Part 2 — Scientific abstract: Meta-reasoning for search

Search problems are ubiquitous in AI and CS, and have extremely diverse characteristics. Despite the tremendous volume of research on search in various settings, it is well known that for many search problems no algorithm dominates all others in all cases. Similarly, many algorithms have tunable parameters that need optimizing. The nature of the required solution (e.g., optimal, suboptimal etc.) also has significant impact on the desired search algorithm. Selecting the best algorithm and parameters for the task at hand is thus a non-trivial and important issue. Moreover, every problem instance induces a search graph, which can be extremely non-uniform and have areas with radically different behavior. This calls for algorithms which change their settings dynamically during the search, so as to better adapt to the task at hand.

This proposal attacks the search space and task diversity problems. We propose a two-level framework for search: the regular *search* level and a *meta reasoning* level (MR). The MR level selects the next computational operator to be used in the current state of the search. The search level applies the selected operator. This dual process is repeated until the search task is completed. This results in *more flexible search algorithms that should perform better than existing inflexible algorithms.*

Rational meta-reasoning, a theory presented over two decades ago by Russell&Wefald, aims at optimal decision-making w.r.t. which computational operator should be applied at every point of the search. It achieves a search algorithm that is (theoretically) optimal in its use of resources, with self-adaptation capabilities to the task and domain at hand. However, this *rational MR theory* is extremely non-trivial to apply in actual search, due to the difficulty in obtaining the requisite quantitative model (utility values and probability distribution), and its intractability (typically, higher than that of the search problem itself!). As a result, this theory has seen relatively little application to real search problems to date.

Nevertheless, recent work (some by PIs of this proposal), using different types of meta-reasoning for different scenarios, results in significant performance improvement. We thus propose to deeply study this two-level meta-reasoning framework and apply it to various settings, which should have great benefit to both search applications, and theoretical understanding of search algorithms. In order to achieve this we need to address the following challenges: **1:** Over which computational operators should we allow the meta-reasoning to operate? **2:** Since full rational meta-reasoning is hopelessly intractable, what simplifications or alternate schemes are appropriate to a given scenario? **3:** What type of utility and probability model do we use for meta-reasoning, and how do we obtain the requisite numbers (e.g. probability distributions)? **4:** What amount of computational resources to devote to meta-reasoning?

Investigating MR will be done through examining the above issues in different search scenarios using both theoretical analysis and empirical evaluation of the resulting algorithms. Most search scenarios we intend to examine involve *state-of-the-art algorithms, thus improving them is another significant outcome of the proposed research*: 1: A* variants: single frontier bidirectional search (should the next expansion be forward or backward?), lazy A* (whether to compute a heuristic *now*, or delay). **2:** Monte-carlo tree search: deciding where to sample and when to stop. **3:** Meta-agent conflict-based search for paths: whether to merge agents into a meta-agent. **4:** Backtracking algorithms for constraint satisfaction: deciding when to use a cheap heuristic vs. an accurate, expensive heuristic. **5:** Cost-aware search: (near-optimal) trade-off of computational resources vs. solution quality.

At later stages of the research, we aim to connect our understanding from the different scenarios to a set of guidelines for *effectively* applying meta-reasoning to search in general.

Part 3 — Detailed description of the research program

1 Introduction

Search for optimal or near optimal solutions of combinatorial problems is a fundamental field of research in classical and applied Artificial Intelligence. Search problems can be found in navigation [55, 16], routing, planning [4, 28], and numerous other combinatorial domains [33, 12, 34]. The common basic problem is to find a (usually cheapest) path or node in a (usually) huge implicitly-specified graph. In the past decade, this field has spread to many new directions and applications [56], including advanced search algorithms [13, 52, 60], better treatment of heuristic evaluation functions [58, 43, 22, 1], better theoretical understanding [6, 5, 21, 70] as well as devising new settings such as bounded cost search [54, 61], suboptimal search [59, 60, 67, 57] and more [53, 51, 9, 7]. Furthermore, the need for efficient search algorithms has emerged in new and real-world applications such as robotics, single- and multi-agent path planning, and numerous other domains. Search algorithms are also applicable in game trees, and in planning in dynamic and partially observable domains, the latter sometimes modeled as POMDPs [50].

Search domains are very diverse and have many different characteristics. It is well known that for many search problems no algorithm dominates all others for all cases [71, 48, 49]. Frequently, researchers try and come up with characterizations of the conditions for an algorithm to perform well or poorly [70]. Furthermore, many algorithms have tunable parameters that need optimizing [3]. Therefore, given a problem instance, currently, one needs to choose an algorithm from a collection of search algorithms, a problem known as the *algorithm selection problem* [42]. In addition, one also needs to choose the best performing values for the parameters of the algorithm. This is a complicated task as the best choice is not always known, and may vary in different problem instances within the same domain. Furthermore, even if one chooses the best suitable algorithm and parameter settings, the diversity of the search space poses a deeper problem. Every problem instance induces a search graph. But the search graph itself can be diverse and have areas with radically different behavior. As an illustrative example consider searching for a route on a typical map. As demonstrated in figure 1 some regions of the map include large highways while other regions include dense small highly connected streets with many cycles. Different parts of the search graph have different characteristics and a given fixed search algorithm might be well suited to one part of the map but less well suited to other parts.

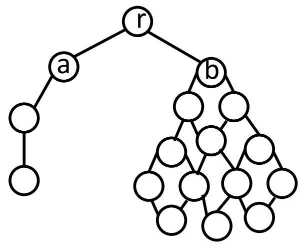


Figure 1: Example of a state space with diversity

Additionally, in some settings (such as cost-aware search in section 3.5) the task is to find the optimal solution only if time permits. A tradeoff may be required between search time and solution quality, and the tradeoff rate may even change dynamically during the search. The search algorithm should be able, for example, to settle for a suboptimal solution of it realizes that computational resources are running out - the task diversity problem.

This proposal addresses the search space and task diversity problems directly by using a framework that allows modifications to the search method on the fly. For example, consider the above navigation example. It might be more efficient to run best-first search when examining the highly connected region, while running depth-first based search on the region without cycles is more appropriate. Moreover, one might consider using one heuristic function for some states of the search, while switching to another heuristic function in other states [29].

A general formalization of the basic building blocks of search algorithms is that search algorithms advance towards a solution by applying a sequence of *computational operators*. A computational operator can be: visiting a node, generating its successors, computing a heuristic evaluation function, updating data-structures, etc. Most search algorithms have strict and inflexible rules for choosing the next computational operator. Our main intention in this proposal is to devise search algorithms which are flexible in choosing the next computational operator, in a disciplined way based on *meta-reasoning*.

1.1 Our proposal: meta-reasoning for search

We thus propose a two-level framework for search: a regular *search* level and a *meta reasoning* (MR) level. The MR level performs reasoning in order to better choose the computational operator for the current state of the search. The search level applies the selected operator. This dual process is repeated until the search terminates. We show that this framework improves search performance despite the computational resource overhead incurred by MR.

Since search scenarios are very diverse, the MR level may need to address different challenges and can take many forms. In fact, different types of *meta reasoning* for choosing between different types of computational operators were used in previous research [24, 10, 13, 11, 57, 60, 61]. However, in most of these cases, MR was hard-wired to work well specifically for the problem at hand and for the search algorithm that was currently used. There was usually little or no study of the MR level for the given problem and, needless to say, little or no understanding of MR in general. There is a need for systematically understanding the pros and cons of using MR within search in order to be able to apply it to new algorithms and to future scenarios. This is the focus of the current proposal.

1.2 Research objectives and expected significance

Our objective in this proposal is to establish a paradigm of flexible search algorithms. Such algorithms follow principles of meta-reasoning to choose computational operators according to the state of the search. We aim to develop this meta-reasoning approach for search, identify the challenges, and study ways to address them.

