Schemata Bandits

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 knapsack problem, 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 schema theorem for genetic algorithms is a source of inspiration from the first area and hierarchical bandits from the second one.

1 Introduction

Genetic algorithms (GAs) are powerful optimisation and search techniques that have been applied with great success to a wide range of applications. The GA processes a population of individuals by the successive application of fitness evaluation of these individuals, selection of the better ones followed by recombination of the genotypes of the selected individuals. According to John Holland, the schema theorem [3] explains that success. It basically states that although the GA operates at the level of individuals it *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. The latter argument is based on the analogy between schemata and arms in the bandit problem. This brings us to the multi-armed bandit problem (MAB).

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 returned drawn according to probability distribution with fixed but *unknown* mean μ_i. The goal is to maximize the total expected reward. 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. Since, the means are unknown one has to allocate a number of trials over the different arm 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* [4].

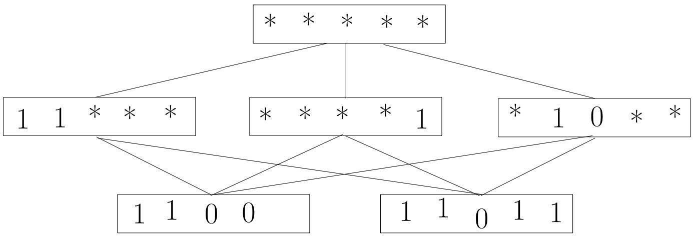

Fig. 1. An example of schemata net for a 5 dimensional strings.

Motivation. The MAB-problem is the focus of a lot of recent research and has been applied successfully in many domains. One 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 [4]. This approach is largely responsible for the success of Monte Carlo Tree Search (MTCS) where other methods fail, e.g. the game of GO [2]. MTCS using UCB as arm selection policy is called Upper Confidence Trees (or UCT) and it builds incrementally a search tree using random samples in the search space by expanding the nodes selected by the arm selection policy [2].

The main contribution. In this paper we propose the schema bandits algorithm that searches a l-dimensional hypercube for an optimal solution. Basically, it can be viewed as hierarchal bandit where each arm is a schema. Next, we introduce the schema bandits algorithm and present some preliminary results.

2 Schema bandits algorithm

Here, 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 the fully instantiated schemata, i.e. bitstrings 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 2. Then the schema $H_1 = 11 * **$ has order 2 and represents the 3-dimensional hypercube of all bit strings of length $*$ 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.

Let H be a schema of dimension $d = d(H)$. If we replace any don't care symbol $*$ by either 0 or 1 then we obtain one of the $2d$ children of H. Each child has dimension $d-1$. The fully instantiated schemata have no children. If we replace any of the instantiated values 0 or 1 by a don't care $*$ then we obtain one of the d parents (with dimension $d-1$) of H.

Let $f(b_1), \cdots, b_i, \cdots, f(b_n) \in H$ be the values of function f (to be optimised) in the bitstrings b_i that moreover belong to a common schema schema H, i.e. $b_i \in H, i = 1, \cdots, n$. Then $\overline{f}(H) = 1/n \sum_{i=1}^{n} f(b_i)$ is the *estimated mean* value of f on the hypercube 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.

The *baseline schema bandits algorithm* builds a tree where each node is a schema, child and parent nodes are described above and the most general schema is the root. It 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 $UCB1$, one of the upper confidence bound policies [1]. $UCB1$ is simple and has appealing theoretical properties, e.g. it gives an upper bound on the loss resulting from choosing non-optimal arms.

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 $b \in H$ generated so far. If a parent schema H has dimension d, then it has $2d$ child schemata denoted as $H_i, i = 1, \cdots, 2d$, and t_i is the number of times that H_i is evaluated so far. First we initialise all 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 minimal value, $\overline{f}_{H_i} \leftarrow 0.01$ and 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 $\sum_{i=1}^{n_i} \frac{\overline{f}(H_i)}{n_i} + \sqrt{\frac{t}{t_i}}$, where n_i is the number of times a complete solution that belongs to this schema is evaluated. Note that the mean value of a more general schema will vary less than the mean value of a less general one.

Expansion: A child node that is not in the schemata graph, i.e. $t_i \leftarrow 1$, is generated. If the child node is 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 is terminated with the generation of a leaf node.

Propagation: Using each of the solutions generated, we update the counters and the mean values of all the schemata in the schemata graph that contain that solution. 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. For $h \ll \ell$, there are considerable more schemas in a schemata bandits, $3^{\ell-h+1}$, than total number of individual solutions, 2^{ℓ}.

Discussion. There are some important differences between the schema bandits algorithm and the UCT-algorithm. 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 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.

Trap	nr block	ℓ	optim fit	mean	nr schema	fun eval	regret
5	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 trap functions.

3 Preliminary results.

The goal of this algorithm is to generate the optimal solution. To assess the quality of the algorithm, we evaluate the number of times each algorithm found the optimal solution and the mean of the generated solutions. 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 regret for each schema, that is the loss resulting from selecting suboptimal children of that schema. In Table 1, we show the root's regret.

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

The following table gives the values of the above enumerated performance measures. 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 solutions and a value 0.80 of optimum fitness means that the algorithm reaches the deceptive optima, and value 0.9 of optimum 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 optimum.

References

1. P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J of Machine Learning Res*, 3:397–422, 2002.
2. 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.
3. J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.
4. Rémi 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.