities *Environment*(j) are between $\frac{1}{3}$ for the item j with worst relation value (Eq. (5)), and $\frac{2}{3}$ for the best one. Although we have tested with other limits, $\frac{1}{3}$ has worked the best.

We have chosen the OR-Library [8] to evaluate epiGA. This is a well-known set of instances of the MKP consisting of problems with $n = 100$, 250, and 500 variables and $m = 5$, 10, and 30 constraints. There are 30 instances of each combination of n and m. We have named them according to the pattern: $n.m\_i$, so that the instance 250.5_1 corresponds to the first instance of MKP problem with 250 variables and 5 constraints. In this approach we have addressed the optimization of 30 instances of the following problems: 100.5, 500.5, 100.10, and 250.10, i.e. 120 instances in total.

In the next section we describe the selected competitors for epiGA.

7. Competitors

Even if our goal is to create a new methodology and contribute new ideas to the domain, we are aware of the advantages of showing from the very beginning that the resulting algorithms could actually work versus published competitors in the literature.

We have chosen several competitors for epiGA, some of which are taken from the state of the art, providing their results are available and can be compared to ours. Specifically, the SACRO-PSO algorithms and Resolution Search + Branch & Bound are included in our comparison. In addition, we have included an exact optimizer (CPLEX), and two well-known metaheuristics for solving combinatorial problems, Simulated Annealing (SA) and a Genetic Algorithm (GA). These five algorithms help us to prove performance for new ideas, even if this article is just devoted to introducing them for future exploitation.

7.1. IBM ILOG CPLEX

IBM ILOG CPLEX[1] is a commercial software developed by ILOG and currently owned by IBM, based on linear programming and the simplex method [43]. We have formalized each working instance as a linear programming problem to be solved by CPLEX as a maximization task (Eq. (6)) subject to the restrictions defined in each instance (Eq. (7)).

$$\text{Maximize} \quad obj = p_0 \cdot x_1 + p_1 \cdot x_2 + \ldots + p_n \cdot x_n \tag{6}$$

$$\text{Subject to} \quad \begin{aligned} c_1 &= w_{1,1} \cdot x_1 + w_{1,2} \cdot x_2 + \ldots + w_{1,n} \cdot x_n \\ c_2 &= w_{2,1} \cdot x_2 + w_{2,2} \cdot x_2 + \ldots + w_{2,n} \cdot x_n \\ &\ldots \\ c_m &= w_{m,1} \cdot x_1 + w_{m,2} \cdot x_2 + \ldots + w_{m,n} \cdot x_n \end{aligned} \tag{7}$$

To make a fair comparison we have restricted the execution of CPLEX (version 12.6.2.0) to one CPU and thread. Furthermore, we have set the relative and absolute MIP gap tolerance (*mipgap* and *absmipgap*) to 0 in order to improve the precision when finding the maximum.

7.2. SACRO-PSO algorithms

In [17] the author presents a novel self-adaptive check and repair operator (SACRO) combined with particle swarm optimization (PSO) to solve the MKP.

SACRO is based on the check and repair operator (CRO) explained in [19]. However, in SACRO the profit/weight utility and profit density are used as alternative pseudo-utility ratios.

In the two resulting algorithms, based on the existing BPSO-TVAC and CBPSO-TVAC [18], the values for all particles are randomly generated and evaluated to obtain the corresponding fitness value. If the constraints of the knapsack are not satisfied, the infeasible solutions are converted into feasible ones, using SACRO. The repair of the infeasible solutions is based on alternative pseudo-utility ratios which varies the approach directions allowing the particles to visit different feasible regions of the search space.

In our study we have used the results of SACRO-BPSO-TVAC and SACRO-CBPSO-TVAC published in [17] corresponding to 100 runs and 20.000 iterations.

7.3. Resolution Search + Branch & Bound

An exact method based on a multi-level search strategy for solving the MKP is proposed in [13].

First, the top level branches are enumerated by using Resolution Search Strategy, in which the authors proposed an improvement of the waning phase in the resolution search principle [20].

Second, the middle level branches are solved by using Branch & Bound [60]. In this stage, the algorithm first searches in the most promising parts of the search tree. In the implementation of the algorithm proposed in [60], done by the same authors, the specific reduced cost propagation was removed to save time.

Third, the lower level branches are enumerated according to a simple Depth First Search enumeration brute force. The branching strategy in this phase consists in fixing the first free variable to 0 and then to 1.

[1] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

7.4. Genetic Algorithm (GA)

In addition to these competitors, we want to compare our results to a well-known implementation of a Genetic Algorithm (GA) [28,32], since our epiGA could easily be seen as the next step in the evolution of algorithms like GA, when epigenetics are used. This is a metaheuristic inspired by nature, specifically by natural selection and genetics, which is used to solve hard optimization problems.

We have designed an elitist, generational GA [7], and used it to solve the selected instances of the MKP after performing a first phase of parameterization and population size studies, in order to achieve a fair comparison afterwards. The pseudocode of the GA is presented in Algorithm 8.

Algorithm 8 Genetic Algorithm (GA).

```
procedure GA($N_i$, $P_c$, $P_m$)
    $t \leftarrow 0$
    $P(0) \leftarrow PopulationInitialization(N_i)$          ▷ P = population
    $Q(0) \leftarrow \emptyset$          ▷ Q = auxiliary population
    while not $TerminationCondition()$ do
        $Q(t) \leftarrow Selection(P(t))$          ▷ Binary tournament
        $Q(t) \leftarrow Crossover(Q(t), P_c)$          ▷ Uniform crossover
        $Q(t) \leftarrow Mutation(Q(t), P_m)$          ▷ Bit flip
        $Evaluation(Q(t))$
        $P(t+1) \leftarrow Replacement(Q(t), P(t))$          ▷ Elitist Replacement
        $t \leftarrow t+1$
```

First, the number of steps t is initialized to 0 and the population $P(0)$ is initialized with random values using the same procedure as in the epiGA. Then, after initializing the auxiliary population $Q(0)$, the main loop is executed while the termination condition is not fulfilled. Inside the main loop, the *Selection* operator is applied to fill the working population $Q(t)$, using binary tournament [29]. Next, the uniform crossover [29] is applied as the *Crossover* operator and after that, as *Mutation* operator, the bit flip mutation [29]. Finally, after the evaluation and reparation of the individuals (same procedure as in epiGA), the new population $P(t+1)$ is obtained by applying the *Replacement* operator in an elitist way, also as in epiGA.

7.5. Simulated Annealing (SA)

Finally, one last competitor is presented in this section. It consists in an implementation of Simulated Annealing (SA) [14,38] a well-known metaheuristic applicable to a wide range of problems.

Simulated Annealing is a probabilistic method used to find an approximate global optimum in a large search space. It is inspired in annealing in metallurgy where a previously heated material is gradually cooled in order to increase its ductility and reduce its hardness. This temperature decrement is interpreted in SA as a reduction of the probability of accepting worse solutions while the search space is being explored.

We have used five parameters in our implementation (Algorithm 9): N is the number of iterations in the same tempera-

Algorithm 9 Simulated Annealing (SA).

```
procedure SA($N$, $T_0$, $T_{\min}$, $P_{\min}$, $P_{\max}$)
    $X \leftarrow GenerateInitialState()$
    repeat
        for $i = 0$ to $N$ do
            $Y \leftarrow Generate(X, T_k)$          ▷ Generates a new solution
            if $Accept(X, Y, T_k)$ then          ▷ New solution acceptance
                $X \leftarrow Y$
        $T_{k+1} \leftarrow Update(T_k)$          ▷ Temperature decrement
        $k \leftarrow k+1$
    until $Termination\tilde{}Condition$
```

ture (internal loop), T_0 is the initial temperature, $T_{\min}$ is the minimum (final) temperature, and $P_{\min}$ and $P_{\max}$ are two extra parameters set up for initially exploring the search space and lately exploiting the best solutions found.

First, the SA algorithm generates an initial solution X as done in GA and epiGA. Second, the main loop begins until the *Termination Condition* is met. Third, the internal loop begins performing N iterations. Each iteration generates a new solution Y (Algorithm 10) which is accepted depending on the current temperature T_k (Eq. (10)). After that, the current temperature T_k is updated following Eq. (8) as used in the Fast Simulated Annealing (FSA) [58]. Finally, if the *Termination Condition* is not

Algorithm 10 Generate function.

```
procedure GENERATE($X$, $T_k$)
    $p \leftarrow m * T_k + h$
    while $j < size(X)$ do
        if $rnd() < p$ then
            $X(j) \leftarrow$ not $X(j)$
```

fulfilled a new iteration begins.

$$T(k) = \frac{T_0}{1+k} \tag{8}$$

Algorithm 10 presents the pseudocode of the function used to generate a new solution. There, the positions (bits, as we are working with binary representations) of the current solution X are visited and their values are flipped depending on probability p. In order to reduce the number of changes made to the solution following the reduction of the temperature, the value of p decreases following a line defined by slope m and intercept value h.