Since MR can take on many forms, our methodology will be to study MR in different search scenarios, most of them involving state-of-the art search algorithms. For each scenario we propose to develop a successful meta-reasoning level which will speed up the search. There will be a direct benefit as speedup is obtained for this specific search task. However, the main efforts in doing so will be to deeply study and understand the *meta reasoning* level for this particular case. Studying MR for different scenarios will allow us to deeply understand this concept and come up with a general theory or a set of guidelines for *effectively* using meta-reasoning in a general way - the main aim of our proposal.

The significance of this proposed research is that almost every application in AI and related fields which requires search should benefit from using this new framework. Users of search algorithms are often faced today with the challenge of choosing the correct search algorithm for a given search task. For example, suppose that one is willing to settle for a suboptimal solution within a given suboptimality rate. There is a range of search algorithms such as Weighted A* [41], Beam search [14, 74], Optimistic Search [57] and Explicit Estimation Search [60]. Each of these search algorithms apply computational search operators in a different manner. Using rational meta-reasoning as we propose will provide a theoretically sound way of applying the *best* computational operators so as to achieve the task requirements. For example, optimally trading off quality for search time based on an objective function defined by the current search task. Minimizing the sum of the search time and the time to traverse a path found by the algorithm, discussed in section 3.5, is an important special case of such a tradeoff.

Furthermore, our proposed research will enable application of combinatorial search methods to a larger variety of practical domains, and will allow solving existing problems (with diverse attributes) much faster.

1.3 Related work

A relatively recent line of research that attempts to address the diversity in search spaces (as well as a way to bypass the algorithm selection problem) is the notion of *portfolios* [69]. The main idea is to run a number of algorithms (or a number of variants, each with different parameter settings) in parallel. When the first algorithm finds a solution the entire process stops. The idea is that portfolios compensate for bad algorithm choice. It has been shown in many cases that when built correctly, portfolios are far better than any of the algorithms in the portfolio when activated alone [68, 47], despite the fact that we now have an overhead of a factor of k, when k algorithm are used in the portfolio. This has been especially evident in the recent domain independent International Planning Competition (IPC), where portfolio-based planners have shown to be superior to the other planning algorithms [23, 40]. However, many portfolios use the same fixed set of algorithms or parameter setting throughout the search and do not allow the flexibility of modifying anything during the search.

As mentioned above, a number of researchers already performed some kind of online meta-reasoning during search. We briefly cover some of them here. When a number of heuristic evaluation function are available the question of which of them to use arises. Thyer and Ruml [57, 60, 61] developed a number of algorithms that may switch on the fly from admissible heuristics to inadmissible heuristics based on special rules which are related the advancement of the search so far. *Selective Max* [10] is given a number of admissible heuristics and selects on the fly a single heuristic for each generated node - a form of meta-reasoning. It uses offline learning (with some online tuning) which is performed to produce a classifier for the selection, based on some features of the node. The above research is restricted to heuristic selection and provides few guidelines for using meta-reasoning in general.

Perhaps the most important work that addresses meta-reasoning for search in a general manner is the *rational meta-reasoning for search* theory, developed several decades ago by Russell and Wefald (denoted hereafter by $R\&W$) [45] (see section 2.2). The *rational meta-reasoning* framework aims at optimal decision-making w.r.t. which computational operator should be applied at every point of the search. This framework is extremely general, and has the potential to achieve everything we want: trading off solution quality for computational resources, optimizing search performance, and supporting diversity of the search problem. And yet, as discussed below, the gap between the theory of meta-reasoning and the practice of actually using it in search is huge, which is one of the main reasons why its use has not been ubiquitous. Since we have recently discovered several methods to help overcome this gap, our proposed research uses many of the ideas of rational meta-reasoning as building blocks, and further develop them in a number of ways.

Meta-reasoning is also closely related to anytime algorithms [8, 75], flexible and continual computation, and various ways to estimate value of information (VOI) [20, 35, 65]. Indeed we make use of the related techniques, too numerous to discuss here, in our proposed research.

2 Detailed proposed research

We begin by considering the basic building blocks of search algorithms. The straightforward understanding is that a search algorithm works by continuously visiting or traversing nodes in the search graph or tree until a goal node is found. Search algorithms differ in the strategy for choosing the next node to traverse (e.g., best-first search, depth-first search etc.). This formalization is rather limited because search algorithms also differ in the computations that are done in order to choose that next node. For example, some search algorithms use a set of heuristics to gain information before selecting which node to visit next [27, 43], while others try to select intelligently which heuristic functions to use [10]. Search algorithms also differ in the actions that are taken in the node and its neighborhood when ”traversing” the node, such as whether to insert into an OPEN list, which data-structures to use and when, etc.

Algorithm 1: The meta-reasoning framework

```
   Input: Search problem, computational operators, optional: utility and probability model
1  activate preprocessing phase        //optional
2  current-knowledge=initiate(start-state)
3  for $t$=0 to $\infty$ do
4      $o$=choose-computational-operator(current-knowledge) //meta-reasoning level
5      apply-operator($o$)                        //search level
6      update current-knowledge
7      if solution found or other termination condition then
8          halt                        // and return current solution(s)
9      end
10 end
```

A more general formalization is: search algorithms advance towards a solution by applying a sequence of computational operators. An operator can be visiting a node, generating its successors, computing a heuristic function, updating data-structures, halting the search, etc.

2.1 The meta-reasoning framework

We now describe the general framework for MR that we propose to study. In this framework a collection of computational operators is provided. Each such operator further advances the search by revealing more knowledge about the search process. The main challenge is to decide which computational operator to apply at each step. The MR framework (shown in Algorithm 1) works in two levels repeatedly. At each time step, we activate the MR level (Line 4) and choose the next computational operator. The search level activates the operator and updates the current knowledge about the searched graph/tree (Lines 5–6). This process is repeatedly executed until search terminates.

Traditional search algorithms are trivial forms of this framework where a single type or a small set of computational operators is used. A simple systematic rule decides which operator to apply and/or how to apply it within the search until a solution is found and the search task is fulfilled. For example, basic best-first search algorithms have only one type of computational operator - expand a node from the OPEN list. There is usually a basic rule for deciding which node to expand - the minimal cost node in OPEN. In our proposed framework an intelligent meta-reasoning phase is used to decide which operator should be activated at each step.

Our proposed MR framework (amongst other capabilities) can be used to change or choose different search algorithms during the search process as follows. Given a fixed search tree, different algorithms can be used under different nodes as long as they pass bounds and solutions in a coherent matter. Consider again the graph in figure 1. Assume two types of computational operators: (1) expand a node according to A*, i.e., remove the node from OPEN and then generate and add its children to OPEN (2) perform a cost-limited search from a node according to IDA*. An algorithm using meta reasoning could work as follows. For each node chosen for expansion the the MR level is invoked and a computational operator is chosen for this node. So, for node a MR might choose to apply IDA* while for node b A* should be the choice.

Meta-reasoning for search is a very general concept and can be used in many ways and take on different forms. However, there are some universal attributes which are valid for all cases, general open challenges which we aim to address in the proposed research:

(1) Given a search task, what are the computational operators that we allow the meta reasoning to choose from?

(2) How do we perform *efficient* meta-reasoning to choose an operator at any given step of the search?

(3) How to obtain the information required (e.g. statistics) for intelligent meta-reasoning?

(4) How often (i.e., in what key steps) do we perform full meta reasoning, or alternately, what is the amount of computational resources that will be allocated to MR?
(5) How would the different operators in different parts pass solutions, bounds, etc. to each other?

