Schema bandits for binary encoded combinatorial optimisation problems

Madalina M. Drugan[1], Pedro Isasi[2], and Bernard Manderick[1]

[1] Artificial Intelligence Lab, Vrije Universitieit Brussels, Pleinlaan 2, 1050- B, Belgium
Madalina.Drugan,Bernard.Manderick@vub.ac.be
[2] Computer Science Department, Carlos III of Madrid University, Spain
pedro.isasi@uc3m.es

Abstract. We introduce the schema bandits algorithm to solve binary combinatorial optimisation problems, like the trap functions and NK landscape, where potential solutions are represented as bit strings. Schema bandits are influenced by two different areas in machine learning, evolutionary computation and multi-armed bandits. The schemata from the schema theorem for genetic algorithms are structured as hierarchical multi-armed bandits in order to focus the optimisation in promising areas of the search space. The proposed algorithm is not a standard genetic algorithm because there are no genetic operators involved. The schemata bandits are non standard schemata nets because one node can contain one or more schemata and the value of a node is computed using information from the schemata contained in that node. We show the efficiency of the designed algorithms for two binary encoded combinatorial optimisation problems.

1 Introduction

A recent trend is to transfer expertise between machine learning (ML) areas, i.e. multi-armed bandits (MAB) and evolutionary computation algorithms (ECs) [15]. There are many similarities between multi-armed bandits and evolutionary computation mainly because they are both optimisation algorithms. The main difference between the two techniques is that MAB is used to optimise stochastic environments [5], whereas the majority of EC based methods are for optimisation of very large but deterministic environments. We want to use the similarities and differences between the two techniques to design new efficient hybrid optimisation algorithms.

Schema theorem. Genetic algorithms (GAs) [11] are powerful optimisation and search techniques that have been applied with great success to a wide range of applications [6, 1]. Let's consider the standard GAs with a binary string encoding of length ℓ for each solution. There are 2^ℓ possible strings. The GA processes a population of individuals by the successive application of fitness evaluation, selection of the better individuals followed by recombination of the genotypes of the selected individuals.

The common part in the representation of several individuals is called a schema [7, 11]. According to John Holland, the schema theorem [7] explains the success of genetic algorithms in general. It basically states that although the GA operates at the level of individuals, GA *implicitly* and *in parallel* processes information about schemata, subsets

of the search space. Moreover, it samples the most interesting schemata called building blocks in a near-optimal way using the analogy between schemata and arms in the bandit problem. That is schemata with the fitness mean above the average are grouped as a bandit arm and the schemata with the fitness below the average are considered to be a second arm. The schema theorem shows that selection increasingly focuses on the schemas with the fitness average above the mean. This brings us to the multi-armed bandit problem (MAB).

The exploration (the search for new useful solutions) versus exploitation (the use and propagation of such solutions) trade-off is an attribute of successful adaptation. The exploration implies the evaluation of new solutions that could have low fitness and the exploitation means the usage of already known good solutions. Holland modelled the exploration vs exploitation trade-off in ECs with multi-armed bandits. The higher fitness solutions are considered (or grouped) as an arm, and the lower fitness solutions are considered as a second arm. Mixing of good building blocks in GAs, i.e. the propagation of good schema in GAs, was studied in [14].

Multi-armed bandits (MAB) problems [4] have been studied since the 1930s and they arise in diverse domains, like the online profit-seeking automated market makers [13] and yahoo recommendation system [10]. In the stochastic MAB-problem, there are K arms and each time an arm i, where the set of arms in $1, \cdots, i, \cdots, K$ is selected, a reward r_i is drawn according to the probability distribution with fixed but *unknown* mean μ_i. The goal is to maximise the total expected reward $\widehat{r}_i$. If the true means of all arms where known, this task would be trivial. One selects the arm with the highest mean reward all the time. A MAB algorithm starts by uniformly exploring the K-arms, and, then, gradually focuses on the arm with the best observed performance. Since, the means are unknown one has to allocate a number of trials over the different arms so that, based on the obtained rewards, the optimal arm is identified as soon as possible and this with (very) high confidence.

To reach this goal, a tradeoff between exploration and exploitation has to be found. Exploration means that one tries a suboptimal arm to improve the estimate its mean reward while exploitation means that one tries the best observed arm which is not necessary the true best one. An arm selection policy determines what arm is selected at what time step based on the rewards obtained so far. The research question is what are (near)-optimal arm selection policies for the MAB-problem. An important heuristic that has emerged is that good policies, e.g. variants of the upper confidence bound (or UCB) policy, are *optimistic in the face of uncertainty* [12].