Eq. (9) shows the calculation of both parameters based on the initial temperature T_0, the minimum one $T_{\min}$ and the parameters $P_{\max}$ and $P_{\min}$ which were obtained during the parameterization of the SA we have done (Section 8). By including these parameters, we are tuning the SA to efficiently solve the MKP in order to be a better competitor of epiGA.

$$m = \frac{P_{max} - P_{min}}{T_0 - T_min}; \qquad h = P_{min} - m \tag{9}$$

Finally, the acceptance of a new generated solution Y is defined in Eq. (10) and is known as the Metropolis probability [38]. There, the energy function c is the fitness of the solutions X and Y, which is calculated as in epiGA. Furthermore, we have included the process that fixes the invalid solutions in the fitness calculation to keep the comparison fair.

$$Accept(X, Y, T_k) = \min\{1, e^{-\frac{c(Y)-c(X)}{T_k}}\} \tag{10}$$

8. Parameterization

In this section we address not only the parameterization of the epiGA, but also the GA and SA in order to improve their performance when solving the MKP and foster a fair result comparison later. All the runs performed in this section were limited to max 1,000,000 evaluations and executed by the same hardware.

8.1. epiGA

The epiGA has three main parameters: the epigenetic probability P_e, the nucleosome probability P_n, and the nucleosome radius R. Additionally, we have included in the parameterization two different population sizes.

First, we have performed 30 independent runs of epiGA with a different combination of the parameters in the same instance of MKP (one of the hardest available so that the maximum would not be found during the optimization time). We have tested $P_e, P_n \in \{0.01, 0.02, \ldots, 0.10\}$ and $R \in \{1, 2, \ldots, 10\}$ which accounts for 1000 combinations, i.e. 30,000 runs. On the left of Table 1 we present the 30 best results achieved (according to the Friedman Rank), the standard deviation, the Friedman Rank itself, and the Wilcoxon p-value for the selected combinations of the parameters. There are two solutions which are statistically equivalent: $P_e = 0.01$, $P_n = 0.10$, $R = 1$ and $P_e = 0.01$, $P_n = 0.06$, and $R = 3$. If we compare the existing Wilcoxon p-value between these two combinations (0.951) we can see that it is high enough to allow us to use any of them. Fig. 7(a)–(c) show the parameterization performed and how the fitness evolves for the different combinations of values when we fix the selected ones. It can be seen that the lower the P_e and R, the better, while P_n tends to depend on the value of R for this problem.

Second, we performed 30 independent runs of epiGA optimizing 30 different instances, using two different population sizes: 200 and 400, which amounts to 1800 runs. We have used as parameters the best combination obtained in the previous experiment ($P_e = 0.01$, $P_n = 0.10$, and $R = 1$). On the right of Table 1 we present the results obtained from the parameterization of the population size performed. We can see that epiGA always obtains a better average fitness when using a population (μ) of 400 individuals to solve the MKP. Additionally, we provide the Wilcoxon p-value of each comparison which shows that they are statistically significant.

8.2. GA & SA

The left of Table 2 shows the parameterization of GA for different combinations of its two parameters, the crossover probability Pc, and the mutation probability P_m. We have performed 30 independent runs for values of $P_c \in \{0.1, 0.2, \ldots, 1.0\}$ and $P_m \in \{0.01, 0.02, \ldots, 0.10\}$ which accounts for 100 combinations, i.e. 3000 runs. We can see that the combination of

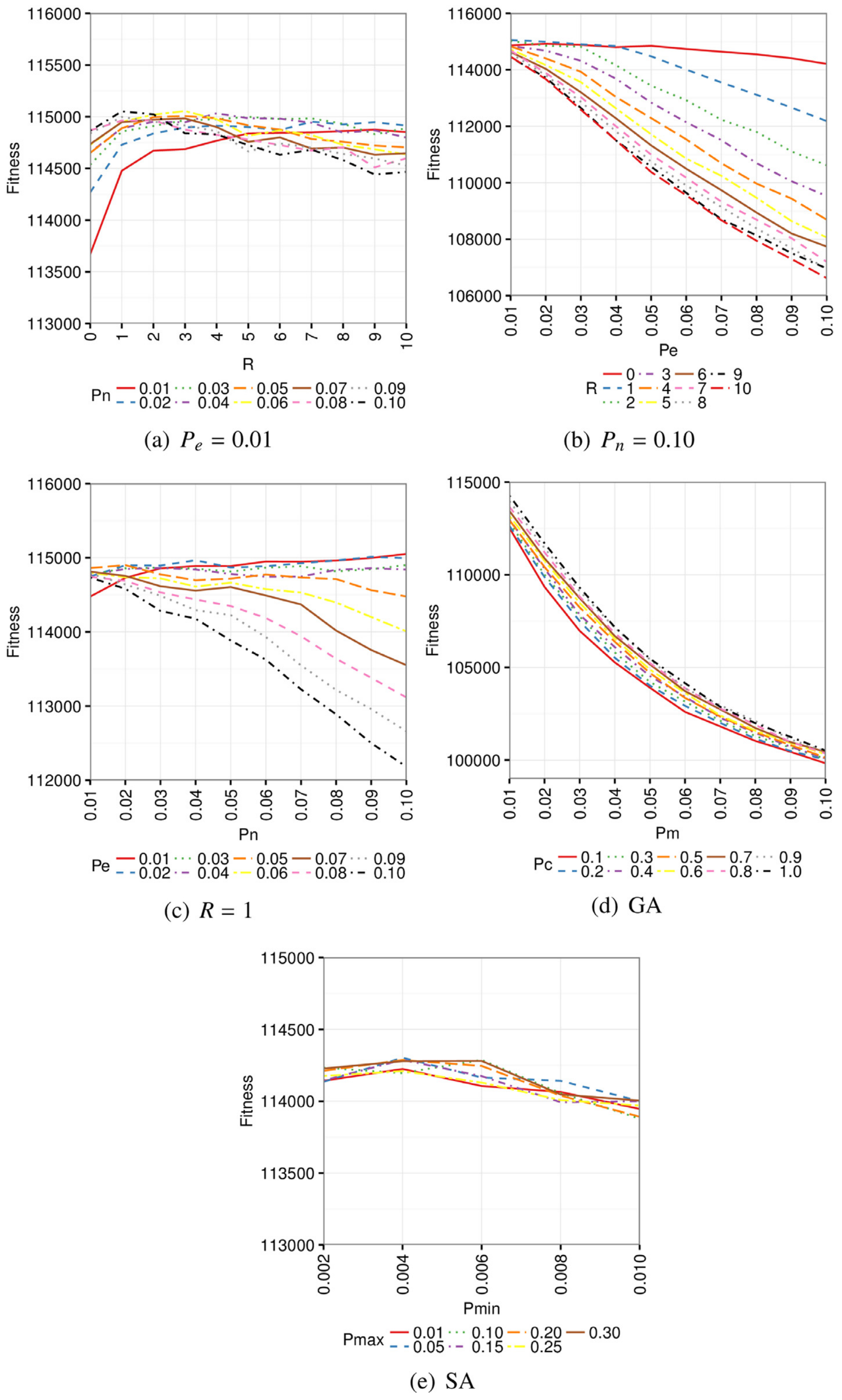

Fig. 7. Fitness variation of epiGA when we set one of the parameters to the best value according to the parameterization (upper row). Additionally, the parameterization of the GA and SA is shown in the lower row.

Table 1
Parameterization of the epiGA. The left of the table shows the parameter tuning (30 best ranked combinations) while the second independent experiment, i.e. population sizes of 200 and 400, is shown on the right.