Below, we examine each of these questions in turn and suggest a number of alternative solutions which we intend to explore and use. However, as meta-reasoning is a central part of this proposal, we first briefly describe some of the basics of the $R\&W$ *rational meta-reasoning* for search [45].

2.2 Background: the rational meta-reasoning theory

The *rational meta-reasoning* framework by $R\&W$ aims at optimal decision-making w.r.t. which computational operator should be applied at every point in the search. In rational meta-reasoning, one can define a meta-level decision problem over states of the search space with computational operators as meta-level actions, as a problem of sequential decision-making under uncertainty. This meta-level problem can be formalized as a (belief state) meta-level Markov decision process (MDP) defined by a tuple $(S, A, \sigma, R,)$, as follows. Each state $s \in S$ of this MDP includes the entire knowledge about the search tree (or graph). For example, in a single-agent search problem, a state includes the known part of the search tree including all available heuristic estimates of nodes of this tree. The actions A are all the potential computational operators that are applicable to a state $s \in S$, such as expanding a node and generating its children, calculating a heuristic for a given node etc. The transition probabilities σ are defined by the distributions over new knowledge that may be revealed by applying each of the potential computational operators. For example, if we have a known distribution over heuristic values [32, 26, 72] that we expect a heuristic to yield, we can use it to describe our expectation of the potential (belief) state of the search after (and if) the heuristic is computed at a search node.

The rewards R are defined by the costs of the computational operators, in terms of time, memory, etc. and by rewards given when the search halts. For example, when the task is to find optimal paths, any state $s \in S$ which includes a provably optimal path receives a very large constant C as the reward. $R\&W$ suggested that, theoretically, optimally solving the meta-level decision problem reveals an optimal policy to adopt at every step of the search. This in turn would result in a search algorithm that provably incurs minimal search effort on average (as measured by the total cost of the different computational operators). Observe that the rational meta-reasoning framework also easily allows for more general search tasks, e.g. a trade-off between search time and solution quality can be achieved simply by modifying the reward function of the meta-level MDP appropriately.

However, the meta-level MDP is actually harder to solve in general than the original search problem! Therefore it makes no sense in practice to actually define and solve this MDP, at least not during search. Instead, $R\&W$ suggested a number simplifying assumptions under which the resulting simplified model can be solved quickly. The following assumptions were suggested for minimax search trees [46] but similar assumptions can be used as standards for other scenarios as well.
(1) Subtree independence: expanding a search tree node does not provide any information about values of other branches in the search tree.
(2) Myopic: all the expected utility computations about value functions in the meta-level problem assume that at most one node expansion will be performed before the search terminates. The computation step picked will be the one that has the best expected value. (In the original paper [46], "myopic" is actually presented as two assumptions: "single-step" and "meta-greedy", an issue we ignore here for simplicity.)
(3) All computational operators have a known time-cost.

Using all these assumptions, as well as a distribution model over what the computational operator will produce ($R\&W$ [46] assumed a normal distribution approximation), one can define (and possibly compute efficiently) a value of information (VOI) of a given computational operator. Under these assumptions it is optimal to perform the computational operator that has the highest expected VOI (taking the time-cost of the operator into account). We call this the *simplified version* of the $R\&W$ meta-reasoning method.

$R\&W$ experimented with the simplified version on a number of special case problems (such as game trees) and obtained positive results. Nevertheless, *none* of the above simplifying assumptions hold in general, and their practical applicability may thus be limited [65]. In fact, even in [46] the myopic assumption is used despite the fact that the algorithm used in the empirical evaluation in the paper actually violates the myopic assumption. Despite that violation, they achieve improved search performance in their experiments.

In general, at least two impediments exist in applying even the *simplified* meta-reasoning scheme to real search algorithms: (1) difficulty in obtaining probabilities and utility values, and (2) the high computational complexity of the meta-reasoning level itself. Given that usually in meta-reasoning one assumes that the amount of computational resources needed for meta-reasoning is *negligible*, the latter issue is very serious indeed. In fact, we believe that these two problems have been a major factor in the fact that rational meta-reasoning has not been massively applied in search to-date.

Nevertheless, the $R\&W$ seminal work on rational meta reasoning serves as a starting point. Applying their methods in general to other domains and generalizing these assumptions to more realistic settings assuring that that the problem remains tractable is possible, though extremely non-trivial. Similarly, devising general methods for obtaining meaningful statistical results for the distributions of the rewards is also a great challenge. Therefore, developing efficient *rational meta reasoning* schemes is a main focus of the proposed research as detailed below.

2.3 Challenge 1: Computational operators

In principle, one could always leave the meta-reasoning level the freedom to choose between all possible computational operators: all possible heuristic evaluation functions, all possible node expansions, etc. Although theoretically this could produce the optimal search algorithm, this complete freedom makes the meta-level decision problem hopelessly intractable in practice.

Instead, one should examine a given search task, and carefully define the different computational operators from which the meta-reasoning level will be allowed to choose. Thus, the main challenge here is to identify the relevant alternatives that will result in a tractable meta-level decision problem and also generate a better search algorithm. As discussed below, typically one begins with an existing algorithm (such as backtracking search, see the appropriate scenario in section 3.4), examines a tradeoff of interest for this algorithm (such as, which heuristic do we want to evaluate at each node), and then defines the computational operators accordingly (in this case, the operators to be chosen by meta-reasoning are heuristic function computations).

The choice of computational operators has a non-trivial impact on the rest of the challenges, as they affect the needed utility model, the distributions one needs to obtain, the complexity of the meta-reasoning (and thus the simplifying assumptions one needs to make), etc. These need to be carefully examined for each such scenario. Once we understand how to do this in general, we might allow for a generalization of this methodology, such as attempting to enlarge the set of computational operators and see if this still results in a practically solvable meta-reasoning problem. In the backtracking scenario, perhaps allow for selection of additional classes of heuristics, or even allow the meta-reasoning to consider a limited choice of nodes to expand. If successful, this automatically results in a different class of "hybrid" search algorithm that should also have better performance. The latter type of extension is, however, extremely non-trivial, and we will attempt it only in advanced stages of the proposed research.

2.4 Challenge 2: Meta reasoning schemes

As optimal rational meta-reasoning is intractable, approximate solutions are needed. This gives rise to a whole set of possible actual meta-reasoning schemes, ranging from a precise solution at the meta-level to ad-hoc heuristic schemes that are not even based on the rational meta-reasoning theory. We now detail a number of such schemes which we aim to use and study. Among them are variants of the rational meta-reasoning theory of $R\&W$.

2.4.1 Optimal solution

Here we attempt to optimally solve the sequential decision meta-reasoning problem (or an appropirate meta-level MDP) and thus perform optimal *rational meta-reasoning*. Usually this is intractable in the extreme, but for very simple cases such solutions may be possible. Optimal solutions may also be possible if the search horizon is extremely close, e.g. when there is time to execute only a very small number of search operators. This method is mainly useful for theoretical purposes, or as a yardstick ("how well does a particular search method do compared to the theoretical, unreachable, optimum"). Applying this method even for these extreme cases is not trivial as there are a number of open research questions such as how do gather the relevant statistics, how to solve the meta-level MDP, etc.

2.4.2 Optimal under the $R\&W$ assumptions

Here, we adopt the *simplified version* of $R\&W$. We solve the meta-reasoning problem (as defined by $R\&W$) under their assumptions: myopic, subtree-independence, and known constant time-cost. It is important to note that these assumptions do not hold in practice for many cases but still work well in settings where the search algorithm and domain do not violate the assumptions too badly. In these cases, these methods may be good approximations that can be computed quickly.