Thus, the trade-off between exploitation and exploration is important for both MAB and EC algorithms. MAB should pull all arms using an exploration strategy, to estimate their performance and it returns feasible, close to optimal, solutions using an exploitation strategy. Exploration in EC means to generate solutions in unexplored regions of the search space, and exploitation means to generate new solutions in promising regions of the search space using structural information about the current solutions. Selecting and using these strategies are not trivial and actually, the trade-off between them can increase the time needed to find an acceptable solution.

One MAB variant is the hierarchical bandit approach where the reward of one arm in the hierarchy is the reward of another one at one level deeper in the hierarchy [12].

Monte Carlo Tree Search (MCTS) is a recently proposed search method that builds a search tree in an incremental and asymmetric manner accordingly to a tree policy that selects the node with the highest priority to expand [3]. The tree policy needs to balance exploration versus exploitation, which for MCTS methods resembles the same trade-off in ECs. Exploration means to search in areas not sampled yet, whereas exploitation means to search in promising areas. Each round in MCTS consists of four steps: selection, expansion, simulation, and back-propagation.

Selection starts from the root, it selects successive expandable child nodes down to a leaf node. It selects child nodes that expand the tree towards most promising moves, which is the essence of MCTS. A node is expandable if it represents a non-terminal state and is unvisited. *Expansion:* unless a stopping criteria is met, MCTS creates one or more child nodes and choses from them a node, designated as the current node, using a tree policy. If no child was created, the simulation starts from a leaf node. *Simulation* plays at random from the current node using a default policy. *Backpropagation* uses the results from the previous steps to update the information in the nodes on the path from the current node to the root. MCTS is a statistical anytime algorithm for which more computing power means better results.

MCTS using *upper confidence bound* (UCB1) [2] as arm selection policy is called Upper Confidence Trees (or UCT). UCB1 is a very simple and efficient stochastic MAB with appealing theoretical properties, i.e. UCB1 is upper bounds the lost in choosing non-optimal arms. Each promising node of MCTS is evaluated accordingly to the UCB1 policy. UCT builds incrementally a search tree using random samples in the search space by expanding the nodes selected by the arm selection policy [3]. This approach is largely responsible for the success of Monte Carlo Tree Search (MCTS) where other methods fail, e.g. the game of GO [3].

The designed algorithms. We want to steer the optimisation in ECs using MCTS by making an analogy with schemas representation of solutions. In Section 2, the search space of ECs is structured as a schemata net with 3^ℓ possible schemata. We reveal some of the properties of the schemata net. Section 3 presents a baseline schemata bandits algorithm where each node in the net, thus each schema, is an arm. This bandit searches in a 3^ℓ dimensional search space for the optimal solution.

Section 4 proposes a condensed representation of the net for the schemata bandits that searches over a reduced search space of 2^ℓ. Each node is itself a bandit of schemata and we denote this algorithm as the bilevel schemata bandits.

Experimental results. Section 5 tests the performance of the proposed schemata bandits on two binary encoded problems: i) a deceptive trap function, and ii) a version of NK landscape that uses the deceptive trap functions and has best known solution. We show that the baseline schemata bandits performs better in terms of best found so far solution but the bilevel schemata bandits perform better in terms of minimising the expected regret and the computational efficiency. Alternative parameters are considered for the UCB1 algorithms in the schemata nets. Section 6 concludes the paper.

2 A schemata net structure

In this section, we present a schemata net structure and its properties.

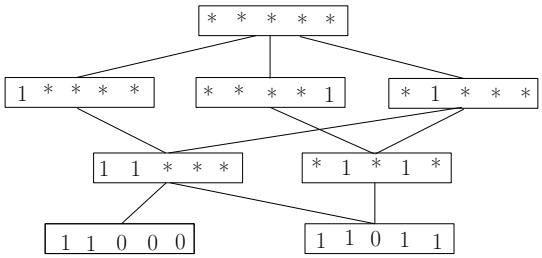

Fig. 1. An example of schemata net for a 5 dimensional strings.