P_e	P_n	R	Fitness		Friedman	Wilcoxon	Instance	Average Fitness		Wilcoxon
			Average	StdDev	Rank	p-value		200	400	p-value
0.01	0.03	3	114,966.2	123.8	996.75	.028	1	115,337.6	**115,577.9**	.000
0.01	0.03	4	114,974.9	188.6	986.35	.084	2	114,170.8	**114,461.3**	.000
0.01	0.03	5	114,999.2	131.1	1013.95	.162	3	116,055.2	**116,280.3**	.000
0.01	0.03	6	114,977.3	178.9	989.72	.086	4	114,701.4	**114,961.7**	.000
0.01	0.03	7	114,983.4	155.5	997.73	.202	5	115,862.3	**116,061.3**	.000
0.01	0.04	4	115,032.6	149.8	1024.25	.604	6	115,049.3	**115,348.0**	.000
0.01	0.04	5	114,985.5	159.8	1001.35	.150	7	113,463.1	**113,678.5**	.000
0.01	0.04	6	114,981.0	135.1	997.68	.047	8	113,599.1	**113,873.7**	.000
0.01	0.05	2	114,996.2	158.8	1009.07	.459	9	114,761.4	**114,971.8**	.000
0.01	0.05	3	115,006.2	177.5	1006.45	.355	10	116,383.8	**116,616.1**	.000
0.01	0.05	4	114,985.8	173.7	993.80	.241	11	217,648.7	**217,839.6**	.000
0.01	0.06	2	115,020.2	134.7	1019.98	.399	12	214,049.5	**214,272.5**	.000
0.01	0.06	3	**115,054.3**	177.9	1025.62	.951	13	215,404.8	**215,617.2**	.000
0.01	0.07	2	114,971.8	141.0	995.82	.026	14	217,361.0	**217,593.3**	.000
0.01	0.07	3	114,983.1	172.5	993.67	.100	15	215,164.7	**215,382.2**	.000
0.01	0.08	1	114,962.1	149.8	988.78	.053	16	215,319.6	**215,489.4**	.000
0.01	0.08	2	114,962.7	185.1	985.00	.074	17	215,459.2	**215,676.3**	.000
0.01	0.09	1	114,997.8	169.2	1004.78	.399	18	215,950.6	**216,179.8**	.000
0.01	0.10	1	115,051.6	154.6	**1028.90**	–	19	216,871.6	**217,073.0**	.000
0.01	0.10	2	115,023.2	141.2	1020.02	.497	20	214,209.3	**214,448.2**	.000
0.02	0.02	5	114,976.3	180.2	992.13	.206	21	301,207.0	**301,384.7**	.000
0.02	0.04	1	114,966.7	169.3	986.25	.074	22	299,644.1	**299,811.3**	.000
0.02	0.04	2	115,008.3	211.7	1001.72	.376	23	304,751.7	**304,876.9**	.000
0.02	0.04	3	114,960.1	136.0	989.23	.048	24	301,554.4	**301,728.8**	.000
0.02	0.05	3	114,993.5	170.3	998.97	.171	25	304,021.4	**304,219.3**	.000
0.02	0.06	2	114,998.1	119.3	1013.88	.109	26	296,549.0	**296,755.2**	.000
0.02	0.07	2	114,975.0	184.7	988.93	.079	27	302,981.1	**303,122.5**	.000
0.02	0.08	1	114,964.3	170.3	984.52	.018	28	306,585.8	**306,771.8**	.000
0.02	0.09	1	115,012.1	138.0	1016.67	.258	29	302,782.0	**302,947.3**	.000
0.02	0.10	1	114,994.9	193.4	997.33	.202	30	300,114.7	**300,302.7**	.000

$P_c = 1.0\}$ and $P_m = 0.01$ is clearly the best for solving the MKP with a confidence interval greater than 99% according to the Wilcoxon p-value calculated.

Furthermore, we have tested two population sizes as in epiGA by performing 30 independent runs on the 30 different instances as in epiGA (900 runs). It can be seen in the center columns of Table 2 that the average fitness values are always better when using a population of 400 individuals (as in epiGA) and that the results are statistically significant for all the instances (above 97%). The parameterization of GA can be seen in Fig. 7(d) where the fitness values improve as the mutation probability decreases and the crossover probability increases.

SA has just two parameters to tune as this type of algorithm does not include a population. These parameters are used for adjusting the decrement rate of the probability of accepting a solution. We have performed 30 independent runs for 35 combinations of $P_{\min} \in \{0.002, 0.004, \ldots, 0.010\}$ and $P_{\max} \in \{0.01, 0.05, \ldots, 0.30\}$ which accounts for 1050 runs. Table 2 (right) presents the 30 best ranked parameter combinations for SA. We can see that the combination $P_{\min} = 0.004$ and $P_{\max} = 0.05$, presents the best average fitness and is also the best ranked one. There are also other best ranked combinations that might be used as well, as they present a p-value high enough to be considered equivalent.

The parameterization of SA can be seen in Fig. 7(e). It is noticeable that the resulting fitness values do not show a big sensitivity to the two parameters (i.e. very robust behavior).

Table 3 presents the best values for the parameters of epiGA, GA, and SA, obtained after the parameterization described.

9. Evaluating epiGA

In this section we evaluate epiGA on 120 instances (four different MKP types) and compare its results to the selected competitors when possible. We wished to know if epiGA is capable of solving these instances and also to compare how competitive it could be. Finally, we address a convergence analysis where we compare the behavior of the epiGA, GA, and SA, when solving each problem.

The instances of the MKP are of four different types obtained from the OR-Library [8]: 100 variables and 5 constraints (100.5), 500 variables and 5 constraints (500.5), 100 variables and 10 constraints (100.10), 250 variables and 10 constraints (250.10).

Table 2
Parameterization of the GA and SA (30 best ranked combinations). We tested different values for the parameters for GA (P_m and P_c) and two population sizes ($\mu = 200$ and $\mu = 400$). Additionally, we tested two parameters for SA ($P_{\min}$ and $P_{\max}$). Note that we provide the Friedman Rank and the Wilcoxon p-value as well.

GA									SA				
P_c	P_m	Fitness	Rank	p-value	Ins.	Average Fitness		p-value	$P_{\min}$	$P_{\max}$	Fitness	Rank	p-value
		(Avg)				200	400				(Avg)		
0.10	0.01	112,528.6	92.10	.000	1	115,294.8	**115,416.2**	.001	0.002	0.01	114,142.3	19.12	.054
0.10	0.02	109,344.5	80.13	.000	2	114,157.9	**114,297.0**	.000	0.002	0.05	114,135.3	18.37	.044
0.20	0.01	112,638.2	92.92	.000	3	115,971.2	**116,174.0**	.000	0.002	0.10	114,233.3	20.43	.365
0.20	0.02	109,880.0	82.27	.000	4	114,621.9	**114,795.4**	.000	0.002	0.15	114,144.7	18.83	.049
0.20	0.03	107,497.9	71.93	.000	5	115,801.9	**115,956.8**	.000	0.002	0.20	114,211.8	20.33	.225
0.30	0.01	112,826.4	93.57	.000	6	114,968.7	**115,165.2**	.000	0.002	0.25	114,175.8	19.13	.064
0.30	0.02	110,002.7	83.23	.000	7	113,429.9	**113,543.0**	.001	0.002	0.30	114,227.5	20.70	.510
0.30	0.03	107,749.3	72.90	.000	8	113,546.7	**113,712.4**	.000	0.004	0.01	114,224.5	19.83	.382
0.40	0.01	112,990.6	94.27	.000	9	114,625.8	**114,813.7**	.000	0.004	0.05	114,302.5	**23.40**	–
0.40	0.02	110,328.6	84.40	.000	10	116,323.1	**116,493.3**	.000	0.004	0.10	114,195.8	19.33	.285
0.40	0.03	107,845.1	73.37	.000	11	217,599.7	**217,690.0**	.001	0.004	0.15	114,287.9	21.87	.902
0.50	0.01	112,950.9	94.00	.000	12	213,993.6	**214,164.6**	.000	0.004	0.20	114,286.8	22.72	.537
0.50	0.02	110,414.8	84.47	.000	13	215,408.3	**215,526.0**	.000	0.004	0.25	114,209.7	19.50	.202
0.50	0.03	108,239.4	75.00	.000	14	217,298.4	**217,488.6**	.000	0.004	0.30	114,279.0	23.03	.781
0.60	0.01	113,173.8	95.12	.000	15	215,107.7	**215,289.1**	.000	0.006	0.01	114,105.9	16.80	.014
0.60	0.02	110,753.0	85.97	.000	16	215,241.5	**215,359.6**	.000	0.006	0.05	114,164.9	18.63	.088
0.60	0.03	108,481.9	76.53	.000	17	215,481.5	**215,551.3**	.013	0.006	0.10	114,287.8	22.33	.491
0.70	0.01	113,448.7	96.80	.000	18	215,928.1	**216,048.2**	.000	0.006	0.15	114,174.7	19.77	.102
0.70	0.02	110,886.3	86.67	.000	19	216,825.4	**216,950.4**	.000	0.006	0.20	114,245.7	21.08	.572
0.70	0.03	108,739.9	77.50	.000	20	214,177.3	**214,289.6**	.001	0.006	0.25	114,129.7	18.23	.086
0.80	0.01	113,668.2	97.60	.000	21	301,204.8	**301,317.8**	.000	0.006	0.30	114,280.5	20.97	.789
0.80	0.02	111,209.3	87.70	.000	22	299,656.5	**299,738.7**	.000	0.008	0.01	114,064.6	15.50	.006
0.80	0.03	108,817.1	77.80	.000	23	304,702.9	**304,774.6**	.002	0.008	0.05	114,142.1	18.75	.040
0.90	0.01	113,974.0	98.70	.000	24	301,527.0	**301,632.3**	.000	0.008	0.10	114,056.2	15.00	.003
0.90	0.02	111,435.8	88.73	.000	25	303,996.0	**304,099.6**	.000	0.008	0.15	113,993.2	15.32	.013
0.90	0.03	109,108.8	79.10	.000	26	296,499.3	**296,625.5**	.000	0.008	0.20	114,039.1	14.98	.001
1.00	0.01	114,278.7	**99.77**	–	27	302,960.5	**303,007.4**	.024	0.008	0.25	114,008.0	15.27	.000
1.00	0.02	111,697.3	89.60	.000	28	306,571.6	**306,673.2**	.000	0.008	0.30	114,047.1	16.73	.015
1.00	0.03	109,307.8	80.13	.000	29	302,778.7	**302,860.5**	.001	0.010	0.15	114,000.6	14.90	.001
1.00	0.04	107,172.6	69.87	.000	30	300,122.5	**300,206.0**	.000	0.010	0.25	113,970.0	14.67	.001

