Domain-Dependent and Domain-Independent Problem Solving Techniques

Roni Stern
Ben-Gurion University of the Negev
sternron@post.bgu.ac.il

Abstract

Heuristic search is a general problem-solving method. Some heuristic search algorithms, like the well-known A* algorithm, are *domain-independent*, in the sense that their knowledge of the problem at-hand is limited to the (1) initial state, (2) state transition operators and their costs, (3) goal-test function, and (4) black-box heuristic function that estimates the value of a state. Prominent examples are A* and Weighted A*. Other heuristic search algorithms are domain-dependent, that is, customized to solve problems from a specific domain. A well-known example is conflict-directed A*, which is specifically designed to solve model-based diagnosis problems. In this paper, we review our recent advancements in both domain-independent and domain-dependent heuristic search, and outline several challenging open questions.

1 Introduction

Heuristic search is one of the fundamental problem-solving techniques used in Artificial Intelligence (AI), dating back to Newell and Simon's General Problem Solving program [Newell *et al.*, 1959]. Heuristic search algorithms, such as A*, are still the cornerstone of performing higher-level AI tasks such as automated planning [Bonet and Geffner, 2001] and Model-Based Diagnosis (MBD) [Reiter, 1987; De Kleer and Williams, 1987].

When solving a problem using a search algorithm, the world is abstracted as a set of *states*. Every state s is associated with a set of *state-transition operators* $O(s)$, where each $o \in O(s)$ is a function that maps s to a possibly different state s'. A *path* from state s to s' is a sequence of state-transition operators such that applying this sequence to s results in s'. It is common for operators to have a cost, and to define the *cost of a path* as the sum of costs of its constituent operators. *Path finding* is a common type of search problem where the objective is to find a path from a given initial state to a *goal state*. A goal state is defined explicitly or implicitly via a goal-test function. Classical STRIPS-based planning is a prime example of a path finding problem [Bonet and Geffner, 2001; Fikes and Nilsson, 1971]. A common requirement in path finding problems is that the path has a low *cost*. An *optimal* solution in such cases is a lowest-cost path from the initial state to a goal state.

A search algorithm is called a *heuristic* search algorithm if it considers some imperfect source of information about the problem it is used to solve. In some heuristic search algorithms, this imperfect information is encapsulated in a black-box heuristic function h that accepts a state s and outputs a number that estimates the cost of a lowest-cost path from s to a goal. We refer to such algorithms as *domain-independent* heuristic search algorithms. We use this term to emphasize that all the problem solver knows about the domain beyond the initial state, operators, and goal-test function, is a black-box heuristic function.

Prominent examples are A* [Hart *et al.*, 1968] and wA* [Pohl, 1970]. Other heuristic search algorithms are domain-dependent, in the sense that their behavior is customized to solve a specific type of problems. For example, Conflict-Directed A* [Williams and Ragno, 2007] is a heuristic search algorithm that is specifically designed to solve MBD problems. Similarly, Increasing Cost Tree Search [Sharon *et al.*, 2013] is specifically designed to solve Multi-Agent Pathfinding (MAPF) problems.

In this paper, we review our contributions in developing algorithms and theory for domain-independent and domain-dependent heuristic search. In particular, we describe several recent algorithms aimed at satisfying various requirements about solution quality, and review a range of domain-dependent heuristic search algorithms we developed. Throughout, we highlight open challenges and topics that are left for future research.

2 Background

Best-first search is a general framework for solving search problems. A best-first search maintains a list of states called OPEN. Initially, OPEN contains the initial state. Then, in every iteration, a single state s is removed from OPEN and *expanded*. To expand a state s means that for every operator $o \in O(s)$ we *generate* a new state $o(s)$ and insert it into OPEN. Later iterations can choose to expand these new states, generating newer states. The search can halt when a goal state is generated.

There are many design choices when implementing a best-first search, e.g., whether to insert to OPEN a state that was

already generated by another state. A particularly important design choice is how to choose which state to remove from OPEN in every iteration. *Heuristic search* algorithm use the heuristic function h to make this choice. For example, the famous A* algorithm chooses from OPEN the state s with the smallest $f(s) = g(s) + h(s)$, where $g(s)$ is the cost of the lowest-cost path known so far from the initial state to s. A heuristic h is *admissible* if for every state s it outputs a value that is smaller than or equal to the cost of a lowest-cost path from s to a goal. Given an admissible heuristic, A* is guaranteed to have found an optimal solution once a goal node is expanded [Hart *et al.*, 1968]. Most heuristic search algorithms that return optimal solutions also rely on an admissible heuristic, e.g., IDA* [Korf, 1985] and RBFS [Korf, 1992].