We focus on *search spaces* that are ℓ-dimensional hypercubes, i.e. $\mathbb{B}^\ell$ where $\mathbb{B} = \{0, 1\}$ is the set of booleans and ℓ is the length of the bitstrings in the search space. A schema H represented as $H \in \{0, 1, *\}$ is a subspace of $\mathbb{B}^\ell$ that is also a hypercube. The don't care symbol $*$ can take on any value in $\mathbb{B}$. The *order* $o(H)$ of a schema H is the number of instantiated values, i.e. either 0 or 1. Here, we will also use the *dimension* $d(H)$ of a schema H: it is the number of don't care symbols and $d(H) = \ell - o(H)$. There are in total 3^ℓ schemata of which the most general schema $**\cdots**$ has dimension ℓ and of which all the 2^ℓ fully instantiated schemata, i.e. bit strings of the search space, have dimension 0. Note that the intersection of 2 schemata is again a schema.

Example 1. Let $\mathbb{B}^5$ be the 5-dimensional hypercube, see Figure 1. Then the schema $H_1 = 11 * **$ has order 2 and represents the 3-dimensional hypercube of all bit strings of length 5 starting with 11, i.e. H_1 contains 8 elements including 11001. And, $H_2 = 1 * * * *$ has dimension 4 and the schemata H_1 and H_2 share the element 11001.

Each node in the schemata net has the following attributes: i) a value, ii) children, and iii) parents. Let H be a schema of dimension $d = d(H)$, where d are the number of symbols $*$, and order $o(H) = \ell - d(H)$, where $o(H)$ is the number of positions where the value of bits is fixed to either 1 or 0.

Children: If we replace any don't care symbol $*$ by either 0 or 1 then we obtain one of the $2 \cdot d$ children of schema H. Each child has dimension $d - 1$. The leave nodes have no children.

In Example 1, schema $H_2 = 1 * * * *$ has $2 \cdot 4 = 8$ children, that are $10 * **$, $11 * **$, $1 * 0 **$, $1 * 1 **$, $1 * *0*$, $1 * *1*$, $1 * * * 0$ and $1 * * * 1$. The schema $H_1 = 11 * **$ that has dimension $d - 3$ has $2 \cdot 3 = 6$ children, that are $110 * *$, $111 * *$, $11 * 0*$, $11 * 1*$, $11 * *0$ and $11 * *1$.

Parents: A parent for the schema H has one fixed position replaced with the symbol $*$. If we replace any of the instantiated values 0 or 1 by a don't care $*$ then we obtain one of the o parents of H. Each parent has the order $o - 1$. The fully uninstantiated schemata have no parents.

For example, the schema H_2 has 1 parent, that is the fully uninstantiated schema, because it has the order $o = 1$, and the schema H_1 has two parents, i.e. $1 * * * *$ and $*1 * **$ with the order 1.

The value of a node: Let $b_1, \cdots, b_i, \cdots, b_n \in H$ be the bit strings, or individuals, evaluated for their common schema H. The values of function f (to be optimised) for

the set of bitstrings b_i are $f(b_i)$, where $i = 1, \cdots, n$. Then

$$\overline{f}(H) = \frac{1}{n} \sum_{i=1}^{n} f(b_i) \tag{1}$$

is the *estimated mean* value of f on the hypercube H, or simpler the value of H, and depends on the samples b_i used. The variance of $\overline{f}(H)$ on H will depend on its dimension $d(H)$: the higher the dimension the higher the variance and if the dimension is 0 then the variance is also 0.

There are two special types of nodes: i) the root, and ii) leaves.

The root is the fully uninstantiated schema, i.e. with the symbol $*$ everywhere. This node has no parents and it has $2 \cdot \ell$ children corresponding with the replacement of each position with a fixed value, 1 or 0.

A leaf is the node H that instantiate all its children, where each leaf has a fixed dimensionality $d \leftarrow |leaf|$. There are 2^d bitstrings for leaf schema corresponding with the d symbols $*$. The value of a leaf is fixed to the mean fitness values of the bitstrings generated for that schema. Let $b_1, \cdots, b_i, \cdots, b_{2^d} \in H$ be the bitstring evaluated for the leaf schema H. Then

$$\overline{f}(H) = \frac{1}{2^d} \sum_{i=1}^{2^d} f(b_i) \tag{2}$$

is the value of the leaf node on the hypercube H.

Because the size of a complete net is usually two large to be of a practical use, we expand the net given new solutions up to a given dimension $d > threshold$. The schemata with a higher dimension $d \leq threshold$ are structured in a tree where only the children but not the entire set of parents are further investigated and stored. For $threshold \ll \ell$, there are considerable more schemas in a schemata bandits, 3^ℓ, than total number of individual solutions, 2^ℓ. If $threshold$ is close to 1 the schemata net is small and the learning properties of the algorithm, e.g. generate bits strings from the best schema, are limited.

