# Meta-level Techniques for Planning, Search, and Scheduling

**Shahaf S. Shperberg**
Ben-Gurion University of the Negev, Be'er Sheva, Israel
shperbsh@post.bgu.ac.il

## Abstract

*Metareasoning* is a core idea in AI at that captures the essence of being both human and intelligent. This idea is that much can be gained by thinking (reasoning) about one's own thinking. In the context of search and planning, metareasoning concerns with making explicit decisions about computation steps, by comparing their 'cost' in computational resources, against the gain they can be expected to make towards advancing the search for solution (or plan) and thus making better decisions. To apply metareasoning, a meta-level problem needs to be defined and solved with respect to a specific framework or algorithm. In some cases, these meta-level problems can be very hard to solve. Yet, even a fast-to-compute approximation of meta-level problems can yield good results and improve the algorithms to which they are applied. This paper provides an overview of different settings in which we applied metareasoning to improve search, planning and scheduling.

## Metareasoning in MCTS

Monte-Carlo tree search (MCTS) is an algorithmic schema commonly used to search huge trees, especially when a good heuristic is not available (Browne et al. 2012). In general, MCTS grows a search tree, using four phases: node-selection, node-expansion, simulation, and backup. When the time for search is over, the action which leads to the most-promising child of the root (the one with the highest estimated utility, denoted as $\alpha$) is performed. The scheme used for node-selection essentially controls the search by deciding were to focus the computational effort. A popular approach for node-selection is the Upper Confidence Bounds for Trees (UCT) (Kocsis and Szepesvári 2006) which aims to minimize cumulative regret (balancing exploration and exploitation). However, it was shown that minimizing cumulative regret is actually inappropriate for move selection and that that simple regret and *value of information* (VOI) criteria are more appropriate, and result in more efficient search (Tolpin and Shimony 2012; Feldman and Domshlak 2014). In essence, the VOI of selecting nodes for simulation corresponds to the probability of changing $\alpha$ (either by increasing the estimated utility of some other child $\beta$ of the root, or by decreasing the estimated utility of $\alpha$) as a result of running simulations on these nodes. VOI can be defined in different ways, each way uses different assumptions which induce different computational overheads. The two previous node-selection schemes that are based on VOI, MGSS (Russell and Wefald 1991) and "blinkered" (Tolpin and Shimony 2012), only consider the VOI of individual nodes, a very myopic approach that often prematurely commitment to $\alpha$.

Our work in the area attempts to relax this myopic assumption. We defined (Shperberg, Shimony, and Felner 2017) a batch value of perfect information (BVPI) as a generalization of value of computation as proposed by Russell and Wefald. We showed that computing BVPI is NP-hard, but it can be approximated in polynomial time. In addition, we proposed a node-selection scheme that intelligently find sets of nodes with high BVPI. Finally, We applied our BVPI selection-scheme to existing MCTS algorithm applications and empirically showed that our methods outperform existing node-selection methods for MCTS in different scenarios.

Our current attempts are to adapt the VOI node-selection schemes to cases where a neural network (NN) provides value estimations instead of simulation (e.g. AlphaZero (Silver et al. 2017)). These schemes traditionally use node-selection methods which are based on variants of UCT, which still try to (wrongfully) minimize cumulative regret. Specifically, instead of having the NN return a single value-estimation for each node, we propose to return a value-distribution estimation. This could be achieved, for example, by replacing the standard reinforcement learning used in AlphaZero with a distributional reinforcement learning (e.g. (Dabney et al. 2018)). By having value-distribution estimations instead of single-value estimations the VOI-based schemes can be applied. These modified schemes can potentially improve the decision making abilities in many applications (such as AlphaGo and AlphaZero).

## Situated Temporal Planning

Agents that plan and act in the real world must deal with the fact that time passes as they are planning. For example, an agent that needs to get to the airport may have two options: take a bus, or take a taxi. Each of these options can be thought of as a *partial plan* to be elaborated into a complete plan that can be executed. Furthermore, consider a second example in which there are two partial plans, each estimated to require five minutes of computation to elaborate into a

complete plan. If only six minutes remain until both plans expire, then we would want the planner to allocate all of its remaining effort to one plan, rather than to fail on both.