3 Domain-Independent Search

Many search problems are just too hard to solve optimally, even with A* and a very accurate heuristic [Helmert *et al.*, 2008]. To this end, a range of suboptimal search algorithms has been developed. One way to classify suboptimal search algorithms is by the theoretical guarantee they provide over the solution they return.

3.1 Incomplete and Not-Optimal Search

Some search algorithms are incomplete, i.e., they may not find a solution even if such exists. Hill Climbing and Genetic Algorithms [Holland, 1992] are examples of incomplete algorithms. Such algorithms are used for extremely difficult search problems.

Other search algorithms are guaranteed to find a solution if such exists, but provide no guarantee regarding the quality of that solution. Greedy Best-First Search (GBFS) [Doran and Michie, 1966], also known as Pure heuristic search, is a popular example. GBFS is a best-first search that expands in every iteration the state in OPEN with the smallest heuristic value. Beam search is another popular algorithm from this class [Zhou and Hansen, 2005; Sabuncuoglu and Bayiz, 1999; Furcy and Koenig, 2005]. There is a limited understanding of the behavior of these algorithms, although recent research has been trying to identify the relation between their performance and the heuristic landscape [Wilt and Ruml, 2016; Heusner *et al.*, 2018].

In many cases, it is not sufficient to find any solution, and there are some restrictions on the cost of the desired solution. We have explored several such cost-related restrictions.

3.2 Bounded-Cost Search

In a *bounded-cost search* problem, the user defines a cost bound C and aims to find a solution whose cost is at most C. This type of search problem arises, for example, in scenarios where there is a limited budget. Bounded-cost search problem has rarely been studied in a domain-independent manner.

We closed this gap by proposing a search algorithm called Potential Search (PS) [Stern *et al.*, 2014] that is specifically designed to solve bounded-cost search problems. PS is a best-first search based on a unique evaluation function $u(s) = \frac{C-g(s)}{h(s)}$. In every iteration, PS expands the state with the highest $u(s)$. The $u(\cdot)$ evaluation function was discovered independently by two research groups [Stern *et al.*, 2011; Van Den Berg *et al.*, 2011] in the context of *anytime search* (see later) and through different mathematical derivations.

The derivation relevant to bounded-cost search relates $u(s)$ to the *potential* of s. The potential of a state with respect to C is the probability that it part of a solution of cost lower than C. Under certain probabilistic relation between the available heuristic function and the cost it estimates, it can be shown that the state with the highest $u(\cdot)$ in OPEN is also the one with the highest potential. Empirically, we observed PS to be useful over a range of search domains. However, we also observed that PS may perform poorly on domains in which the *length* of a path, i.e., the number of its constituent operators, is not correlative with its cost.

For such domains, we proposed Bounded-cost Explicit Estimation Search (BEES) [Thayer *et al.*, 2012]. BEES is based on Explicit Estimation Search (EES) [Thayer and Ruml, 2011], and considers additional heuristics including a heuristic that estimates the length of an optimal path to a goal. Empirical evaluations showed that BEES is particularly useful in these non-uniform domains. An open research challenge is to provide a more rigorous theory for the behavior of PS and BEES, and in general to bounded-cost search problems.

3.3 Bounded-Suboptimal Search

A common way to quantify the *suboptimality* a solution is by measuring the ratio between between its cost and the cost of an optimal solution. This is often called the *suboptimality* of a solution.[1] The suboptimality of an optimal solution is one, and higher suboptimality means a worse solution (i.e., higher cost). A *bounded-suboptimal* search algorithm is an algorithm that accepts a parameter $B \geq 1$ and is guaranteed to return a solution whose suboptimality is at most B. An ideal bounded-suboptimal search algorithm trades off solution cost for runtime, i.e., increasing B results in worse (higher) solution cost and better (lower) search runtime.

wA* is a well-known bounded-suboptimal search algorithm [Pohl, 1970] that is still widely used. It expands in every iteration the state n in OPEN with the smallest $g(n) + B \cdot h(n)$.

A$^*_\epsilon$ [Pearl and Kim, 1982] is another well-known bounded-suboptimal search algorithm. In A$^*_\epsilon$, the minimal f value in OPEN, referred to as f_{min}, is maintained. f_{min} is used to define FOCAL, which is the set containing every state s in OPEN for which $f(s) \leq B \cdot f_{min}$. A$^*_\epsilon$ chooses in every iteration to expand a state from FOCAL. To choose which state in FOCAL to expand, A$^*_\epsilon$ uses a different, possibly inadmissible heuristic. Since A$^*_\epsilon$ does not define this secondary heuristic, it is, in fact, a search framework, which is known today as *Focal search* [Thayer and Ruml, 2011; Ebendt and Drechsler, 2009].