Table 3
Configuration of epiGA, GA, and SA.

epiGA	$P_e = 0.01$	$N_i = 400$	GA	$P_c = 1.00$	SA	$P_{\min} = 0.004$
	$P_n = 0.10$	$N_c = 1$		$P_m = 0.01$		$P_{\max} = 0.05$
	$R = 1$			$\mu = 400$		

In our experimentation we conducted one run of the CPLEX algorithm (it is deterministic) and 30 independent runs of GA, SA, and epiGA, per instance and problem, which amounts to 10,920 runs. Additionally, we have included the results from two SACRO-PSO algorithms [17] (SACRO-BT and SACRO-CBT) and Resolution Search + Branch & Bound [13] (RS + B&B) for the instances in which they were available.

We wished to test if epiGA was able to solve the different instances but also to know if its results were competitive not only against the standard GA and SA, but also against the state-of-the-art algorithms included in our study. We have replicated the results already published according to the experimentation done on SACRO-PSO and RS + B&B algorithms so that we have more data to compare. The author of the SACRO-PSO algorithms carried out 100 runs, optimizing each instance for 20,000 iterations. RS + B&B, like CPLEX, needed just one run. In addition, we set up the conditions for CPLEX, GA, SA, and epiGA, equally, i.e. one execution core and thread, 2 Gigabytes of RAM, and one hour as maximum execution time.

In the following tables we show the results obtained from our experiments consisting of the best fitness (profit) found by the algorithms and also whether they are better or worse compared to the epiGA's results (negative percentages mean worse values, and vice versa).

Table 4 shows the results of the optimization of 30 instances of the MKP with 100 variables and 5 constraints done by CPLEX, SACRO-BT, SACRO-CBT, GA, SA, and epiGA.

We can see that CPLEX, GA, and epiGA have reached the best profit over the 30 instances. In fact, those are the best-known values for these instances. SACRO based algorithms, in turn, have found the best result only in roughly 50% of

Table 4
Accuracy of the algorithms on 30 instances of the 100.5 MKP (100 variables and 5 constraints).

Instance	CPLEX		SACRO-BT		SACRO-CBT		GA		SA		epiGA
	Best	%	Best	%	Best	%	Best	%	Best	%	Best
100.5_1	**24,381**	0.00%	24,343	−0.16%	24,343	−0.16%	**24,381**	0.00%	**24,381**	0.00%	**24,381**
100.5_2	**24,274**	0.00%	**24,274**	0.00%	**24,274**	0.00%	**24,274**	0.00%	**24,274**	0.00%	**24,274**
100.5_3	**23,551**	0.00%	23,538	−0.06%	23,538	−0.06%	**23,551**	0.00%	23,538	−0.06%	**23,551**
100.5_4	**23,534**	0.00%	23,527	−0.03%	23,527	−0.03%	**23,534**	0.00%	23,527	−0.03%	**23,534**
100.5_5	**23,991**	0.00%	**23,991**	0.00%	23,966	−0.10%	**23,991**	0.00%	23,966	−0.10%	**23,991**
100.5_6	**24,613**	0.00%	24,601	−0.05%	24,601	−0.05%	**24,613**	0.00%	24,601	−0.05%	**24,613**
100.5_7	**25,591**	0.00%	**25,591**	0.00%	**25,591**	0.00%	**25,591**	0.00%	**25,591**	0.00%	**25,591**
100.5_8	**23,410**	0.00%	**23,410**	0.00%	**23,410**	0.00%	**23,410**	0.00%	**23,410**	0.00%	**23,410**
100.5_9	**24,216**	0.00%	24,204	−0.05%	**24,216**	0.00%	**24,216**	0.00%	**24,216**	0.00%	**24,216**
100.5_10	**24,411**	0.00%	24,399	−0.05%	**24,411**	0.00%	**24,411**	0.00%	24,399	−0.05%	**24,411**
100.5_11	**42,757**	0.00%	42,705	−0.12%	42,705	−0.12%	**42,757**	0.00%	**42,757**	0.00%	**42,757**
100.5_12	**42,545**	0.00%	42,494	−0.12%	42,471	−0.17%	**42,545**	0.00%	42,510	−0.08%	**42,545**
100.5_13	**41,968**	0.00%	41,959	−0.02%	41,959	−0.02%	**41,968**	0.00%	41,946	−0.05%	**41,968**
100.5_14	**45,090**	0.00%	**45,090**	0.00%	**45,090**	0.00%	**45,090**	0.00%	**45,090**	0.00%	**45,090**
100.5_15	**42,218**	0.00%	**42,218**	0.00%	**42,218**	0.00%	**42,218**	0.00%	42,192	−0.06%	**42,218**
100.5_16	**42,927**	0.00%	**42,927**	0.00%	**42,927**	0.00%	**42,927**	0.00%	42,886	−0.10%	**42,927**
100.5_17	**42,009**	0.00%	**42,009**	0.00%	**42,009**	0.00%	**42,009**	0.00%	**42,009**	0.00%	**42,009**
100.5_18	**45,020**	0.00%	45,010	−0.02%	**45,020**	0.00%	**45,020**	0.00%	45,000	−0.04%	**45,020**
100.5_19	**43,441**	0.00%	**43,441**	0.00%	43,381	−0.14%	**43,441**	0.00%	**43,441**	0.00%	**43,441**
100.5_20	**44,554**	0.00%	**44,554**	0.00%	44,529	−0.06%	**44,554**	0.00%	**44,554**	0.00%	**44,554**
100.5_21	**59,822**	0.00%	**59,822**	0.00%	**59,822**	0.00%	**59,822**	0.00%	59,799	−0.04%	**59,822**
100.5_22	**62,081**	0.00%	**62,081**	0.00%	**62,081**	0.00%	**62,081**	0.00%	**62,081**	0.00%	**62,081**
100.5_23	**59,802**	0.00%	**59,802**	0.00%	59,754	−0.08%	**59,802**	0.00%	**59,802**	0.00%	**59,802**
100.5_24	**60,479**	0.00%	60,478	0.00%	60,478	0.00%	**60,479**	0.00%	**60,479**	0.00%	**60,479**
100.5_25	**61,091**	0.00%	61,055	−0.06%	61,079	−0.02%	**61,091**	0.00%	61,079	−0.02%	**61,091**
100.5_26	**58,959**	0.00%	**58,959**	0.00%	58,937	−0.04%	**58,959**	0.00%	**58,959**	0.00%	**58,959**
100.5_27	**61,538**	0.00%	**61,538**	0.00%	**61,538**	0.00%	**61,538**	0.00%	**61,538**	0.00%	**61,538**
100.5_28	**61,520**	0.00%	61,489	−0.05%	**61,520**	0.00%	**61,520**	0.00%	**61,520**	0.00%	**61,520**
100.5_29	**59,453**	0.00%	**59,453**	0.00%	**59,453**	0.00%	**59,453**	0.00%	**59,453**	0.00%	**59,453**
100.5_30	**59,965**	0.00%	59,960	−0.01%	59,960	−0.01%	**59,965**	0.00%	**59,965**	0.00%	**59,965**
Average:	**42,640.4**	0.00%	42,630.7	−0.02%	42,626.9	−0.03%	**42,640.4**	0.00%	42,632.1	−0.02%	**42,640.4**

the instances (16 SACRO-BT and 15 SACRO-CBT), and SA only in 18 of them. It is good to see that the idea of epigenetics, put to work in an algorithm is able to beat published results and, in this case, completely accurately for all 30 instances.