2.4.3 Relaxing the assumptions

As stated above, many search domains seriously violate these strong assumptions. We thus propose to carefully relax the strong assumptions. In general, the set of assumptions/relaxations should either hold in the search algorithm towards which they are applied, or be close enough that their violation does not nullify their effectiveness. We now list a number of such relaxations which we propose to study.

(1) Semi myopic framework: A serious limitation of the myopic assumption can occur when the VOI is highly concave, e.g. when the VOI of every computational operator is very low (or even zero), and yet the VOI of a batch of operators is high. Using the pure myopic approach, the meta-reasoning level would decide that no computational operator is worthwhile, and incorrectly stop the search [65]. In the *semi-myopic* framework, value of information is computed for *batches* of computational operators, rather than just a single operator, treating each batch as a single operator w.r.t. the VOI calculation. The semi-myopic relaxation allows for lookahead of various forms, for example one could allow all batches consisting of up to k operators, thereby providing a lookahead to depth k, which overcomes the premature stopping problem. (Note: the above *myopic assumption* is an extreme special case where all batches contains one operator, and the lookahead depth is 1.) The main challenge here is to pick batch sets for which VOI can be computed quickly enough, that fit the algorithm where meta-reasoning is used. An important special case of such a **semi myopic assumption** we mean to pursue is the recently introduced *blinkered* assumption [65]. This relaxation allows one to consider batches of operators "along the same branch of the search tree". The blinkered approach is promising (see Section 3.2).

(2) Relaxing the subtree-independence assumption: the subtree-independence assumption is also frequently violated. In certain cases, one can adjust for this using an appropriate dependency model [65]. However, this compli-

cates the meta-reasoning process, and may require more efficient ways to estimate the value of information, which also needs to be examined.
(3) Unknown costs: relax the assumption that operators have known costs and learn the costs online.

2.4.4 Rule based reasoning

In this direction we aim to provide off-line a simple rule or set of rules to decide which search operators to use. These rules are based on simple observations on the attributes of the search domain, rather than on *rational* meta reasoning. Most search algorithms actually use an extreme version of this scheme, such as the rule "expand the node that has the smallest f-cost" used in A*. Likewise, most search algorithms continually use the same search strategy, so a trivial rule is - apply the same operator type as at the parent. The advantage of such rules is that they are usually simple and easy to implement. Furthermore, in many cases where meta-reasoning was already used in search, such rules were applied. As an example, in our SFBDS algorithm (see Section 3.1.1) a simple rule was to choose to expand the side with the smallest branching factor. We propose to compare such rules with the general rational meta-reasoning framework on the different scenarios and study their tradeoffs.

2.4.5 Compiling rational meta-reasoning into rule based meta-reasoning

Another direction we propose to pursue is to perform an offline analysis of the meta-reasoning problem. In some simple cases the result may be a simple set of rules to be applied at runtime, as done for constraint satisfaction (see Section 3.4). This method "compiles" the information about the search algorithm (and possibly the domain) into a more efficient runtime approximate solution of the rational meta-reasoning problem.

2.5 Challenge 3: gathering statistics for rational meta reasoning.

In order to perform rational meta-reasoning, one must have appropriate distribution models and utilities. Each type of computational operator needs a separate distribution model, and these may be both algorithm dependent and domain dependent. These models also need to be very simple so as not to become intractable to evaluate. The utilities related to computation steps are algorithm dependent and have domain-dependent parameters. We have already implemented these steps for backtracking search in constraint satisfaction problems [63] (see section 3.4) and in Monte-Carlo tree search [64, 18] (see section 3.2). Similarly, if we have a known distribution over heuristic values [32, 26, 72] that we expect a heuristic to yield, we can use it to describe the expectation over possible outcomes of the relevant computational operator. These steps must be repeated or adapted for other cases.

The basic techniques employed in these scenarios can be used in a domain-independent way, but are specific to the type of algorithm into which meta-reasoning is introduced. Some of these techniques can be adapted to other algorithms, although this is extremely non-trivial.

2.6 Challenge 4: frequency of applying meta-reasoning

The MR level itself can be selectively activated to different degrees. Naturally, this spans a broad range of options where MR switches from simple rules to more complicated ones. The most important issue here is the time/memory overhead of the of the different MR decision procedures. This itself can vary and might be very low for some decision points (where the choice of operators is done using a simple rule), and more comprehensive at other points. There are a number of possible options for when and how to perform the MR level, with natural tradeoffs.
Always: Serves mainly as a benchmark for more sophisticated policies. Here full MR is done at each node of the search tree, or at every possible point where MR is relevant. Note that if MR is fast (such as the simple rules above,

or MR compiled offline), this policy might be the best.

Fixed accumulation rules: Here MR is performed once per constant number of search events, where events are measures of the search or the search tree. For example, do MR once per 1000 node expansions. Another example: perform MR only at even levels of the search tree, etc. When MR is not activated, a trivial rule is used, e.g., apply the last computational operator type again.

Sophisticated rules: E.g. keep track of *several* types of events, and invoke MR once a certain condition is satisfied.

Choosing the frequency of applying MR might depend on other settings. For example, the granularity of the computational operators might be in close relation to the frequency of applying meta-reasoning. Meta reasoning for computational operators of fine granularity (e.g computing a heuristic, expanding one child of the current node) might be applied at every node. By contrast, with computational operators of coarse granularity (e.g. running a complete depth-limited IDA* from a certain node), MR is used less frequently.

2.7 Challenge 5: technical aspects

Suppose that MR chose to change the search strategy (in cases where an operator with coarse granularity refers to a complete search algorithm). In some cases this change demands specific adjustments of data-structures or variables of the search. In our example of figure 1, IDA* is applied in the left part of the tree while A* is applied for the right side of the tree. How are results from the different searches combined together? A possible solution might be that one of these algorithms is activated at the base level (e.g., A* with OPEN) and the other algorithm (e.g., IDA* with the appropriate bound) is called for specific nodes. A ”communication protocol” between the two algorithms should be developed. In our case, some bound/value passing mechanism from IDA* to the base level A* should be developed. These technical aspects are radically different for each meta-reasoning scenario, and therefore need to be examined separately for each case.

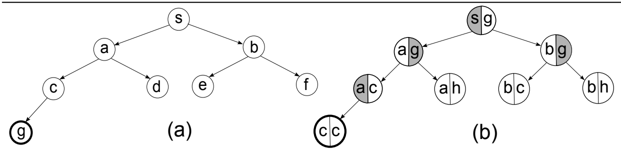
3 Search scenarios: methodology and preliminary results

Our overall methodology is to initially examine scenarios where MR is more likely to be applicable and gain performance. Most of the scenarios involve improving a specific algorithm across *multiple* search domains. For some of the scenarios, described below, theoretical guarantees should be possible or have already been achieved, as well as empirically shown improvements of the respective algorithms. Ideally, we might wish to address all the challenges listed above in all the scenarios. In practice, this is not feasible or even applicable as the scenarios are very different. We thus plan to apply a suitable subset of the techniques to each of these scenarios. We list the most important techniques within each scenario below but others might be also applied. Nevertheless, we intend to perform a systematic study over these scenarios as described above.

The benefit of using these scenarios is twofold. First, speedup in the search can be obtained for these algorithms and new state-of-the-art results achieved for the particular scenario. This is of great importance as these algorithms are vastly used by others. Second and more importantly, through the use of these test cases, we aim to better understand the general framework of how MR can be applied more generally in practice. Based on all our experience, we aim to provide a theory or at least a set of guidelines for using MR on top of search algorithms in general.

3.1 Scenario 1: Variants of A*

A* has a large number of variants, all committed to expanding nodes in a best-first search order according to the cost function $f(n) = g(n) + h(n)$. A number of recent A* variants have internal decision points that can greatly benefit from systematic meta-reasoning.