All Focal search algorithms are bounded-suboptimal because throughout the search f_{min} is always a lower bound on the cost of an optimal solution, given that f is computed with

[1] Other ways to define suboptimality of a solution exist, e.g., the difference from the optimal solution cost [Valenzano *et al.*, 2013].

an admissible heuristic. This understanding allowed us to develop a novel bounded-suboptimal search algorithm called Dynamic Potential Search (DPS). DPS considers both f_{min} and the $u(\cdot)$ function used by PS [Gilon *et al.*, 2016]. DPS is a best-first search that expands in every iteration the state with the highest $u(\cdot)$. Since a cost bound C is not given in a bounded-suboptimal search problem, DPS sets C to be $B \cdot f_{min}$. This guarantees that any solution found will indeed be bounded suboptimal. If f_{min} changes, DPS will re-sort OPEN to update the u values. In a sense, DPS is running a sequence of bounded-cost searches, where the cost bound is $B \cdot f_{min}$. A major advantage of DPS over Focal search algorithms is that it does not need to maintain FOCAL. Empirically, DPS performs well on a range of domains.

DPS, as well as EES, wA*, and A*$_\epsilon$, are all in the best-first search framework. Thus, there are cases where memory will be a bottleneck to the search. To address this, there are several modern bounded-suboptimal search algorithms that run in linear space [Hatem and Ruml, 2014; Hatem *et al.*, 2013] and outperform traditional bounded-suboptimal linear-space algorithms such as weighted IDA* and weighted RBFS [Korf, 1992].

Indeed, the past decade has seen an explosion of bounded-suboptimal search algorithms. However, the theoretical understanding of these algorithms is, to-date, limited. For example, in optimal search, under some conditions, any best-first search must expand at least the set of nodes expanded by A* [Dechter and Pearl, 1985; Goldenberg *et al.*, 2014; Holte and Zilles, 2019]. An equivalent claim does not exist for bounded-suboptimal search. Also, there is no clear answer for which bounded-suboptimal algorithm to use in a given domain. There are important directions for future work.

3.4 Anytime Search

An *anytime* algorithm is an algorithm "whose quality of results improves gradually as computation time increases" [Zilberstein, 1996]. An anytime search algorithm finds an initial solution quickly, and then, given more running time, finds better solutions. Several anytime search algorithms have been proposed over the years. Prominent examples are Anytime Repairing A* (ARA*) [Likhachev *et al.*, 2004], Anytime Weighted A* (AWA*) [Hansen and Zhou, 2007], Anytime Windowed A* [Aine *et al.*, 2007], and Restarting Weighted A* [Richter *et al.*, 2010]. A major limitation of all these algorithms is that they accept a parameter and there is no clear theory on how to set it. To address this, we developed a *non-parametric* anytime search algorithm, that works as follows. First, it finds an initial solution with GBFS. Then, we run PS and set the cost-bound to be the cost of the incumbent solution. This will return a new, better, solution. Then, we run PS again with the cost of the new solution. This process continues until time has run out or the optimal solution has been found. The resulting anytime search algorithm is, to-date, one of the best anytime search algorithms.[2]

This anytime algorithm and the bounded-suboptimal algorithm DPS are both built on top of PS. However, they demonstrate a more general approach in which one can use a bounded-cost search algorithm to construct an anytime search algorithm as well as a bounded-suboptimal search algorithm. Thus, future work on developing better bounded-cost search algorithms can have a wide impact beyond solving bounded-cost search problems.

[2]Due to its dual origin, this algorithm has two names: Anytime PS or Anytime Non-Parametric A* (ANA*).

3.5 Probably Bounded-Suboptimal Search

Recently, we explored the notion of *probabilistic* solution quality guarantees, where the suboptimality of a solution must be bounded in *most* cases, but not always [Stern *et al.*, 2019]. More formally, a Probably Bounded-Suboptimal (PBS) search algorithm is an algorithm that accepts, in addition to a suboptimality bound B, a *confidence* parameter δ. The guarantee provided by a PBS is that the suboptimality of the solution it returns is at most B with probability at least $1-\delta$. We proposed several PBS algorithms based on running an anytime search algorithm and analyzing the domain and previously solved problems in it. Experimentally, we showed that allowing a non-zero δ, i.e., a non-zero probability of returning a solution without the desired suboptimality, can lead to significant search speedup.