Table 5 shows the results of the optimization of 30 instances of the MKP with 500 variables and 5 constraints for all the algorithms

In this second set of larger instances, CPLEX and RS + B&B have found the best profit for all the 30 instances. Our proposal, epiGA, has found the best values for nine instances and its average results are almost the same than CPLEX and RS + B&B (lower than 0.01% on average, 0.04% max). GA, SA, and SACRO algorithms have found values which are far worse than the RS + B&B, never achieving the highest values, although GA's results are 0.02% under epiGA on average. We can see that values for the 15th instance of SACRO algorithms are missing in the original paper [17].

Table 6 shows the results of the optimization of 30 instances of the MKP with 100 variables and 10 constraints by CPLEX, SACRO-BT, SACRO-CBT, GA, SA, and epiGA.

It can be seen that CPLEX, GA, and epiGA have found the best results for all the instances of 100.10 MKP. SACRO-BT has found 18 optimums and SACRO-CBT 17 of them, while SA has only found the optimum for 7 instances. This again represents a nice endorsement of the recently born epiGA, as we are more convinced that it is not just a new nice inspiration, but an accurate and efficient technique.

Our last experiment consists in the optimization of 30 instances of MKP with 250 variables and 10 constraints. The results achieved by CPLEX, RS + B&B, GA, SA, and epiGA are presented in Table 7.

We can see that again RS + B&B has achieved the best results, followed by CPLEX, failing to achieve them in only two instances (just because of the time restriction, with enough time the optimum will of course appear). Additionally, epiGA has reached the best results in 17 instances and its average best profit is otherwise just 0.01% around RS + B&B and CPLEX. On the other hand, GA found the optimums in only 11 instances while SA found none, actually it presents the worst values (0.3% below the epiGA ones). SACRO's values are not reported here because its authors have not experimented with these more complex instances.

Table 8 presents a summary of the results achieved compared with our competitors including not only hit rates, but also execution times.

We can see that the average execution time was 241 s (528 max.) for CPLEX, 5 s for GA (34 max.), 240 s for SA (799 max.) and 3 s for epiGA (22 max.) when optimizing the 30 instances of 100.5_x. It was a good finding to see our epiGA

Table 5
Accuracy of the algorithms on 30 instances of the 500.5 MKP (500 variables and 5 constraints).

Instance	CPLEX		RS + B&B		SACRO-BT		SACRO-CBT		GA		SA		epiGA
	Best	%	Best	%	Best	%	Best	%	Best	%	Best	%	Best
500.5_1	**120,148**	0.03%	**120,148**	0.03%	119,867	−0.20%	120,009	−0.08%	120,050	−0.05%	119,438	−0.56%	120,107
500.5_2	**117,879**	0.00%	**117,879**	0.00%	117,681	−0.17%	117,699	−0.15%	117,843	−0.03%	117,037	−0.71%	**117,879**
500.5_3	**121,131**	0.01%	**121,131**	0.01%	120,951	−0.14%	120,923	−0.16%	121,050	−0.05%	120,218	−0.74%	121,116
500.5_4	**120,804**	0.02%	**120,804**	0.02%	120,450	−0.28%	120,563	−0.18%	120,720	−0.05%	119,875	−0.75%	120,783
500.5_5	**122,319**	0.01%	**122,319**	0.01%	122,037	−0.22%	122,054	−0.20%	122,248	−0.04%	121,538	−0.62%	122,302
500.5_6	**122,024**	0.01%	**122,024**	0.01%	121,918	−0.08%	121,901	−0.09%	121,960	−0.04%	121,241	−0.63%	122,011
500.5_7	**119,127**	0.00%	**119,127**	0.00%	118,771	−0.30%	118,846	−0.24%	119,055	−0.06%	118,371	−0.63%	119,126
500.5_8	**120,568**	0.00%	**120,568**	0.00%	120,364	−0.17%	120,376	−0.16%	120,486	−0.07%	119,744	−0.68%	**120,568**
500.5_9	**121,586**	0.03%	**121,586**	0.03%	121,201	−0.29%	121,185	−0.30%	121,504	−0.04%	120,732	−0.67%	121,552
500.5_10	**120,717**	0.04%	**120,717**	0.04%	120,471	−0.17%	120,453	−0.18%	120,665	−0.01%	119,934	−0.61%	120,674
500.5_11	**218,428**	0.00%	**218,428**	0.00%	218,291	−0.06%	218,269	−0.07%	218,347	−0.03%	217,748	−0.31%	218,419
500.5_12	**221,202**	0.01%	**221,202**	0.01%	221,025	−0.07%	221,007	−0.08%	221,130	−0.03%	220,490	−0.32%	221,188
500.5_13	**217,542**	0.01%	**217,542**	0.01%	217,337	−0.09%	217,398	−0.06%	217,470	−0.02%	216,815	−0.33%	217,524
500.5_14	**223,560**	0.00%	**223,560**	0.00%	223,429	−0.06%	223,450	−0.05%	223,513	−0.02%	222,925	−0.28%	223,558
500.5_15	**218,966**	0.00%	**218,966**	0.00%	N/A	–	N/A	–	218,962	0.00%	218,304	-0.30%	**218,966**
500.5_16	**220,530**	0.00%	**220,530**	0.00%	220,337	−0.09%	220,428	−0.04%	220,490	−0.02%	220,034	−0.22%	220,527
500.5_17	**219,989**	0.00%	**219,989**	0.00%	219,686	−0.14%	219,734	−0.12%	219,982	0.00%	219,349	−0.29%	**219,989**
500.5_18	**218,215**	0.00%	**218,215**	0.00%	218,094	−0.06%	218,096	−0.05%	218,175	−0.02%	217,647	−0.26%	**218,215**
500.5_19	**216,976**	0.00%	**216,976**	0.00%	216,785	−0.09%	216,851	−0.06%	216,967	0.00%	216,316	−0.30%	**216,976**
500.5_20	**219,719**	0.00%	**219,719**	0.00%	219,561	−0.07%	219,549	−0.08%	219,675	−0.02%	219,082	−0.29%	219,717
500.5_21	**295,828**	0.00%	**295,828**	0.00%	295,346	−0.16%	295,309	−0.18%	295,790	−0.01%	295,429	−0.13%	**295,828**
500.5_22	**308,086**	0.00%	**308,086**	0.00%	307,666	−0.14%	307,808	−0.09%	308,054	−0.01%	307,581	−0.16%	308,083
500.5_23	**299,796**	0.00%	**299,796**	0.00%	299,292	−0.17%	299,393	−0.13%	299,788	0.00%	299,298	−0.16%	299,788
500.5_24	**306,480**	0.00%	**306,480**	0.00%	305,915	−0.18%	305,992	−0.16%	306,441	−0.01%	305,932	−0.18%	306,476
500.5_25	**300,342**	0.00%	**300,342**	0.00%	299,810	−0.18%	299,947	−0.13%	300,301	−0.01%	299,957	−0.13%	**300,342**
500.5_26	**302,571**	0.00%	**302,571**	0.00%	302,132	−0.15%	302,156	−0.14%	302,536	−0.01%	302,194	−0.12%	**302,571**
500.5_27	**301,339**	0.00%	**301,339**	0.00%	300,905	−0.14%	300,854	−0.16%	301,305	−0.01%	300,832	−0.16%	301,329
500.5_28	**306,454**	0.01%	**306,454**	0.01%	306,132	−0.10%	306,069	−0.12%	306,433	0.00%	305,948	−0.16%	306,430
500.5_29	**302,828**	0.01%	**302,828**	0.01%	302,436	−0.12%	302,447	−0.12%	302,788	−0.01%	302,318	−0.16%	302,809
500.5_30	**299,910**	0.00%	**299,910**	0.00%	299,456	−0.15%	299,558	−0.12%	299,881	−0.01%	299,510	−0.13%	299,904
Average:	**214,168.8**	0.00%	**214,168.8**	0.00%	213,701.6	−0.21%	213,735.3	−0.20%	214,120.3	−0.02%	213,527.9	−0.29%	214,158.6

Table 6
Accuracy of the algorithms on 30 instances of the 100.10 MKP (100 variables and 10 constraints).