3 A baseline schemata bandits algorithm

The *baseline schema bandits algorithm* builds a tree where each node is a schema. The starting point for each iteration of this algorithm is the root that is the most general schema. The pseudo-code for this algorithm is presented in Algorithm 1.

Considering the steps specific for the Monte Carlo tree search algorithm, cf MCTS, the schemata bandits algorithm consists of three steps:

Selection: Starting from the root, select successively child nodes down to a leaf node. As in UCT, we select each time the child node that expands the tree towards the most promising parts of the search space. A node is expandable if it is unvisited. A popular policy to select the next node to expand is $UCB1$ [2] that upper bounds the loss resulting from choosing non-optimal arms. The UCB1 goal is to play often the optimal arm, in our case the optimal schema.

Let H be the selected node. The reward corresponding with each schema H, the arms or bandits in UCT, is the estimated mean $\overline{f}(H)$ over H based on all bitstrings

Algorithm 1: A baseline schemata bandits algorithm

Initialise the schemata bandits algorithm with n random individuals
for a fixed number of schemata net iterations **do**
  Select the root schemata, $H \leftarrow root$
  while a leaf node is *NOT* reached **do**
    Select the most promising child H_i of the current schema H using the UCB1 algorithm, cf. $argmax_i \overline{f}(H_i) + C \cdot \sqrt{\frac{2\log(t)}{t_i}}$
    Update the counters of the current and selected child schemata, cf $t \leftarrow t + 1$, $t_i \leftarrow t_i + 1$
    Update the current schema $H \leftarrow H_i$
  end while
  if the leaf node L was not expanded before **then**
    Expand all the individual solutions $2^{d(L)}$ in the leaf node L
    Update value of the parents of the schemata on the path between the root and the leaf node with the $2^{d(L)}$ bitstrings
  end if
end for
return the best found solution

$b \in H$ generated so far as in Equation 1. If a schema H has dimension d, then H has $2 \cdot d$ child schemata denoted as $H_i, i = 1, \cdots, 2 \cdot d$, and t_i is the number of times that H_i was evaluated so far. In order to play UCB1, we initialise the child schemata H_i as follows: for each child schema H_i the number of trials is set to one, $t_i \leftarrow 1$, and the estimated mean value is set to its minimum value, $\overline{f}(H_i) \leftarrow 0.01$. Thus, the number of trials t of the parent schema H is set to $t \leftarrow \sum_{i=1}^{2d} t_i$. $UCB1$ selects the child node H_i with the maximum index

$$\overline{f}(H_i) + C \cdot \sqrt{\frac{2\log(t)}{t_i}} \tag{3}$$

where the second term $C \cdot \sqrt{\frac{2\log(t)}{t_i}}$ represents the confidence term and encourage exploration of suboptimal schemata, and C is a constant scalar value, $C > 0$, usually set to 1. A larger value for $C > 1$ would encourage the exploration of new (possible) suboptimal schemata, and a smaller value $C < 1$ would promote almost exclusively the schema with the optimal expected mean even though this expected mean depends on the already generated bitstrings.

In the beginning, all children will be played equally often. As the number of samples t increase towards ∞, the confidence term increases and the suboptimal schemas that were not visited for a long time will attain the maximum value. The schema with the maximum expected mean is played the most. Note that the mean value of a more general schema will vary less than the mean value of a less general one.

Expansion: If in the selection step, a child node H_i that is not in the schemata graph is selected, i.e. $t_i \leftarrow 1$, a node in the net is created. If the selected child node H_i is already in the schemata graph, the counters are incremented, $t_i \leftarrow t_i + 1$ and $t \leftarrow t + 1$, and a child of this schema is selected. The expansion finishes with the

generation of a leaf node L. When a leaf node is reached, all the corresponding $2^{d(L)}$ bitstrings are generated and evaluated. Thus, the set of bitstrings $\{b_1, \ldots, b_{2^{d(L)}}\}$ is evaluated to $\{f(b_1), \ldots, f(b_{2^{d(L)}})\}$. These bitstrings update the expected mean of each matching child and parent of the schemata on the path between the root and the leaf L schemata.