MR Policy	Nodes	Time
Forward	2,478,269,076	3,086s
JIL(0)	260,506,693	362s
JIL(1)	17,336,052	120s

Figure 2: Left: illustration of SFBDS. Right: results on the 17 pancake puzzle

3.1.1 Example 1: Single frontier bidirectional search

The *single frontier bidirectional search* algorithm (SFBDS) [13] requires internal decision making on "direction change", as follows. In traditional unidirectional search every node in the search tree corresponds to a single state in the state-space. A node in the SFBDS search tree (called a *double node* [38]) consists of a pair of states, s and g, and corresponds to the *task* of finding the shortest path between them. This task is recursively decomposed by expanding either s or g and thus generating new tasks from either (1) the neighbors of s to g, or (2) the neighbors of g to s. At every double node a *side choice* policy decides which of the two states to expand at this node, i.e., whether to proceed forward or backward at this node. Given a policy, a tree is induced which can be searched using any admissible search algorithm. Therefore, a node with minimal f-value is chosen and needs to be further developed.

Unidirectional search and SFBDS are illustrated in Figure 2(Left). The objective is to find a shortest path from the start state, s, to the goal state, g. In unidirectional search (Figure 2a) every node implicitly solves the task of getting from the current node to g, and the search proceeds across the tree until g is found. Now, consider searching the same tree with SFBDS (Figure 2b). Double nodes are labeled with the shortest-path task that should be solved beneath them. Within each node, the state expanded is shaded in the figure where left (resp. right) means expanding the forward (resp. backward) state. The policy used in this example is the *alternate* policy, so that at even depths the forward state is expanded and at odd depth the backward state is expanded. For example, at the root, the task is (s, g) resulting in two children, (a, g) and (b, g). At node (a, g), however, the *alternate* policy chooses g for expansion. This generates nodes for all neighbors of g, leading to (a, c) in our example. Finally, at (a, c), state c is chosen for expansion, generating a goal node (c, c). There are therefore, two computational operators types in this scenario – the two sides that may be expanded next.

SFBDS has freedom in choosing which side to expand. This is where meta-reasoning comes in. In [13, 38] only simple policies for choosing which side to expand were used. The table in Figure 2(right) compares several simple MR schemes for choosing which side to expand at each node on 10 random instances of the 17-pancake puzzle with a 7-token PDB [13]. The first line uses a fixed policy of always expanding the forward side. The next two lines are simple *rule-based* policies which perform lookahead to level 0 or level 1 (we omit the details here). Even these simple polices achieve significant speedup up to two orders of magnitude in the overall time over a fixed policy.

We propose to use our more sophisticated policies and study their effect on SFBDS. In particular, better hard-wired rule-based methods can be provided. More importantly, rational meta-reasoning variants are also relevant here. Statistics should be gathered w.r.t. the size of the subtree below each side and the likelihood of including a goal. These will be used for the rational MR variants.

3.1.2 Example 2: A* - IDA* hybrid

A* [17] works well in search spaces with cycles and transpositions, while IDA* [31] works well in tree-like search spaces.[1] Consequently, given a search problem one would choose either A* or IDA*, according to prior knowledge of the structure of the problem's search space. However, since different parts of the search space may exhibit different structures (as in figure 1), choosing to search only with A* or only with IDA* may be inefficient. We thus propose

[1]There are other considerations, such as the range of f-values and the time per node.

to combine A* and IDA* using MR as follows. The baseline algorithm is A*, having OPEN and CLOSED lists. For every node $n \in OPEN$, MR considers the regular computational operators of expanding n, generating its children and evaluating their heuristics. However, an additional operator is considered: running an IDA* iteration, rooted at n and with threshold $f(n)$. As a result of this operator, either: 1) a goal is found with cost $\leq f(n)$, 2) such a goal is not found, in which case we can obtain a stronger admissible cost function for n.

In preliminary work, we examined the *A* with Lookahead* algorithm (AL*) [52]. AL* is a variant of A* that performs an IDA* iteration from *every* node n generated with a threshold $T = f(n) + k$ for some constant input parameter k. The results show that AL* with some values of k is faster than both A* and IDA* on a range of domains. The open challenge is how to set k.

Rational MR can address this challenge neatly and more generally. Instead of deciding on a static value of k for the entire search, different decisions can be made for every state. In MR terms, an IDA* iteration from a node is one possible computational operator to deliberate upon. Estimating the effectiveness and computational cost of such an IDA* iteration for a specific node n can be learned in preprocessing (e.g., by abstractions) or from past experience gained from previously generated nodes. We expect the resulting algorithm to enjoy the complementary benefits of A* and IDA* in a more general way than AL*.

3.1.3 Example 3: Selecting heuristics in lazy A*

Lazy A* (LA*) is a variant of A* used in [73]. Suppose we have two heuristics, h_1 and h_2, that a node n is generated, and that f_{best} is the lowest f-value in OPEN, LA* consults only h_1 first. If after applying this heuristic $f(n) > f_{best}$, n is inserted into OPEN with its current $f(n)$. Later, when $f(n) = f_{best}$, we consult h_2 and if $f(n)$ increases further, n is re-inserted into OPEN. Otherwise n is expanded. Thus, LA* lazily applies the heuristics one at a time until either $f(n) > f_{best}$ or, if all heuristics were applied and $f(n) \leq f_{best}$ then n is expanded and its children are generated. The advantage of LA* is the saving in computing heuristic functions for nodes where the first heuristic is large enough so that they will never be expanded. The down side is that for some nodes both heuristics are evaluated and such nodes are inserted into OPEN more than once. We have already experimented with regular A* vs. LA*. The results show that although lazy A* frequently outperforms A* this was not always the case, due to overheads.

A better method would be to apply rational MR. Given two heuristics, there are three computational operator types: consult h_1, consult h_2, and expand the node. MR should choose when to compute which heuristic. As a first approximation, we intend to define an appropriate (approximate) VOI in terms of how much search time evaluating each heuristic is expected to save. This is contrasted with the (approximately) known time to compute the heuristic.

3.2 Scenario 3: Monte carlo tree search

Monte-Carlo tree search (MCTS) is a very useful technique for many intractable search problems. Perhaps the most prominent special case is the success of UCT search [30] in the game of Go [15], resulting in a breakthrough improvement in the quality of computer Go. UCT is a scheme for deciding which branch of the search tree to sample next (a complete single sample is also called a rollout), using an equation based on solid theoretical guarantees for the multi-armed bandit problem [2, 30].

Alternatively, rational meta-reasoning is most appropriate for this type of search, because it lays the foundations for optimal sampling. Additionally, the hardest hurdle for meta-reasoning, i.e. the problem of obtaining distributions, has a natural solution here, as the needed distributions are simply obtained from the sampling process itself. In MCTS, the computational operators among which we need to decide by meta-reasoning are: which branches of the search tree to sample, and when to stop sampling. For this scenario, we propose to develop MR to optimize the sampling process, both in generating samples and in providing a stopping criterion.

Preliminary results: We examined the assumptions behind the state-of-the-art UCT algorithm [30]. Although based on solid theoretical guarantees for the multi-armed bandit problem [2], UCT is based on a good solution to a problem that is *not* the same as the meta-reasoning needed in Monte-Carlo tree search. For the first step in each rollout (for both game-tree search for planning under uncertainty), a more appropriate problem is the *selection problem* [65]. That is, while the UCB formula at the core of UCT (approximately) optimizes "cumulative regret" (i.e. maximizing total of all rewards), MR principles imply that at the first step of the rollout it is more appropriate to optimize "simple regret" (i.e. maximize only the final reward minus the cost of experimentation) [64]. A simple adaption of the UCB formula based on these observations already achieves superior results, both in theory and in practice [64]. An attempt to use a crude estimation of value of information (VOI) of a sample have been shown to work even better in several domains, such as the sailing domain and computer Go [64, 18].

