# Schemata Monte Carlo Network Optimization

P. Isasi[1], M. Drugan[2], B. Manderick[2]

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

**Abstract.** In this work we introduce a general combinatorial optimization method, based on Monte Carlo Tree search. The method constructs a tree whose branches are binary schema from which solutions for any optimization problems could be generated. The Monte Carlo Upper Confidence Policy (UCP) allows the generation of promising schemata allowing the abrupt reduction of the search space, keeping good compromise in the generation of good solutions. Moreover, the method includes a net structure that increases the accuracy in the estimations of the schemata, keeping the number of evaluation unchanged.
The method have been tested in hard binary optimization domains, including knapsack problems, multimodal functions and deceptive problems.

## 1 Introduction

Monte Carlo Tree search methods have been taken an increasing attention in the last 15 years. Originally his best success was in the context of the game of GO [Bru93], where its performance was bigger that any other automatic player method so far. From that, MCTS have been extremely successful in designing expert computer players for many others two players games, like Hex [Arn09,Arn10], Kriegspiel [Cia10], and Poker [Rub11]. Moreover, MCTS has been shown to outperform classic alphabeta search even in games where good heuristic evaluations are difficult to obtain.
In recent years, some other domains different from games have been addressed, mainly combining traditional ideas from minimax search with the construction of the trees to adapt the procedure to planning domains.
Binary combinatorial problems consists in the finding of a binary string that represents the optimal combination of yes/no alternatives. The difficulty of this problems resides in the weakness of the nature of the binary representation. The Genetic Algorithm perspective, for instance, tries to overcome this problem with the implicit parallel evaluation of the schemata performed by the method.
However this implicit evaluation is not followed of a direct use of the schemata for generating new solutions, and consequently new and better schemata.

In this work we follow the idea of evaluating how promising an schema could be, in terms of its ability to generate good solutions. However, generating good schemata have many intrinsic problems. First, the space of schemata is much bigger that those of the solutions. In a binary combinatorial problem of size L, the searching space has a size of $2^L$ while the size of the schemata space is $3^L$. So finding good schemata must be much harder than finding good solutions. Second, schemata evaluation is a difficult task because each schemata contains many examples that grows up exponentially with the size of the schemata, and they have, as well, a hierarchical structure that make them to share many examples, of even being one's subset of other.
To overcome the above feebleness we propose the use of the potential of Monte Carlo Tree Search (MCTS) to generate good candidates because:

- They could deal with incomplete information. They not need to have complete information about the efficiency of the candidates, they can estimate that efficiency from very few examples, and to improve those estimations as more information is available.
- They use a tree structure that fits very well the intrinsic nature of the schemata

Therefore, we propose a method for generating potential good schemata in terms of their capacity to represent good solutions. The method constructs a network whose nodes are binary schema from which solutions for any optimization problems could be generated. The Monte Carlo Upper Confidence Policy (UCP) will produce gradually more promising schemata, allowing the abrupt reduction of the search space, mantaining a good compromise in the generation of good solutions.
The net structure has been included for increasing the accuracy in the estimations of the schemata, keeping the number of evaluation unchanged.

## 2 Building a schemata network

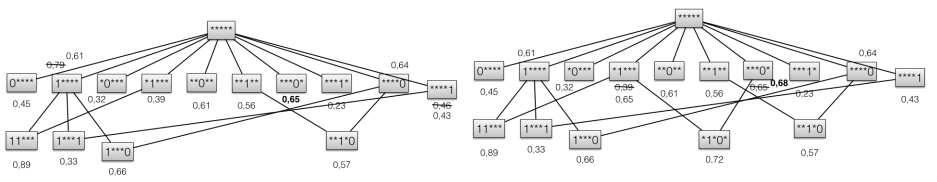
The construction of the tree is performed selecting and expanding a node per iteration, making the network growing in a unstructured and unbalanced way, taking into account the estimation of the nodes as good candidates to generate good solutions. In each iteration all the nodes present in the network could be considered for selection, except those who has been expanded in all their descendants.
The procedure is performed in four phases:

*Selection* The more promising unexpanded node of the network must be selected. A tree policy is designed to decide the meaning of being a promising node. There are many tree policy, in this work we propose the use of the Upper Confidence Policy (UCP), but any other could be used without further modifications in the method.