Propagation: Using each of the generated solutions, b_i where $1 \leq i \leq 2^{d(L)}$, we update the mean values of all the schemata in the schemata graph that contain that solution. Thus, if b_i belongs to the schema H, $b_i \in H$, than its expected mean is updated $\overline{f}(H) \leftarrow \frac{n\overline{f}(H)+f(b_i)}{n+1}$ and the sample counter n is also updated, $n \leftarrow n+1$. This means that, in the propagation step, the schemata (and thus the inner nodes) that contain an individual solution are created if they do not already exists in the schemata bandits and if their dimension is smaller than $threshold$. The root schema is updated for all bitstrings. A higer dimensional schema is updated more often than a lower dimensional schema because there are more bitstrings that match to a higher dimensional schema than to a lower dimensional schema.

Related work. There are some important differences between the schema bandits algorithm and the standard UCT-algorithm [8, 3]. We actually define a graph where each node is the child of several parents. Because of the strong overlap between some of the schemata, the rewards of the corresponding nodes are strongly correlated while UCT assumes that the rewards independent. The creation of a schema can occur both during expansion and propagation. Therefore one way to improve the performance of the proposed algorithm is to prune the unpromising branches of the graph.

The schema bandits algorithm also relates to Estimation Distribution Algorithms [9] since no genetic operator is needed to generate new individuals. In addition, the schema bandits approach could offer theoretical guaranties on the convergence to the optimal solution.

Discussion. The version of schemata bandits introduced here is designed for bistrings. The extension of this algorithm to a representation with k-valued strings is straightforward, where $k \leq 2$. However, the dimensionality of such schemata net would grow exponentially to k^ℓ, where ℓ is the size of the string as before. Although the number of schemata in the net increases, the sparsity of the net decreases because a schema will have the same number of children regardless of the value of k.

Note that, in principle this algorithm is a *parameter free* optimisation algorithm. However, to increase the practical implications of this algorithm, we consider several parameters that decrease the computational complexity of the algorithm or tune the exploration / exploitation trade-off of the UCB1 algorithm. To decrease the search space of the schemata net, we have introduced the parameter $threshold$ that separates schemata net, which is very densely connected, to the schemata tree that is sparse. The parameter $d(L)$ decreases the computational time to run a schemata net by generating more that one bitstrings from a good schemata. This problem is also recognisable in the standard UCT algorithms with alternative solutions proposed in [12]. The constant value C gives the exploration / exploitation component of the UCB1 algorithm, but in the standard setting of UCB1 is set to 1.

A alternative version of this algorithm could select a random schemata from the net to start the net iteration. UCB1 could be again considered to select a schemata, but

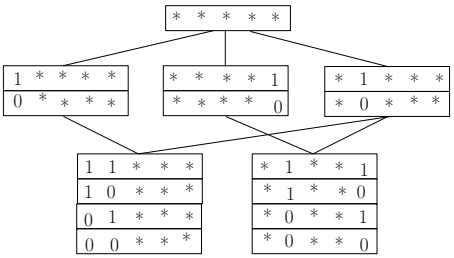

Fig. 2. An example of a bilevel schemata net for a 5 dimensional strings.

this time the number of arms is much larger means that the algorithm would require a longer time to learn the good schemata. New schemata added recently to the net are more often selected than the schemata that were created close the initialisation of the net even though the initial schemata have a good expected mean compared with the newly generated schemata. Note that in this case we have two UCB1 algorithms, one UCB1 algorithm over all schemata in the net and another UCB1 algorithm only for the children of a schemata.

4 A bilevel schemata bandits algorithm

The dimensionality of the schemata net, i.e. the number of total schemata in the net, is a problem to a search algorithm as it is much larger than the dimensionality of the problem itself, i.e. a schemata net has a 3^ℓ search space whereas the search space of the initial problem is only 2^ℓ. The reduction of the dimensionality of the schemata net could mean the increase in this algorithm efficiency. In this section, we propose a version of the schemata bandits that groups the schemata with the same indifferent $*$ positions such that the search space of this schemata bandits is 2^ℓ.

The group schemata net. Consider G a set of schemata with dimension $g \leftarrow d(G)$ corresponding the number of symbols $*$. There are $o(G) \leftarrow 2^{\ell-g}$ schemata contained in this node corresponding to the all schemata where the fixed positions are assigned one of the values 0 or 1. The root node contains only one schema, the most general schema with only symbols $*$. The root's children contains 2 schemata corresponding with one fixed position, and the children for the root children contains 4 schemata corresponding with the 2 fixed positions.