A further step is taken in [18] which revisits a formal definition of the belief-space MDP for the appropriate selection problem from [19], and theoretically analyzes its properties. Two schemes are used, a Bayesian scheme that assumes known distributions, and a frequentist scheme that generates bounds on VOI of samples using only statistics of previous samples. Since under the pure myopic assumption the value of information of a sample is usually zero, one could not use $R\&W$'s simplified model directly. Instead, the bound is based on the "blinkered" semi-myopic assumption: many samples are allowed, but all future samples will be done in the same branch of the search tree.

One of the theoretical results we obtained is a bound on the VOI for N samples [18]. Denote the number of samples already taken from node i by n_i, and the sample average of the outcome of n_i samples at node i by $\overline{X}_i^{n_i}$. α is node number with the best sample average value, and β is that of the second best node.

Theorem 1 *The "blinkered" value of information Λ_i^b for N samples at node i is bounded from above as*

$$\begin{aligned}\Lambda_\alpha^b &\le \frac{N\overline{X}_\beta^{n_\beta}}{n_\alpha}\Pr(\overline{X}_\alpha^{n_\alpha+N} \le \overline{X}_\beta^{n_\beta})\\ \Lambda_{i|i\ne\alpha}^b &\le \frac{N(1-\overline{X}_\alpha^{n_\alpha})}{n_i}\Pr(\overline{X}_i^{n_i+N} \ge \overline{X}_\alpha^{n_\alpha})\end{aligned} \tag{1}$$

An upper bound (*not* in probability!) on the VOI Λ_i^b is then obtained (φ is an appropriate constant):

$$\begin{aligned}\Lambda_\alpha^b &\le \frac{2N\overline{X}_\beta^{n_\beta}}{n_\alpha}\exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha}-\overline{X}_\beta^{n_\beta})^2 n_\alpha\right)\\ \Lambda_{i|i\ne\alpha}^b &\le \frac{2N(1-\overline{X}_\alpha^{n_\alpha})}{n_i}\exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha}-\overline{X}_i^{n_i})^2 n_i\right)\end{aligned} \tag{2}$$

Empirical evaluation in several domains (sailing and Go) show significant performance improvements [18], compared to both UCT and to our earlier results from [64].

Proposed extensions: In this scenario, we intend to advance in three directions, which should yield further performance improvements: **(1)** Applying these methods of analysis and VOI estimation to additional steps of the rollout beyond the first step. This issue is extremely non-trivial as the analysis differs depending on whether this is an adversarial game, or a "game against nature". **(2)** Improve the stopping criterion to consider value of information in future moves. This is complicated by the fact that value of time requires consideration of the value of search time in an unknown future depending on the moves the opponent (or nature) actually make. **(3)** The methods we tested can be seen as offline meta-reasoning compiled into a simple runtime rule. We also intend to examine wherther a direct runtime approximate solution of the belief-space MDP stated above will yield a better sampling scheme.

#agents	0 (a) always	1	5	10	100	500	never (CBS)
5	899	190	185	181	181	**180**	256
15	1,621	2,241	1,708	**1,702**	1,713	1,738	1,807
25	7,675	8,327	1,701	**1,620**	1,731	2,071	3,264
35	15,736	12,655	**4,974**	4,993	7,199	18,998	$> 50{,}050$

Table 1: Different fixed merging polices for MA-CBS

3.3 Scenario 3: Meta-agent conflict based search

The multi-agent path finding problem (MAPF) has attracted many researchers in the past decade. The task in MAPF is to find (optimal) collision-free paths for multiple agents, each with a different start and goal position. A straightforward approach to this problem is to run A* on a standard formalization of this task as a search problem.

Conflict-based search (CBS), and its enhancement *meta-agent conflict-based search* (MA-CBS) are a two-level algorithms that optimally solve MAPF [48, 49]. At the high-level, search is performed on a tree based on conflicts between agents. A conflict, labeled (a_i, a_j, l, t) is when the path of two agents a_i and a_j include the same location l at the same time t. When such a conflict is found the search branches to resolve this conflict. In the right (resp. left) branch, we add a constraint to agent a_i (resp., a_j) which prevents its being in location l at time t. The low level tries to find paths for individual agents that are consistent with its constraints. CBS proved to be very efficient in many cases. Meta-agent CBS (MA-CBS) adds another phase on top of CBS. A *meta-agent* is a generalization of an agent which includes k agents. It includes a vector of the ID of the k agents and a vector of their locations. When a meta-agent needs to move, all possible internal combinations for the different k agents are considered.

MA-CBS starts with only regular agents and proceeds in the same manner as CBS. However, if two agents are found to have many conflicts, MA-CBS may choose to merge them into a *meta-agent* instead of branching according to their last conflict as in CBS. MA-CBS then treats this meta-agent as just another agent in the system. Thus, at each conflict found there are two computational operators - 1) merge the conflicting agents 2) do not merge and continue to resolve the conflict. The main challenge in MA-CBS is to decide when to merge agents. The authors [48, 49] recognized that the $should - merge()$ function is critical. However, despite providing only simple rules for this decision, a dramatic speedup was obtained in many test cases.

The columns of table 1 compare different merging polices. These policy are simple ruled-based MR based on the number of conflicts seen between the two agents in question. If this number reaches a bound b we merge the agents. Each column corresponds to a different b value, where the extreme cases are 0 (always merge) and ∞ (never merge = CBS). Each line is for a different number of agents. Results show that the selective variants are much better than the two extremes. However, there was no universally winning value and for each number of agents there was a different best value for b. This calls for more sophisticated MR approaches, in particular better rule-based techniques, such as considering the relative density of obstacles and other agents in the area - as an indication for future conflicts. We will also attempt to use techniques from rational MR, as one can measure the distribution of conflicts based on the density and create a distribution and utility model for this decision.

3.4 Scenario 4: Constraint satisfaction

In this domain, most exhaustive search algorithms use (depth-first) backtracking with additional improvements. Several categories of heuristics, such as variable ordering heuristics and value ordering heuristics [66], are used. Here we examine the tradeoff between using expensive and cheap value-ordering heuristics (the computational operators in this case), and the role of meta-reasoning is to decide when to use the more expensive heuristic(s). An underlying assumption here is that the algorithm is a standard backtracking exhaustive search.

Preliminary results: A simple cheap heuristic (min conflict, denoted MC) was always used. The MR decision was about whether or not to evaluate an expensive value-ordering heuristic (SC) that estimates the number of solutions in a subtree of the search [39]. In this limited instantiation, we could define a simple appropriate utility model. The distributions in this model were defined in a very ad-hoc manner: for example, we assumed that running the heuristic will indeed provide a very good estimate on the number of solutions. Rather simplistic assumptions were made about the expected search time in each subtree. Table 2 summarizes results for one application domain - generalized Sudoku. Despite the very coarse assumptions, the resulting algorithm (VSC) performed considerably better than the following competitors: not using the expensive heuristic at all (denoted by MC in table 2), always using the expensive heuristic (SC in the table), and always using an even more expensive heuristic, probabilistic arc consistency (pAC). For completeness, we also examined whether simple rules (randomly or level-based) work, such as evaluating SC a fraction of time similar to that of our MR based scheme. The answer to that is negative, implying that the elaborate meta-reasoning is really needed.