Cashmore et al. (2018) recognized the problem of node expiration in the context of temporal planning with timed initial literals (TIL), where the TILs occur at times that are relative to when *planning* starts, rather than to when execution starts. However, their approach is relatively superficial and used merely to prune nodes that become infeasible based on their latest start time estimation. Such a planner fails on the the second example given above.

We defined the problem of selecting which nodes to focus during planning as a meta-level problem called *Allocating Effort when Actions Expire* (AE2, (Shperberg et al. 2019a)). AE2 abstracts away from the planning problem and merely assumes $n$ independent processes, each modeled by a distribution over wall clock times denoting the deadline and a distribution over the required time allocation to complete computation. The objective of AE2 is to schedule processing time over the $n$ processes such that the probability that at least one process finds a solution before its deadline is maximized. We have analyzed properties of the AE2 problem, developed a pseudo-polynomial time solution for the special case of known deadlines, and proposed an effective greedy algorithm for the general case. In addition, we have also tackled the extended problem (called ACE2) where processes (plans) have cost and the aim is to minimize the expected cost (Shperberg et al. 2020). Finally, we showed how our greedy scheme for the AE2 problem can be applied inside a practical situated temporal planner (OPTIC). An empirical evaluation suggests that the new planner provides state-of-the-art results on problems where external deadlines play a significant role (Shperberg et al. 2021).

Our current focus is to extend the metareasoning techniques to interleave planning and execution. Allowing to start execution before having a complete plan can increase the chance to find and execute a plan on time.

## Algorithm and Instance Selection

In the context of multiple agents sharing the same resource, metaresoning can be used for dividing the shared resource among all agents. In this context, we have considered (Shperberg, Shimony, and Yehezkel 2019) a set of black-box agents/algorithms (each with its own strengths and weaknesses) attempting to solve a pool of optimization problem instances. Unlike standard algorithm selection settings in which there is an individual time limit for each instance, here we have a global time limit for the entire set of instances. The goal is to maximize the sum of solution qualities, where each instance can be solved more then once, but only the best solution counts. Thus, a policy is a pair of problem instance and algorithm to execute at any given moment. The original motivation for this work was combining multiple programs that compete in the AI Angry Birds competition. In this competition, agents have 30 minute to play 8 unseen levels with total score being the sum of level scores. We have formulated the problem as a selection problem and showed that it is NP-hard even when the time and score performance profiles (distributions) of agents on levels are known and independent. Nonetheless, we have developed an an approximation algorithm for one simple case, as well as faster greedy algorithm for the general case which works well empirically on data collected from the angry birds game. Then, we have combined this greedy algorithm with a Bayesian learning scheme for obtaining the performance profiles. The combined algorithm was evaluated in past years competition settings, outperforming the individual agents.

## Bidirectional Heuristic Search

In bidirectional heuristic search (Bi-HS) we are given an implicit graph, a start vertex $s$, a goal vertex $g$, and a heuristic which estimates the minimal cost between vertices. The aim is to find a least-cost path between the $s$ and $g$. Bi-HS algorithms maintain two search frontiers (one from $s$ and the other from $g$) and need to connect them to find a solution.

A fundamental question in Bi-HS is *"how much of the search effort to invest in each frontier?"*. In an attempt to answer this question, Eckerle et al. (2017) investigated which nodes must be expanded by any Bi-HS algorithm in order to prove solutions optimality. While in unidirectional search there is a specific set of nodes for every problem instance that must be expanded (Dechter and Pearl 1985), in Bi-HS there is no such unique set. Instead, for every problem instance there are pairs of nodes that must be expanded (denoted as must-expand pairs or MEPs), each pair contains a forward-frontier node and a backward-frontier node. A pair is expanded if at least one of its nodes is expanded. Thus, these pairs induce many sets of nodes (of different size) from which one set needs to be fully expanded. Each set corresponds to a division of the search effort between the frontiers. Aiming to find a small set to expand, Chen et al. (2017) converted the problem of expanding all MEPs to the problem of finding a vertex cover in an abstract graph which they called $G_{\text{MX}}$. Thus, the minimal vertex cover (MVC) of the $G_{\text{MX}}$ is the minimal number of node expansions required to prove optimality of solutions. While the $G_{\text{MX}}$ can only be fully constructed in post analysis, it is possible to obtain edges from it during the search. NBS (Chen et al. 2017) is a prominent Bi-HS algorithm which always obtain such an edge and expands both of its nodes in order to get a $2\times$ MVC bound on the number of expansions. Our work (Shperberg et al. 2019c) presents other ways to exploit the $G_{\text{MX}}$ structure for node expansion. Instead of choosing a single edge from the $G_{\text{MX}}$ our algorithm, called DVCBS, maintains a dynamic sub-graph of the $G_{\text{MX}}$ (denoted as $DG_{\text{MX}}$) using frontier nodes. Then, DVCBS computes an MVC for this $DG_{\text{MX}}$ (in linear time) and uses it to choose which nodes to expand. DVCBS faces a tradeoff. On the one hand, constructing $DG_{\text{MX}}$ and computing its MVC often is more accurate and leads to better search results as new frontier nodes become available. On the other hand, these operations are computationally expansive. This tradeoff induces a metareasoning problem which we briefly discussed. Finally, we also developed a method for enabling existing algorithm to benefit from the $G_{\text{MX}}$ structure by incorporating this information into the heuristic function (Shperberg et al. 2019b).