Instance	CPLEX		SACRO-BT		SACRO-CBT		GA		SA		epiGA
	Best	%	Best	%	Best	%	Best	%	Best	%	Best
100.10_1	**23,064**	0.00%	**23,064**	0.00%	**23,064**	0.00%	**23,064**	0.00%	23,055	−0.04%	**23,064**
100.10_2	**22,801**	0.00%	22,739	−0.27%	22,750	−0.22%	**22,801**	0.00%	22,739	−0.27%	**22,801**
100.10_3	**22,131**	0.00%	**22,131**	0.00%	**22,131**	0.00%	**22,131**	0.00%	22,081	−0.23%	**22,131**
100.10_4	**22,772**	0.00%	**22,772**	0.00%	22,717	−0.24%	**22,772**	0.00%	22,650	−0.54%	**22,772**
100.10_5	**22,751**	0.00%	**22,751**	0.00%	**22,751**	0.00%	**22,751**	0.00%	22,697	−0.24%	**22,751**
100.10_6	**22,777**	0.00%	22,725	−0.23%	22,716	−0.27%	**22,777**	0.00%	22,614	−0.72%	**22,777**
100.10_7	**21,875**	0.00%	**21,875**	0.00%	**21,875**	0.00%	**21,875**	0.00%	21,785	−0.41%	**21,875**
100.10_8	**22,635**	0.00%	22,551	−0.37%	22,542	−0.41%	**22,635**	0.00%	22,476	−0.70%	**22,635**
100.10_9	**22,511**	0.00%	**22,511**	0.00%	22,438	−0.32%	**22,511**	0.00%	**22,511**	0.00%	**22,511**
100.10_10	**22,702**	0.00%	**22,702**	0.00%	**22,702**	0.00%	**22,702**	0.00%	22,561	−0.62%	**22,702**
100.10_11	**41,395**	0.00%	**41,395**	0.00%	41,388	−0.02%	**41,395**	0.00%	41,354	−0.10%	**41,395**
100.10_12	**42,344**	0.00%	**42,344**	0.00%	**42,344**	0.00%	**42,344**	0.00%	42,227	−0.28%	**42,344**
100.10_13	**42,401**	0.00%	42,350	−0.12%	42,350	−0.12%	**42,401**	0.00%	42,347	−0.13%	**42,401**
100.10_14	**45,624**	0.00%	45,585	−0.09%	45,511	−0.25%	**45,624**	0.00%	45,479	−0.32%	**45,624**
100.10_15	**41,884**	0.00%	41,799	−0.20%	41,833	−0.12%	**41,884**	0.00%	**41,884**	0.00%	**41,884**
100.10_16	**42,995**	0.00%	**42,995**	0.00%	**42,995**	0.00%	**42,995**	0.00%	42,941	−0.13%	**42,995**
100.10_17	**43,574**	0.00%	43,497	−0.18%	43,517	−0.13%	**43,574**	0.00%	43,553	−0.05%	**43,574**
100.10_18	**42,970**	0.00%	**42,970**	0.00%	**42,970**	0.00%	**42,970**	0.00%	42,914	−0.13%	**42,970**
100.10_19	**42,212**	0.00%	**42,212**	0.00%	**42,212**	0.00%	**42,212**	0.00%	**42,212**	0.00%	**42,212**
100.10_20	**41,207**	0.00%	41,123	−0.20%	41,134	−0.18%	**41,207**	0.00%	41,050	−0.38%	**41,207**
100.10_21	**57,375**	0.00%	**57,375**	0.00%	**57,375**	0.00%	**57,375**	0.00%	**57,375**	0.00%	**57,375**
100.10_22	**58,978**	0.00%	58,922	−0.09%	**58,978**	0.00%	**58,978**	0.00%	58,975	−0.01%	**58,978**
100.10_23	**58,391**	0.00%	**58,391**	0.00%	**58,391**	0.00%	**58,391**	0.00%	58,370	−0.04%	**58,391**
100.10_24	**61,966**	0.00%	**61,966**	0.00%	**61,966**	0.00%	**61,966**	0.00%	61,903	−0.10%	**61,966**
100.10_25	**60,803**	0.00%	**60,803**	0.00%	**60,803**	0.00%	**60,803**	0.00%	**60,803**	0.00%	**60,803**
100.10_26	**61,437**	0.00%	61,368	−0.11%	61,368	−0.11%	**61,437**	0.00%	61,336	−0.16%	**61,437**
100.10_27	**56,377**	0.00%	**56,377**	0.00%	**56,377**	0.00%	**56,377**	0.00%	56,353	−0.04%	**56,377**
100.10_28	**59,391**	0.00%	59,332	−0.10%	**59,391**	0.00%	**59,391**	0.00%	**59,391**	0.00%	**59,391**
100.10_29	**60,205**	0.00%	**60,205**	0.00%	**60,205**	0.00%	**60,205**	0.00%	60,165	−0.07%	**60,205**
100.10_30	**60,633**	0.00%	60,629	−0.01%	60,629	−0.01%	**60,633**	0.00%	**60,633**	0.00%	**60,633**
Average:	**41,606.0**	0.00%	41,582.0	−0.06%	41,580.8	−0.06%	**41,606.0**	0.00%	41,547.8	−0.14%	**41,606.0**

needed the lowest running time out of all the techniques: this at least suggests that its complexity is average-to-low, which is again good news.

For 500.5_x, CPLEX took 2097 s on average (1 h max.), RS + B&B took 43 s (303 max.), GA and SA consumed their 60 min without finding any best value, and epiGA needed 1310 s on average (2699 max.) to find nine optimums (30% of instances). On these instances, epiGA had a better run time than a standard GA and an SA, only improved by CPLEX and RS + B&B. Note that we have not included in the execution times the instances in which an algorithm did not find the optimum.

The average execution times when optimizing 100.10_x was 386 seconds for CPLEX (665 max), 75 s for GA (1824 max), 557 s for SA (1769 max.), and 59 s for epiGA (1461 max.). we can see that again epiGA achieves the best execution time on average having found 100% of the best values. Authors of RS + B&B algorithm have not included these instances into their experimentation.

Finally, CPLEX took 3179 s to find 93% of the optimums for 250.10_x, RS + B&B found all of them in 3093 s, GA needed 2445 s to find 20%, and epiGA, 2520 s to find 60% of optimums. Note that RS + B&B needed by far the longest execution times, exceeding in four instances the maximum time set for the rest of algorithms (1 h).

Having tested epiGA in 120 instances and after achieving promising results, our next step was to analyze its convergence compared to the GA and SA, as described in the next section.

9.1. Convergence analysis

In this section we aim to clarify the internal behavior of epiGA and its relationship with GA and SA. Fig. 8 shows the convergence analysis of epiGA for the first instance of the four instance sets addressed. The graphs, which correspond to the best run (out of 30), show that SA converges very slowly compared to the other two algorithms. Additionally, epiGA hits the best value before GA and usually it is a better value (higher, as we are maximizing the profit).

Table 7
Accuracy of the algorithms on 30 instances of the 250.10 MKP (250 variables and 10 constraints).

Instance	CPLEX		RS + B&B		GA		SA		epiGA
	Best	%	Best	%	Best	%	Best	%	Best
250.10_1	**59,187**	0.00%	**59,187**	0.00%	**59,187**	0.00%	58,859	−0.55%	**59,187**
250.10_2	**58,781**	0.13%	**58,781**	0.13%	58,705	0.00%	58,390	−0.54%	58,705
250.10_3	**58,097**	0.01%	**58,097**	0.01%	58,094	0.00%	57,625	−0.81%	58,094
250.10_4	**61,000**	0.02%	**61,000**	0.02%	60,957	−0.05%	60,763	−0.37%	60,989
250.10_5	**58,092**	0.00%	**58,092**	0.00%	58,070	−0.04%	57,728	−0.63%	**58,092**
250.10_6	**58,824**	0.00%	**58,824**	0.00%	58,765	−0.10%	58,325	−0.85%	**58,824**
250.10_7	**58,704**	0.00%	**58,704**	0.00%	58,618	−0.15%	58,174	−0.90%	**58,704**
250.10_8	58,933	0.00%	**58,936**	0.01%	58,933	0.00%	58,547	−0.65%	58,933
250.10_9	**59,387**	0.01%	**59,387**	0.01%	59,381	−0.01%	59,056	−0.55%	59,384
250.10_10	**59,208**	0.00%	**59,208**	0.00%	**59,208**	0.00%	58,777	−0.73%	**59,208**
250.10_11	**110,913**	0.02%	**110,913**	0.02%	110,875	−0.02%	110,542	−0.32%	110,894
250.10_12	108,715	0.01%	**108,717**	0.01%	108,689	−0.01%	108,317	−0.35%	108,702
250.10_13	**108,932**	0.00%	**108,932**	0.00%	**108,932**	0.00%	108,581	−0.32%	**108,932**
250.10_14	**110,086**	0.00%	**110,086**	0.00%	110,037	−0.04%	109,687	−0.36%	110,081
250.10_15	**108,485**	0.00%	**108,485**	0.00%	108,458	−0.02%	108,209	−0.25%	**108,485**
250.10_16	**110,845**	0.00%	**110,845**	0.00%	110,821	−0.02%	110,452	−0.35%	**110,845**
250.10_17	**106,077**	0.00%	**106,077**	0.00%	106,075	0.00%	105,797	−0.26%	106,075
250.10_18	**106,686**	0.00%	**106,686**	0.00%	**106,686**	0.00%	106,380	−0.29%	**106,686**
250.10_19	**109,829**	0.00%	**109,829**	0.00%	109,825	0.00%	109,518	−0.28%	109,825
250.10_20	**106,723**	0.00%	**106,723**	0.00%	**106,723**	0.00%	106,502	−0.21%	**106,723**
250.10_21	**151,809**	0.01%	**151,809**	0.01%	151,801	0.00%	151,639	−0.11%	151,801
250.10_22	**148,772**	0.00%	**148,772**	0.00%	**148,772**	0.00%	148,545	−0.15%	**148,772**
250.10_23	**151,909**	0.00%	**151,909**	0.00%	151,900	−0.01%	151,765	−0.09%	**151,909**
250.10_24	**151,324**	0.03%	**151,324**	0.03%	151,269	0.00%	151,035	−0.16%	151,275
250.10_25	**151,966**	0.01%	**151,966**	0.01%	151,948	0.00%	151,694	−0.17%	151,948
250.10_26	**152,109**	0.00%	**152,109**	0.00%	**152,109**	0.00%	151,795	−0.21%	**152,109**
250.10_27	**153,131**	0.00%	**153,131**	0.00%	**153,131**	0.00%	152,884	−0.16%	**153,131**
250.10_28	**153,578**	0.00%	**153,578**	0.00%	**153,578**	0.00%	153,383	−0.13%	**153,578**
250.10_29	**149,160**	0.00%	**149,160**	0.00%	**149,160**	0.00%	148,879	−0.19%	**149,160**
250.10_30	**149,704**	0.00%	**149,704**	0.00%	**149,704**	0.00%	149,474	−0.15%	**149,704**
Average:	106,365.5	0.01%	**106,365.7**	0.01%	106,347.0	−0.01%	106,044.1	−0.30%	106,358.5