	$\overline{T_{SC}}$, sec	$\overline{\left(\frac{T_{VSC}}{T_{SC}}\right)}$	$\overline{\left(\frac{T_{MC}}{T_{SC}}\right)}$	$\overline{\left(\frac{T_{pAC}}{T_{SC}}\right)}$
4x3, 90 holes	1.809	0.755	1.278	22.421
7x4, 357 holes	21.328	0.868	3.889	3.826

Table 2: Results on Generalized Sudoku. Relative performance to the baseline SC algorithm

Proposed extensions: We aim to focus on the following issues: improving the ad-hoc utility and distribution models, allowing meta-reasoning over additional types of heuristics (such as ones for variable ordering), both of which should further improve performance. In addition we mean to compare rational meta-reasoning to the method of just learning which heuristic to use at each stage, as done in *Selective Max* [10] for planning.

3.5 Scenario 5: Cost-aware search

In *cost-aware search* there is a tradeoff between the cost of the returned solution (the time to traverse the returned *solution path*, called *traversal time*) and time spent in search, the *computation time*. Consider a GPS application where the goal is to arrive at the destination as quickly as possible. In this case the traversal time (cost of a solution) is the estimated time it takes to follow the returned path. Consequently, if a search algorithm finds a path to the goal that has a traversal time of five minutes, there is no point in searching for a better path if one estimates that finding a better path will also take five minutes.

More formally, in *cost-aware search* the search algorithm is given a (user-defined) utility function $u(C, T)$, which defines the tradeoff between the traversal time C, and the computation time T. That is, $u(C, T)$ is the utility of finding a solution of cost C after searching for T time steps. An interesting question arises: suppose that a non-optimal solution of cost C' has been found. Should we spend time to search for a better solution?

Although many real-life search problems are in fact cost-aware, relatively little work has been done in cost-aware search, other than in the anytime algorithm framework [8, 75]. A pioneering algorithm for cost-aware search is Best-First Utility Guided Search (BUGSY) [44]. BUGSY expands the node with the highest utility, computed as a linear combination of the estimated lowest cost solution under the evaluated node, and the estimated computation time required to achieve that goal. The weights of this linear combination are parameters set according to user preferences. While important, BUGSY has several limitations that can be addressed more comprehensively with our MR approach. First, the way that BUGSY computes the solution cost and computation cost (BUGSY uses cost-to-go and distance-to-go estimates). More recent solution cost prediction algorithms [36, 37] and search effort estimators [62, 9] have been proposed to better estimate solution cost and computation time. Second, BUGSY only considers the estimated lowest cost solution under a node, rather than the distribution of possible goals beneath a given node. Additionally, BUGSY is myopic, considering only the utility of the next expansion.

Proposed extensions: In addition to selecting which node to expand, the issue of a stopping criterion is crucial. That is, suppose a solution of cost C was found, which action optimizes expected utility $E[u(C, T)]$: halting with the current solution, or continuing the search? The added utility gained from finding a solution with cost $< C$ is contrasted with the estimated decrease in utility caused by the time spend in finding that solution. This requires modeling the probability that a given node lead to a solution of a given cost, as well as a model of the search time required to achieve that solution.

We examined special cases of *cost-aware* search where such models are obtainable. Specifically, in *bounded-cost search* the aim is to find a solution with cost under a given bound B [54]. Potential Search (PTS) [54] is an appropriate algorithm that has a distribution model, but does not consider the costs of computation. We intend to apply rational MR as a control mechanism, both to improve PTS, and as a method of cost-aware search.

4 Principal investigators, collaboration and resources

Developing the MR framework requires considerable expertise in heuristic search, in uncertainty models, and in decision-making. The principal investigators have complementarily expertise in this regards. They have already made considerable progress in the direction proposed here and have realized that joining the different types of their expertise (theoretical and empirical/heuristic) greatly advances this research.

Eyal (Solomon) Shimony specializes in probabilistic reasoning and decision making. He heads the artificial intelligence lab at BGU-CS and has several graduate students to handle the empirical side of the work, including one already working on parts of the proposed research. His main task will be the mathematical contributions and theoretical aspects of this research, and *rational* meta-reasoning.

Ariel Felner specializes in general heuristic search algorithms and in creating good heuristics. Some of his students are already working on problems that are part of the proposal. Standard computing support is also available from the department labs (BGU-ISE). His main task will be the experimental aspects of this research.

Roni Stern is a young researcher specializing in search, especially search in non-trivial scenarios. At the time of submission he is a postDoc at Harvard, and a candidate for a tenure track position at BGU starting October 2013. Currently, he is formally ineligible to be a PI in this proposal. However, he has co-authored this proposal, and will participate in research just like the PIs. His main task is to apply the ideas to real-world search scenarios.

References

[1] Shahab Jabbari Arfaee, Sandra Zilles, and Robert C. Holte. Learning heuristic functions for large state spaces. *Artif. Intell.*, 175(16-17):2075–2098, 2011.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the Multiarmed bandit problem. *Mach. Learn.*, 47:235–256, May 2002.

[3] T. Bartz-Beielstein and S. Markon. Tuning search algorithms for real-world applications: a regression tree based approach. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 1111 – 1118, June 2004.

[4] Blai Bonet and Hector Geffner. Heuristic search planner 2.0. *AI Magazine*, 22(3):77–80, 2001.

[5] Teresa Breyer and R. E. Korf. Recent results in analyzing the performance of heuristic search. In *International Workshop on Search in Artificial Intelligence and Robotics (held in conjunction with AAAI)*, 2008.

[6] Teresa Maria Breyer and Richard E. Korf. Independent additive heuristics reduce search multiplicatively. In *AAAI*, 2010.

[7] Ethan Burns, J. Benton, Wheeler Ruml, Sung Wook Yoon, and Minh Binh Do. Anticipatory on-line planning. In *ICAPS*, 2012.

[8] Thomas Dean and Mark S. Boddy. An analysis of time-dependent planning. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *AAAI*, pages 49–54. AAAI Press / The MIT Press, 1988.

[9] Austin J. Dionne, Jordan Tyler Thayer, and Wheeler Ruml. Deadline-aware search using on-line measures of behavior. In *SOCS*, 2011.

[10] Carmel Domshlak, Erez Karpas, and Shaul Markovitch. To max or not to max: Online learning for speeding up optimal planning. In *National Conference on Artificial Intelligence (AAAI-07)*, 2010.

[11] A. Felner, M. Goldenberg, R. Stern, G. Sharon, T. Beja, R. Holte, J. Schaeffer, N. Sturtevant, and Z. Zhang. Partial-expansion a* with selective node generation. *Proc. AAAI, To appear.*, 2012.

[12] Ariel Felner. Finding optimal solutions to the graph partitioning problem with heuristic search. *Ann. Math. Artif. Intell.*, 45(3-4):293–322, 2005.

[13] Ariel Felner, Carsten Moldenhauer, Nathan R. Sturtevant, and Jonathan Schaeffer. Single-frontier bidirectional search. In *AAAI*, pages 59–65, 2010.

[14] David Furcy and Sven Koenig. Limited discrepancy beam search. In *IJCAI*, pages 125–131, 2005.

[15] Sylvain Gelly and David Silver. Achieving master level play in 9 9 computer go. *AAAI-08*, 2008.

[16] M. Goldenberg, A. Felner, N. R. Sturtevant, and J. Schaeffer. Portal-based true-distance heuristics for path finding. In *Second International Symposium on Combinatorial Search (SoCS)*, 2010.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SCC-4(2):100–107, 1968.

[18] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *UAI*, pages 123–132, 2012.

[19] Nicholas Hay and Stuart J. Russell. Metareasoning for Monte Carlo tree search. Technical Report UCB/EECS-2011-119, EECS Department, University of California, Berkeley, Nov 2011.

[20] D. Heckerman, E. Horvitz, and B. Middleton. An approximate nonmyopic computation for value of information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(3):292–298, 1993.