## References


Browne, C.; Powley, E. J.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Liebana, D. P.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE AI Games* 4(1): 1–43.

Cashmore, M.; Coles, A.; Cserna, B.; Karpas, E.; Magazzeni, D.; and Ruml, W. 2018. Temporal Planning while the Clock Ticks. In *ICAPS*, 39–46. AAAI Press.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions. In *IJCAI*, 489–495. ijcai.org.

Dabney, W.; Rowland, M.; Bellemare, M. G.; and Munos, R. 2018. Distributional Reinforcement Learning With Quantile Regression. In *AAAI*, 2892–2901. AAAI Press.

Dechter, R.; and Pearl, J. 1985. Generalized Best-First Search Strategies and the Optimality of A*. *J. ACM* 32(3): 505–536.

Eckerle, J.; Chen, J.; Sturtevant, N. R.; Zilles, S.; and Holte, R. C. 2017. Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search. In *ICAPS*, 79–87.

Feldman, Z.; and Domshlak, C. 2014. Simple Regret Optimization in Online Planning for Markov Decision Processes. *J. Artif. Intell. Res.* 51: 165–205.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *ECML*, volume 4212 of *Lecture Notes in Computer Science*, 282–293. Springer.

Russell, S. J.; and Wefald, E. 1991. *Do the right thing: studies in limited rationality*. MIT press.

Shperberg, S. S.; Coles, A.; Cserna, B.; Karpas, E.; Ruml, W.; and Shimony, S. E. 2019a. Allocating Planning Effort When Actions Expire. In *AAAI*, 2371–2378. AAAI Press.

Shperberg, S. S.; Coles, A.; Karpas, E.; Ruml, W.; and Shimony, S. E. 2021. Situated Temporal Planning Using Deadline-aware Metareasoning. In *ICAPS*.

Shperberg, S. S.; Coles, A.; Karpas, E.; Shimony, S. E.; and Ruml, W. 2020. Trading Plan Cost for Timeliness in Situated Temporal Planning. In *IJCAI*, 4176–4182. ijcai.org.

Shperberg, S. S.; Felner, A.; Shimony, S. E.; Sturtevant, N. R.; and Hayoun, A. 2019b. Improving Bidirectional Heuristic Search by Bounds Propagation. In *SOCS*, 106–114. AAAI Press.

Shperberg, S. S.; Felner, A.; Sturtevant, N. R.; Shimony, S. E.; and Hayoun, A. 2019c. Enriching Non-Parametric Bidirectional Search Algorithms. In *AAAI*, 2379–2386. AAAI Press.

Shperberg, S. S.; Shimony, S. E.; and Felner, A. 2017. Monte-Carlo Tree Search using Batch Value of Perfect Information. In *UAI*. AUAI Press.

Shperberg, S. S.; Shimony, S. E.; and Yehezkel, A. 2019. Algorithm Selection in Optimization and Application to Angry Birds. In *ICAPS*, 437–445. AAAI Press.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T. P.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of Go without human knowledge. *Nat.* 550(7676): 354–359.

Tolpin, D.; and Shimony, S. E. 2012. MCTS Based on Simple Regret. In *AAAI*. AAAI Press.