Table 8
Summary of the results. We report the percentage of hits of each algorithm and the average and maximum execution times (in seconds) for the instances in which the optimum was found. Note that the execution times of SACRO algorithms are not available and that proof time is also reported for RS + B&B.

Instances	CPLEX		RS + B&B		SACRO		GA		SA		epiGA	
	Hits	Time	Hits	Time	Hits	Hits	Hits	Time	Hits	Time	Hits	Time
100.5_x	**100%**	241	–	–	53%	50%	**100%**	5	60%	240	**100%**	**3**
		538		–				34		799		**22**
500.5_x	**100%**	2097	**100%**	**43**	0%	0%	0%	–	0%	–	**30%**	1310
		3600		**303**				–		–		2699
100.10_x	**100%**	386	–	–	60%	57%	**100%**	75	23%	557	**100%**	**59**
		665		–				1824		1769		1461
250.10_x	93%	3179	**100%**	3093	–	–	20%	**2445**	0%	–	**60%**	2520
		3600		37,031				**3600**		–		**3600**

10. Conclusions and future work

Through this article we have introduced epigenetics and its mechanisms with the aim of using it to build new computational algorithms. Our goal was to understand a biological domain that could represent an interesting source of inspiration on building new algorithms. These new algorithms will not only have different operators with respect to the standard ones, but a different representation that takes ideas from Nature itself. They are used to evolve more complex structures while expressing different relationships between genes beyond simple Mendelian ideas. The additional interest in learning from the environment and acting on inherited chromosomes is also a particular way of thinking that could be exploited by other researchers.

We have introduced four state-of-the-art competitors, IBM ILOG CPLEX, SACRO-PSO algorithms (SACRO-BT and SACRO-CBT), and Resolution Search + Branch & Bound. Moreover, we have implemented two well-known metaheuristics, namely Genetic Algorithm and Simulated Annealing, as complementary competitors.

We have moved from explanation/construction to actual evaluation. For this, we have parameterized all the algorithms implemented and tested them on 120 instances of the Multidimensional Knapsack Problem extracted from the OR-Library.

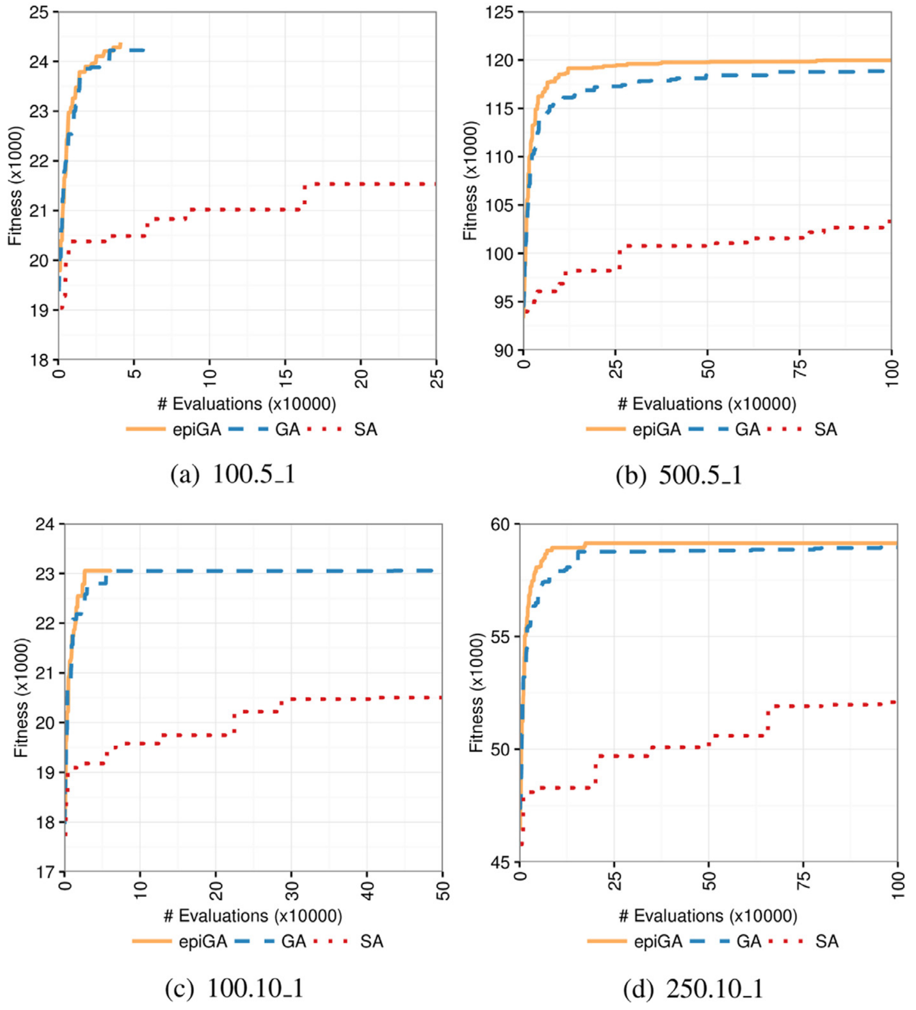

Fig. 8. Convergence analysis of epiGA, GA, and SA.

Although our goal in this paper is not to improve upon the state of the art, we did perform similarly and sometimes better in time than published results in the literature. In general, our actual goal is to provide the basis for a versatile customizable tool.

We strongly believe that the epiGenetic Algorithm presented in this article could be a powerful tool for solving combinatorial optimization problems as it can be tuned to different types of problems by using specially made representations, environments, and choosing the right set of epigenetic mechanisms to explore the solution space.

As a matter of future work we hope to test epiGA in other, different problems. Also, we will expand its capabilities by including several cells into each individual, which will allow us to implement a parallel version of it. Finally, as we have not tested all the existing epigenetic mechanisms, we plan to do so in the near future to further improve the searching process, adapting it to each problem's characteristics and environments.

Acknowledgments

This research is partially funded by the Spanish MINECO project TIN2014-57341-R (http://moveon.lcc.uma.es). Daniel H. Stolfi is supported by a FPU grant (FPU13/00954) from the Spanish Ministry of Education, Culture and Sports.

References

[1] V. Aguiar-Pulido, V. Suarez-Ulloa, J.M. Eirin-Lopez, J. Pereira, G. Narasimhan, Chapter 6 - computational methods in epigenetics, in: T.O. Tollefsbol (Ed.), Personalized Epigenetics, Academic Press, Boston, 2015, pp. 153–180.
[2] C.D. Allis, T. Jenuwein, D. Reinberg, M.-L. Caparros, Epigenetics, Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY, 2007.
[3] E. Angelelli, R. Mansini, M.G. Speranza, Kernel search: a general heuristic for the multi-dimensional knapsack problem, Comput. Oper. Res. 37 (11) (2010) 2017–2026.
[4] M.D. Anway, A.S. Cupp, M. Uzumcu, M.K. Skinner, Epigenetic transgenerational actions of endocrine disruptors and male fertility, Science 308 (5727) (2005) 1466–1469.