*Expansion* The selected node is expanded, generating one descendant, following random expansion policy. Once a descendant is generated, it is liked with all their parents, those that are less general and matching the new one, as showed in figure 1

*Simulation* The new generated node is from a sampling set of individuals represented by the schema of that node. The solutions in the sampling set are generated randomly, fitting the rules of the schema, and are not stored. The value assigned to the schema will be the average of the evaluation values of its sampling set.

*Backpropagation* All the evaluation values of all the nodes that are ancestors to the new created node are updated as showed in figure 1



a) b)

**Fig. 1.** Expansión and update of nodes in a network structure

## 3 Results

There are many well known combinatorial problems that could be used for testing new methods. We decide to present here a extensive used problem, the knapsack, and a very difficult theorethical function, the deceptive trap function.
The knapsack problem is one of the 21NP-complete problems established by Richard Karp [Kar72], and it has been intensively studied since the mid-twentieth century. It has the advantage of having a simple formulation for a very complex resolution. The formulation is:
Maximize:

$$\sum_{i=1}^{n} w_i x_i$$

Such a:

$$\sum_{i=1}^{n} w_i x_i \leq W$$

$$x \in \{0, 1\}$$

On the other hand the deceptive trap, is a theoretical problem used to frustrate escalade methods.
The function has a continuously growing in some direction until reaching a maximum value. However this value is a false optimal, because the function has a real punctual optimal value in the opposite direction. This is done by making blocks of five bits where de deceptive local maximum is 4, with all five bits equal to 0, decreasing till 0 with four bits equal to 1, and with the optimal value of 5, for all five bits equal to 1.

| Deceptive function | | | |
|---|---|---|---|
| Size | Best solution | Accuracy (mean, deviation) | Evaluations (mean, deviation) |
| 20 | 0,96 | 0,94±0,02 | 154831±152774 |
| 25 | 0,96 | 0,92±0,01 | 282726±185339 |
| 30 | 0,91 | 0,89 ±0,01 | 404122±269166 |
| 35 | 0,91 | 0,88±0,01 | 526249±237347 |
| 40 | 0,9 | 0,87±0,01 | 623444±141584 |
| 45 | 0,89 | 0,86±0,01 | 754003±146172 |
| 50 | 0,88 | 0,85±0,008 | 890116±104346 |
| Knapsack problem | | | |
| 24 | 1,0 | 1,0±0,0 | 368562±93717 |

**Table 1.** Results for the testing functions

The results, for 30 runs, are summarized in table 1. For the deceptive problem, more experiments were performed for sizes below 20. In all the cases the optimal value was reached, as well as for the knapsack problem, in a short period of time, so we can argue that 20 is a good threshold for considering the deceptive problem difficult enough..
For sizes above 20, iIt can be shown how the size of the problem doesn't disturb too much the results achieved. Even augmenting the size of the searching space up to $2^50$, a size of more that $10^{15}$, the optimal value is reached in almost the 50% of the blocks of the trap function, in a very small proportion of evaluations against search space ($8 \times 10^{10}$). We have to remark that in all the runs, the experiments were limited to not perform more than a million evaluations. More exhaustive experiments are needed to verify the potential of the method in larger runs. Based on preliminary test we can expect even better results for really huge sizes, where most popular methods performs poorly.

## References

[Bru93] Brügmann, B., Monte Carlo Go. Technical report, Syracuse University, NY, USA, (1993)

[Arn09] Arneson, B., Hayward, R., Henderson, P., MoHex wins Hex tournament. International Computer Games Association Journal 32 (2), 114–116 (2009)

[Arn10] Arneson, B., Hayward, R., Henderson, P., Monte Carlo tree search in hex. IEEE Transactions on Computational Intelligence and AI in Games. 2 (4), 251–258 (2010)

[Rub11] Rubin, J., Watson, I., Computer poker: A review. Artificial Intelligence 175 (56), 958–987 (2011)

[Cia10] Ciancarini, P., Favini, G. P., Monte Carlo tree search in Kriegspiel. Artificial Intelligence 174 (11), 670–684 (2010)

[Kar72] Richard M. Karp: Reducibility Among Combinatorial Problems. Complexity of Computer Computations. New York: Plenum. 85-–103 (1972).