[21] Malte Helmert. *Understanding Planning Tasks: Domain Complexity and Heuristic Decomposition*, volume 4929 of *Lecture Notes in Computer Science*. Springer, 2008.

[22] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*, 2009.

[23] Malte Helmert, Gabriele Roger, and Erez Karpas. Fast downward stone soup: A baseline for building planner portfolios. In *PAL 2011: Proceedings of the 3rd Workshop on Planning and Learning at ICAPS '11*, pages 28–35, 2011.

[24] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, 14:253–302, 2001.

[25] Jörg Hoffmann and Bart Selman, editors. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.

[26] R. C. Holte, A. Felner, J. Newton, R. Meshulam, and D. Furcy. Maximizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence*, 170:1123–1136, 2006.

[27] R. C. Holte, J. Newton, A. Felner, R. Meshulam, and D. Furcy. Multiple pattern databases. In *International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 122–131, 2004.

[28] Erez Karpas and Carmel Domshlak. Optimal search with inadmissible heuristics. In *ICAPS*, 2012.

[29] Erez Karpas, Michael Katz, and Shaul Markovitch. When optimal is just not good enough: Learning fast informative action cost partitionings. In *ICAPS*, 2011.

[30] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *ECML*, pages 282–293, 2006.

[31] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[32] R. E. Korf, M. Reid, and S. Edelkamp. Time complexity of iterative-deepening-A*. *Artificial Intelligence*, 129(1–2):199–218, 2001.

[33] Richard E. Korf. An improved algorithm for optimal bin packing. In *IJCAI*, pages 1252–1258, 2003.

[34] Richard E. Korf. Multi-way number partitioning. In *IJCAI*, pages 538–543, 2009.

[35] Andreas Krause and Carlos Guestrin. Optimal nonmyopic value of information in graphical models – efficient algorithms and theoretical limits. In *Proc. of IJCAI*, pages 1339–1345, 2005.

[36] Levi Lelis, Roni Stern, Ariel Felner, Sandra Zilles, and Robert C. Holte. Predicting optimal solution cost with bidirectional stratified sampling. In *ICAPS*, 2012.

[37] Levi H. S. Lelis, Roni Stern, Ariel Felner, Sandra Zilles, and Robert C. Holte. Predicting optimal solution cost with bidirectional stratified sampling (abstract). In *SOCS*, 2012.

[38] Marco Lippi, Marco Ernandes, and Ariel Felner. Efficient single frontier bidirectional search. In *SOCS*, 2012.

[39] Amnon Meisels, Solomon Eyal Shimony, and Gadi Solotorevsky. Bayes networks for estimating the number of solutions of constraint networks. *Ann. Math. Artif. Intell.*, 28(1-4):169–186, 2000.

[40] Sergio Núñez, Daniel Borrajo, and Carlos Linares López. Performance analysis of planning portfolios. In *SOCS*, 2012.

[41] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204, 1970.

[42] John R. Rice. The algorithmselectionproblem. *Advances in Computers*, 15:65–118, 1976.

[43] Gabriele Röger and Malte Helmert. The more, the merrier: Combining heuristic estimators for satisficing planning. In *ICAPS*, pages 246–249, 2010.

[44] Wheeler Ruml and Minh Binh Do. Best-first utility-guided search. In *IJCAI*, pages 2378–2384, 2007.

[45] Stuart Russell and Eric Wefald. *Do the right thing: studies in limited rationality*. MIT Press, Cambridge, MA, USA, 1991.

[46] Stuart J. Russell and Eric Wefald. On optimal game-tree search using rational meta-reasoning. In *Proc. of IJCAI*, pages 334–340, 1989.

[47] Jendrik Seipp, Manuel Braun, Johannes Garimort, and Malte Helmert. Learning portfolios of automatically tuned planners. In *ICAPS*, 2012.

[48] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *AAAI*, 2012.

[49] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Meta-agent conflict-based search for optimal multi-agent path finding. In *SOCS*, 2012.

[50] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 520–527, Arlington, Virginia, United States, 2004. AUAI Press.

[51] Roni Stern, Ariel Felner, and Robert Holte. Probably approximately correct heuristic search. In *SOCS*, 2011.

[52] Roni Stern, Tamar Kulberis, Ariel Felner, and Robert Holte. Using lookaheads with optimal best-first search. In *AAAI*, pages 185–190, 2010.

[53] Roni Tzvi Stern, Ariel Felner, and Robert C. Holte. Search-aware conditions for probably approximately correct heuristic search. In *SOCS*, 2012.

[54] Roni Tzvi Stern, Rami Puzis, and Ariel Felner. Potential search: A bounded-cost search algorithm. In *ICAPS*, 2011.

[55] Nathan R. Sturtevant, Ariel Felner, Max Barrer, Jonathan Schaeffer, and Neil Burch. Memory-based heuristics for explicit state spaces. In *IJCAI*, pages 609–614, 2009.

[56] Nathan R. Sturtevant, Ariel Felner, Maxim Likhachev, and Wheeler Ruml. Heuristic search comes of age. In *AAAI*, 2012.

[57] J. Tyler Thayer and W. Ruml. Faster than weighted a*: An optimistic approach to bounded suboptimal search. In *ICAPS*, pages 355–362, 2008.

[58] J. Tyler Thayer and W. Ruml. Using distance estimates in heuristic search. In *ICAPS*, 2009.

[59] Jordan Tyler Thayer and Wheeler Ruml. Finding acceptable solutions faster using inadmissible information. In *SOCS*, 2010.

[60] Jordan Tyler Thayer and Wheeler Ruml. Bounded suboptimal search: A direct approach using inadmissible estimates. In *IJCAI*, pages 674–679, 2011.

[61] Jordan Tyler Thayer, Roni Stern, Ariel Felner, and Wheeler Ruml. Faster bounded-cost search using inadmissible estimates. In *ICAPS*, 2012.

[62] Jordan Tyler Thayer, Roni Stern, and Levi H. S. Lelis. Are we there yet? - estimating search progress. In *SOCS*, 2012.

[63] David Tolpin and Solomon Eyal Shimony. Rational deployment of CSP heuristics. In Toby Walsh, editor, *IJCAI*, pages 680–686. IJCAI/AAAI, 2011.

[64] David Tolpin and Solomon Eyal Shimony. MCTS based on simple regret. In Hoffmann and Selman [25].

[65] David Tolpin and Solomon Eyal Shimony. Semimyopic measurement selection for optimization under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(2):565–579, 2012.

[66] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[67] Richard Anthony Valenzano, Shahab Jabbari Arfaee, Jordan Tyler Thayer, and Roni Stern. Alternative forms of bounded suboptimal search. In *SOCS*, 2012.

[68] Richard Anthony Valenzano, Hootan Nakhost, Martin Müller, Jonathan Schaeffer, and Nathan R. Sturtevant. Arvandherd: Parallel planning with a portfolio. In *ECAI*, pages 786–791, 2012.

[69] Richard Anthony Valenzano, Nathan R. Sturtevant, Jonathan Schaeffer, Karen Buro, and Akihiro Kishimoto. Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In *ICAPS*, pages 177–184, 2010.

[70] Christopher Makoto Wilt and Wheeler Ruml. When does weighted a* fail? In *SOCS*, 2012.

[71] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[72] U. Zahavi, A. Felner, N. Burch, and R. C. Holte. Predicting the performance of IDA* (with BPMX) with conditional distributions. *JAIR*, 37:41–83, 2010.

[73] Lei Zhang and Fahiem Bacchus. MAXSAT heuristics for cost optimal planning. In Hoffmann and Selman [25].

[74] Rong Zhou and Eric A. Hansen. Beam-stack search: Integrating backtracking with beam search. In *ICAPS*, pages 90–98, 2005.

[75] Shlomo Zilberstein. *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1993.