Example 2. Figure 2 shows with a simple example of such schemata bandits. A schemata group of order $o(G) \leftarrow 2$, i.e. there are two bits that can be fixed, has $2^2 = 4$ component schemata. Thus $00***$, $01***$, $10***$, and $11***$ are schemata in the same schemata group. Similarly, the schemata $*0**$, $*0**1$, $*1**0$ and $*1**1$ also belong to the same schemata group.

There are fundamental differences between the two schemata bandits approaches. With this approach, each bit string will match exactly one schema from a schemata group. Thus all the bitstrings match all schemata group. Each schema has an expected

mean value given in Equation 1. The value of a schemata group is given by the best expected mean value for that group of schemata. Each group schemata of dimension g has g children and g parents.

Note that the leaf nodes contains $2^{\ell-L}$ schemata, which for $L = 1$ means $2^{\ell-1}$ schemata where only one position has the symbol $*$. Therefore, for computational efficiency reasons, we set $L > 2$ on a high value. The variance in the expected mean values of schemata is larger for the leaf nodes than for the root node. In general, a group schemata with a higher dimension has a larger variance then a group schemata from a lower dimension.

The algorithm. In the following, we introduce the bilevel schemata bandits algorithms that has only two steps unlike the baseline schemata bandits introduced in Section 3 which has three steps. The expansion step is not necessarily in this implementation because each bitstring matches each node in the net. We then generate upfront all the nodes of the net up to leaves. There will be $\sum_{i=0}^{L} g^i$ group nodes, each node containing 2^i schemata. Thus, in total, there will be a fixed number of $\sum_{i=0}^{L} g^i \cdot 2^i$ schemata in the net.

Selection: Each iteration, the search in the net starts with the root node. Using an UCB1 algorithm, a child is selected from the current node and the selection process continues until a leaf node. When a leaf node is reached, there are sampled 2^{th} bitstrings from that leaf node proportional with the expected mean value of each schemata, where $th > 0$ is a constant.

Propagation: We update the matching schemata in each node with each generated bitstring, and the expected mean value for the matching schema is updated.

Performance of schemata bandits. The goal of the schemata bandits algorithms is to generate the optimal solution. The schemata bandits is a combination of evolutionary algorithms and multi-armed bandits thus measuring the performance of these algorithms is a complex task. To assess the quality of an schemata bandits algorithm, we evaluate the number of times each algorithm found the optimal solution and the mean of the generated solutions. Since storing and generating a schemata net is computational expensive, we need adequate performance metrics to compare the standard GAs and the schemata bandits algorithms. To measure its computational complexity, we take into account the number of schemas generated and the number of function evaluations.

To measure the performance of the $UCB1$-algorithm, we evaluate the expected regret of each schema that is the loss resulting from selecting suboptimal children of that schema. Each schema s has a regret that is calculated as

$$R_s = \sum_{i=1}^{g} \mathbb{E}[T_i(N)] \cdot (\overline{f}^*(N) - \overline{f}_i(N))$$

where $T_i(N)$ is the number of times the child node i was selected in N tree iterations and $\overline{f}_i(N)$ is the performance of a child node as before. We denote with $\overline{f}^*(N)$ the performance of the optimal child node, i.e. the node with the maximum expected mean value. We argue that the performance of the schema bandits algorithms is an issue that needs further investigation.

Nr block	ℓ	best sol	mean	nr schema	fun eval	regret
1	5	1.00 ± 0	0.34 ± 0	210 ± 0	32 ± 0	58 ± 0
2	10	1.00 ± 0	0.34 ± 0	35870 ± 70	1024 ± 0.45	3598 ± 0
3	15	1.00 ± 0	0.34 ± 0	171104 ± 0	31818 ± 0	169063 ± 0
4	20	0.95 ± 0.05	0.34 ± 0.01	127019 ± 5	276017 ± 46	1926978 ± 8507
5	25	0.92 ± 0.04	0.34 ± 0.01	179329 ± 32	318433 ± 54	2786122 ± 10451
6	30	0.88 ± 0.02	0.34 ± 0.01	230715 ± 6	319954 ± 4	3312440 ± 31333
7	35	0.86 ± 0.03	0.33 ± 0.01	281795 ± 0	319998 ± 0	3865973 ± 35264
8	40	0.85 ± 0.05	0.33 ± 0.01	332736 ± 20	320000 ± 0	4434108 ± 35609
9	45	0.76 ± 0.01	0.34 ± 0.01	383740 ± 14	320000 ± 0	5025488 ± 26731
10	50	0.73 ± 0.01	0.34 ± 0.01	434779 ± 9	320000 ± 0	5483765 ± 48880

Table 1. Performance of the schema bandits for 10 deceptive trap functions where the block size is 5.