[5] O.T. Avery, C.M. MacLeod, M. McCarty, Studies on the chemical nature of the substance inducing transformation of pneumococcal types induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III, J. Exp. Med. 79 (2) (1944) 137–158.
[6] T. Back, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford university press, 1996.
[7] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.
[8] J.E. Beasley, OR-Library: distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (1990) 1069–1072.
[9] E.S. Belyaeva, O.V. Demakova, G.H. Umbetova, I.F. Zhimulev, Cytogenetic and molecular aspects of position-effect variegation in drosophila melanogaster, Chromosoma 102 (8) (1993) 583–590.
[10] J. Bender, DNA methylation and epigenetics, Annu. Rev. Plant Biol. 55 (2004) 41–68.
[11] J.M. Berg, J.L. Tymoczko, L. Stryer, Biochemistry, 6th, WH Freeman & Company Limited, 2006.
[12] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, Computing 35 (2003) 268–308.
[13] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, P. Michelon, A multi-level search strategy for the 0-1 multidimensional knapsack problem, Discrete Appl. Math. 158 (2) (2010) 97–109.
[14] V. Černý, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, J. Optim. Theory Appl. 45 (1) (1985) 41–51.
[15] V.L. Chandler, M. Stam, Chromatin conversations: mechanisms and implications of paramutation, Nat. Rev. Genet. 5 (7) (2004) 532–544.
[16] H.S. Chandra, V. Nanjundiah, The evolution of genomic imprinting, Development 108 (1990) 47–53. (Supplement)
[17] M. Chih, Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem, Appl. Soft Comput. J. 26 (2015) 378–389.
[18] M. Chih, C.J. Lin, M.S. Chern, T.Y. Ou, Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem, Appl. Math. Model 38 (4) (2014) 1338–1350.
[19] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86.
[20] V. Chvátal, Resolution search, Discrete Appl. Math. 73 (1) (1997) 81–99.
[21] J. Craig, N.C. Wong, Epigenetics: A Reference Manual, Horizon Scientific Press, 2011.
[22] F. Djannaty, S. Doostdar, A hybrid genetic algorithm for the multidimensional knapsack problem, Int. J. Contemp. Math.Sci. 3 (9) (2008) 443–456.
[23] G. Egger, G. Liang, A. Aparicio, P.A. Jones, Epigenetics in human disease and prospects for epigenetic therapy, Nature 429 (6990) (2004) 457–463.
[24] M. Egli, W. Saenger, Principles of Nucleic Acid Structure, 1st, Springer-Verlag, New York, 1984.
[25] A.P. Feinberg, B. Vogelstein, Hypomethylation distinguishes genes of some human cancers from their normal counterparts, Nature 301 (5895) (1983) 89–92.
[26] A.S. Fukunaga, A branch-and-bound algorithm for hard multiple knapsack problems, Ann. Oper. Res. 184 (1) (2011) 97–119.
[27] M.R. Gary, D.S. Johnson, Computers and Intractability: aGuide to the Theory of NP-Completeness, 1979.
[28] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[29] D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, Foundations of Genetic Algorithms, 1991.
[30] D.E. Goldberg, E. Smith, R.E. Smith, Nonstationary function optimization using genetic algorithms with dominance and diploidy, in: International Conference on Genetic Algorithms, 1987, pp. 59–68.
[31] A. S.M., H.M. Markowitz, On the solution of discrete programming problems, Econometrica 25 (1) (1957) 84–110.
[32] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1992.
[33] R. Holliday, J.E. Pugh, DNA modification mechanisms and gene activity during development, Science 187 (4173) (1975) 226–232.
[34] R. Holliday, J.E. Pugh, DNA modification mechanisms and gene activity during development, Cold Spring Harbor Monogr. Arch. 32 (1996) 639–645.
[35] E. Jablonka, M.J. Lamb, Epigenetic inheritance in evolution, J. Evol. Biol. 11 (2) (1998) 159–183.
[36] R. Jaenisch, A. Bird, Epigenetic regulation of gene expression: how the genome integrates intrinsic and environmental signals, Nat. Genet. 33 (2003) 245–254. Suppl (march)
[37] J.K. Kim, M. Samaranayake, S. Pradhan, Epigenetic mechanisms in mammals, Cell. Mol. Life Sci. 66 (4) (2009) 596–612.
[38] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[39] L. En, Chromatin modification and epigenetic reprogramming in mammalian development, Nat. Rev. Genet. 3 (9) (2002) 662–673.
[40] J.H. Lorie, L.J. Savage, Three problems in rationing capital, J.Bus. 28 (4) (1955) 229–239.
[41] J.-B.M. de Lamark, Recherches sur l'organisation des corps vivants: précédé du Discours d'ouverture du cours de zoologie donné dans la Muséum d'histoire naturelle, Fayard, 1802.
[42] R.J. Moraga, G.W. Depuy, G.E. Whitehouse, Meta-raPS approach for the 0-1 multidimensional knapsack problem, Comput. Ind. Eng. 48 (1) (2005) 83–96.
[43] K.G. Murty, Linear Programming, John Wiley & Sons, Inc., New York, 1983.
[44] R. Ohlsson, K. Hall, M. Ritzén, Genomic imprinting: Causes and consequences, Cambridge University Press, 1995.
[45] S. Periyasamy, A. Gray, P. Kille, The epigenetic algorithm, Evol. Comput. (2008) 3228–3236.
[46] R.H.A. Plasterk, RNA silencing: the genome's immune system, Science 296 (5571) (2002) 1263–1265.
[47] J. Puchinger, G.R. Raidl, U. Pferschy, The multidimensional knapsack problem: structure and algorithms, INFORMS J. Comput. 22 (2) (2010) 250–265.
[48] G.W. Redberry, Gene Silencing: New Research, Nova Publishers, 2006.
[49] W. Reik, W. Dean, J. Walter, Epigenetic reprogramming in mammalian development, Science 293 (5532) (2001) 1089–1093.
[50] A.D. Riggs, X inactivation, differentiation, and DNA methylation, Cold Spring Harbor Monogr. Arch. 32 (1996) 646–662.
[51] A.D. Riggs, T.N. Porter, Overview of epigenetic mechanisms, Cold Spring Harbor Monogr. Arch. 32 (1996) 29–45.
[52] K.D. Sarge, O.-K. Park-Sarge, Mitotic bookmarking of formerly active genes: keeping epigenetic memories from fading, Cell Cycle 8 (6) (2009) 818–823.
[53] L. Simó-Riudalbas, M. Esteller, Targeting the histone orthography of cancer: drugs for writers, erasers and readers, Br. J. Pharmacol. 172 (11) (2015) 2716–2732.
[54] M.K. Skinner, A new kind of inheritance, Sci. Am. 311 (2) (2014) 44–51.
[55] M.K. Skinner, M. Manikkam, C. Guerrero-Bosagna, Epigenetic transgenerational actions of environmental factors in disease etiology, Trends Endocrinolo. Metab. 21 (4) (2010) 214–222.
[56] R.E. Smith, An Investigation of Diploid Genetic Algorithms for Adaptive Search of Nonstationary Functions, 1988 Ph.D. thesis.
[57] J.A. Sousa, E. Costa, Designing an epigenetic approach in artificial life: the epiAL model, Agents Artif. Intell. (2011) 78–90.
[58] H. Szu, R. Hartley, Fast simulated annealing, Phys. Lett. A 122 (3–4) (1987) 157–162.
[59] I. Tanev, K. Yuta, Epigenetic programming: genetic programming incorporating epigenetic learning through modification of histones, Inf. Sci. 178 (23) (2008) 4469–4481.
[60] Y. Vimont, S. Boussier, M. Vasquez, Reduced costs propagation in an efficient implicit enumeration for the 0-1 multidimensional knapsack problem, J. Comb. Optim. 15 (2) (2008) 165–178.
[61] A.P. Wolffe, M.A. Matzke, Epigenetics: regulation through repression, Science 286 (5439) (1999) 481–486.
[62] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, Trans. Evol. Comp. 1 (1) (1997) 67–82.
[63] N.A. Youngson, E. Whitelaw, Transgenerational epigenetic effects, Annu. Rev. Genomics Hum. Genet. 9 (1) (2008) 233–257.
[64] S.K. Zaidi, D.W. Young, M.A. Montecino, J.B. Lian, A.J.V. Wijnen, J.L. Stein, G.S. Stein, Mitotic bookmarking of genes: a novel dimension to epigenetic control, Nat. Rev. Genet. 11 (8) (2010) 583–589.