The PBS algorithms we proposed can be viewed as a middle-ground between domain-independent and domain-specific search. Indeed, we envision such forms of analysis of the domain can provide a bridge to introducing data-driven methods into heuristic search algorithms that provide some form of solution quality guarantees.

4 Domain-Dependent Search

Domain-independent heuristic search algorithms, such as those listed in the previous section, are widely used in a range of domains. However, an in-depth understanding of the problem domain is needed in some domains in order to guide the search effectively. In this section, we describe a partial list of domain-specific search problems that we have worked on.

4.1 Multi-Agent Pathfinding

A MAPF problem consists of a graph and n agents. Each agent has a unique start node and a unique goal node. Time is discretized into time steps. In each time step, each agent can either move to an adjacent node or wait in its current node. A *plan* for a single agent is a sequence of (more or wait) actions that moves the agent from its start to its goal. Two agents cannot occupy the same node at the same time, as that would cause a collision. A valid solution to a MAPF problem is a *joint plan*, which is a set of plans, one for each agent, such that no collisions will occur. MAPF has topical applications in warehouse management [Wurman *et al.*, 2008], airport towing [Morris *et al.*, 2016], autonomous vehicles, robotics [Veloso *et al.*, 2015], and digital entertainment [Ma *et al.*, 2017].

In many cases, there is also a requirement to minimize some cumulative cost function, such as the time spent or costs incurred until all agents have reached their goals. Solving MAPF optimally is NP-Hard [Yu and LaValle, 2013; Surynek, 2010]. MAPF is particularly challenging for domain-independent search algorithms because the number

of operators in a given state is exponential in the number of agents, since agents can move concurrently.

To address this, we developed the Enhanced Partial Expansion A* (EPEA*) algorithm [Goldenberg *et al.*, 2014], that uses a domain-specific *operator selection function* to choose to use only a subset of the applicable operators when a state is expanded. EPEA* is, in fact, applicable to any problem with a large branching factor. We also developed more sophisticated, domain-specific MAPF, that solve MAPF by solving two different search problems that interact with each other. For more details, see Felner et al. [2017].

4.2 Longest Simple Path

The Longest Simple Path (LSP) problem is the problem of finding the longest *simple* path in a graph from one node to another. A path is *simple* iff no node appears twice in the path. LSP is a classical graph theory problem with various applications. A key challenge in LSP is that it is *non-monotonic*, i.e., longer paths are better. This makes it harder to identify when an optimal solution has been found [Stern *et al.*, 2014]. We describe how to adapt A* to such problems, and propose several heuristics for LSP, and for a particular type of LSP called *snake in the box*, which has application in coding [Palombo *et al.*, 2015].

A second challenge in LSP is the simple-path requirement. This requirement prohibits pruning multiple paths that reach the same node, thereby making the search much harder. A similar challenge exists in other search problems, such as the target-value search [López *et al.*, 2013]. Creating a general, domain-independent algorithm for problems with this requirement, is an open challenge.

4.3 Model-Based Diagnosis

An MBD problem occurs when an observation of a system is inconsistent with an available model of that system and the assumption that all the system components are not faulty. The task of a consistency-based MBD algorithm is to infer *diagnoses*, which are assumptions about which system components are faulty that explain the observation.

MBD is a classical problem in the Artificial Intelligence literature with numerous applications [Reiter, 1987; De Kleer and Williams, 1987]. MBD can be modeled as a search problem. A state is an assumption about which component is faulty. The initial state assumes all components are not faulty. A state transition operator adds an assumption that one component is faulty. A goal-test function checks if the assumption represented by a state is consistent with the observation and the available system model.

MBD is a challenging problem because the number of operators applicable in a state is the number of system components. A common way to address this is to limit the search using *conflicts*. A conflict is a set of components that contain at least one faulty component. Conflict-directed A* [Williams and Ragno, 2007] and other algorithms [Reiter, 1987; De Kleer and Williams, 1987] work by identifying conflicts, and then limiting the search to only consider hitting sets of identified conflicts.

Identifying conflicts, however, is a hard problem in its own right. To address this, we proposed an MBD-specific search algorithm that runs two searches in parallels: one searching for conflicts and the other searching for diagnoses [Stern *et al.*, 2012]. Due to the duality between conflicts and diagnoses, these searches effectively interact, where finding diagnoses helps finding conflicts and vice versa.

5 Conclusion

Recent years have shown tremendous progress in both domain-independent and domain-dependent heuristic search algorithms. We believe that cross-fertilization between domain-independent and domain-dependent search algorithms can lead to impactful advancement in the field. This calls for automating the process of learning domain properties and extracting useful information from them that can be used in search.