5 Experimental results

In this section, we experimentally compare the performance of the two versions of schemata bandits on two binary encoded functions: i) deceptive trap functions and ii) an NK problem.

Deceptive trap functions. As test functions, we concatenate deceptive trap functions of 5 bits. The maximum value is for all bits 1s is 5, and the deceptive local maximum for all bits 0s is 4. If there is only a single bit 1, the value is 3, for two bits 1, the value is 2, for three bits 1 the value is 1, and for four bits 1 the value is 0. Let b be a bitstring, where $b = b_1, b_2, \ldots, b_\ell$ and ℓ is a multiple of 5. We have that $f(1) = 3$, $f(2) = 2$, $f(3) = 1$, $f(4) = 0$, the deceptive optimum is $f(0) = 4$ and the global optimum for 5 bits is $f(5) = 5$. Let b is the number of deceptive blocks and $b \in \{1, \ldots, 10\}$ and the size of a deceptive block $k = 5$. Then, the normalised value of the fitness function is

$$f(b) = \frac{1}{b * max_{1 \leq j \leq k} f(j)} \sum_{i=0}^{b-1} f(\sum_{j=1}^{k} b_{i \cdot b + j})$$

The trap functions are considered a difficult test problem for GAs because the large basin of attraction of the deceptive local optimum.

Table 1 gives the values of the above enumerated performance measures for the deceptive trap function. We run each experiment for 30 times and the schema bandits is iterated, i.e. selection, expansion and propagation, for 10^4 times. A leaf node evaluates $2^5 = 32$ solutions, i.e. $leaf \leftarrow 32$. Because of computational reasons related to memory usage, we set $threshold \leftarrow 3$. For example, for $\ell = 30$, there are $(2 \cdot 30)^3 = 216000$ schemata in the net to be stored and evaluated in the propagation step. A value 0.8 of mean fitness means that the algorithm reaches the deceptive optima, and value 0.9 of mean fitness means that 50% of the component tap functions found the global optimum. Note that the deceptive optimum is (almost always) reached in less than 320.000 function evaluations, and for $\ell \leq 40$ at least a quarter of the trap functions reach their optimal value.

For the bilevel schemata bandits, we set $th \leftarrow 5$ and $leaf \leftarrow 5$. We repeat each experiment 30 times and we iterate each schemata bandits for 10^3 times. In Table 2, the results in terms of finding the global optimum are not drastically deteriorated by the 10 times less iterations of the bilevel schemata bandits. Furthermore, the mean fitness

Nr block	ℓ	best sol	mean	nr nodes	fun eval	regret
3	15	0.98 ± 0.03	0.61 ± 0.05	2523 ± 10	32000 ± 0	162305 ± 9551
4	20	0.94 ± 0.02	0.59 ± 0.05	6957 ± 11	32000 ± 0	341496 ± 19586
5	25	0.92 ± 0.04	0.49 ± 0.01	16128 ± 10	32000 ± 0	635886 ± 40376
6	30	0.86 ± 0.02	0.44 ± 0.01	32824 ± 10	32000 ± 0	3312440 ± 31333
7	35	0.79 ± 0.02	0.44 ± 0.01	60465 ± 9	32000 ± 0	3865973 ± 35264
8	40	0.73 ± 0.05	0.42 ± 0.01	103039 ± 3	32000 ± 0	2752405 ± 401730

Table 2. Performance of the bilevel schema bandits for the deceptive trap function where the block size is 5.

Alg	ℓ	best sol	mean	nr nodes	fun eval	regret
Baseline	15	1.0 ± 0.0	0.35 ± 0.0	114426 ± 15	32000 ± 0	184406 ± 4140
schemata	20	0.84 ± 0.03	0.35 ± 0.00	560380 ± 12	32000 ± 0	245091 ± 9037
Bilevel	15	0.99 ± 0.03	0.63 ± 0.01	2434 ± 5	32000 ± 0	141655 ± 12068
schemata	20	0.83 ± 0.03	0.65 ± 0.01	8295 ± 10	32000 ± 0	374702 ± 41451

Table 3. Performance of the baseline (top half) and bilevel (bottom half) schemata bandits for the deceptive NK problem where the block size is 5 and for 10^3 schemata net iterations.

value, which is computed over the best mean fitness value for all the schemata in a node, is much higher than the mean fitness value of the baseline schemata bandits. Also the regret is smaller and a reason might be the smaller number of children arms for a single node. The number of fitness evaluation is also 10 smaller than in Table 1 and the number of nodes is about three times smaller.

The deceptive NK problem. We propose to use a NK problem that overlaps several deceptive trap functions. The fitness function is a sum of deceptive trap functions that overlap in $k-1$ positions, here $k=5$. Let b be a bitstring, where $b = b_1, b_2, \ldots, b_\ell$. Then, the fitness function of the deceptive NK problem is

$$f(b) = \frac{1}{(\ell-k)\cdot max_{1\leq j\leq k} f(j)} \sum_{i=1}^{\ell-k} f(\sum_{j=1}^{k} b_{i+j})$$

Note that the resulting NK problem is more complex that the initial deceptive trap functions since all the bits should be set at once in order to obtain the global maximum. This fact is reflected in the experimental results by seldom identifying the optimal solution, see Table 3. Comparing the two algorithms, the performance is similar, but the number of nodes generated is 2 orders smaller for the bilevel schemata bandits than for the standard algorithm.

In conclusion, the baseline and the bilevel schemata bandits have a similar performance on the two tested combinatorial problems. The advantage of baseline schemata bandits is the good performance in terms of the best found solution, but the bilevel schemata bandits have the advantage of grouping similar schemata and thus of a more compact net structure.

6 Conclusions

We combine techniques from evolutionary computation and Monte Carlo Tree Search paradigms in order to create new efficient optimisation algorithms. The schemata bandits algorithm combines the schemata theory with multi-armed bandits and its goal is to

generate the optimum solution. The baseline schemata bandits algorithm considers that each schema is a node in the schemata net which is connected both with schemata that are more general that the current schema, denoted as parents, and with schemata that are more specialised than the current schema, denoted as children. The main drawback of this schemata bandits is the dimensionality of the net, i.e. 3^ℓ where ℓ is the size of the bitstrings, for a smaller search space 2^ℓ. The bilevel schemata bandits is the second proposed algorithm with a reduced dimension net of 2^ℓ nodes, where each node is the group of schemata with the same positions for the symbol $*$. We test and compare the two proposed algorithms on two binary combinatorial optimisation problems. The experimental showed that the baseline schemata bandits is better in finding optimal solutions whereas the bilevel schemata bandits are performing better in terms of optimising the regret. We conclude that schemata bandits is a viable alternative for the genetic algorithms that deserve further investigation towards a very efficient optimisation algorithm.

References

1. M. Affenzeller, S. Winkler, S. Wagner, and A. Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. CRC Press, 2009.
2. P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J of Machine Learning Res*, 3:397–422, 2002.
3. C. Browne, E. Powley, D. Whitehouse, S. Lucas, P.I. Cowling, P. Rohlfshanger, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Trans on Comp Intel and AI in Games*, 4(1):1–46, 2012.
4. S. Bubeck and N. Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems.*, volume 5 of *Foundations and Trends in Machine Learning*. 2012.
5. W.J. Gutjahr and A. Pichler. Stochastic multi-objective optimization: a survey on non-scalarizing methods. *Ann Oper Res*, 2013.
6. R. L. Haupt and S. E. Haupt. *Practical genetic algorithms*. Wiley, 2004.
7. J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.
8. L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. In *Machine Learning: European Conference of Machine Learning*, 2006.
9. P. Larrañaga and J. A. Lozano. Editorial introduction special issue on estimation of distribution algorithms. *Evolutionary Computation*, 13(1), 2005.
10. Lihong Li, Wei Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of International Conference on World Wide Web*, 2010.
11. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
12. R. Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.
13. N.D. Penna and M.D. Reid. Bandit market makers. In *NIPS*, 2013.
14. D. Thierens. *Analysis and design of genetic algorithms.* PhD thesis, Katholieke Universiteit Leuven, Belgium, 1995.
15. Jun Zhang, Zhi-Hui Zhan, Ying Lin, and Ni Chen. Evolutionary computation meets machine learning: A survey. *Computational Intelligence Magazine, IEEE*, 6(4):68–75, 2011.