Acknowledgments

The contributions in this paper are a result of collaborations with many other researchers. Special thanks go to Ariel Felner, my long-time mentor and friend, as well as to Meir Kalech, Rami Puzis, and Robert Holte, for their key role in this research and my academic education. This research is supported by ISF grants no. 210/17 to Roni Stern.

References

[Aine *et al.*, 2007] Sandip Aine, PP Chakrabarti, and Rajeev Kumar. AWA*-a wind constrained anytime heuristic search algorithm. In *IJCAI*, 2007.

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2), 2001.

[De Kleer and Williams, 1987] Johan De Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial intelligence*, 32(1):97–130, 1987.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[Doran and Michie, 1966] James E Doran and Donald Michie. Experiments with the graph traverser program. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 294(1437):235–259, 1966.

[Ebendt and Drechsler, 2009] Rüdiger Ebendt and Rolf Drechsler. Weighted A* search - unifying view and application. *Artif. Intell.*, 173(14):1310–1342, 2009.

[Felner *et al.*, 2017] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan R. Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *the International Symposium on Combinatorial Search (SoCS)*, pages 29–37, 2017.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

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

[Gilon *et al.*, 2016] D. Gilon, A. Felner, and R. Stern. Dynamic potential search - A new bounded suboptimal search. In *Symposium on Combinatorial Search (SOCS)*, pages 36–44, 2016.

[Goldenberg *et al.*, 2014] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan Sturtevant, Robert C Holte, and Jonathan Schaeffer. Enhanced partial expansion A*. *Journal of Artificial Intelligence Research*, 50:141–187, 2014.

[Hansen and Zhou, 2007] Eric A Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28:267–297, 2007.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[Hatem and Ruml, 2014] Matthew Hatem and Wheeler Ruml. Bounded suboptimal search in linear space: New results. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

[Hatem *et al.*, 2013] Matthew Hatem, Roni Stern, and Wheeler Ruml. Bounded suboptimal heuristic search in linear space. In *Sixth Annual Symposium on Combinatorial Search*, 2013.

[Helmert *et al.*, 2008] Malte Helmert, Gabriele Röger, et al. How good is almost perfect?. In *AAAI*, volume 8, 2008.

[Heusner *et al.*, 2018] Manuel Heusner, Thomas Keller, and Malte Helmert. Best-case and worst-case behavior of greedy best-first search. In *IJCAI*, 2018.

[Holland, 1992] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[Holte and Zilles, 2019] Robert C Holte and Sandra Zilles. On the optimal efficiency of cost-algebraic A*. In *AAAI*, 2019.

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

[Korf, 1992] Richard E. Korf. Linear-space best-first search: Summary of results. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI, pages 533–538, 1992.

[Likhachev *et al.*, 2004] Maxim Likhachev, Geoffrey J. Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *NIPS*, 2004.

[López *et al.*, 2013] Carlos Linares López, Roni Stern, and Ariel Felner. Target-value search revisited. In *IJCAI*, 2013.

[Ma *et al.*, 2017] H. Ma, J. Yang, L. Cohen, T. K. S. Kumar, and S. Koenig. Feasibility study: Moving non-homogeneous teams in congested video game environments. In *AIIDE*, 2017.

[Morris *et al.*, 2016] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, TK S. Kumar, and S. Koenig. Planning, scheduling and monitoring for airport surface operations. In *AAAI Workshop*, 2016.

[Newell *et al.*, 1959] Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *International Conference on Information Processing*, 1959.

[Palombo *et al.*, 2015] Alon Palombo, Roni Stern, Rami Puzis, Ariel Felner, Scott Kiesel, and Wheeler Ruml. Solving the snake in the box problem with heuristic search: First results. In *Symposium on Combinatorial Search (SoCS)*, 2015.

[Pearl and Kim, 1982] Judea Pearl and Jin H. Kim. Studies in semi-admissible heuristics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(4):392–399, 1982.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4), 1970.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[Richter *et al.*, 2010] Silvia Richter, Jordan Tyler Thayer, and Wheeler Ruml. The joy of forgetting: Faster anytime search via restarting. In *ICAPS*, pages 137–144, 2010.

[Sabuncuoglu and Bayiz, 1999] Ihsan Sabuncuoglu and M Bayiz. Job shop scheduling with beam search. *European Journal of Operational Research*, 118(2):390–412, 1999.

[Sharon *et al.*, 2013] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 2013.

[Stern *et al.*, 2011] Roni Stern, Rami Puzis, and Ariel Felner. Potential search: a bounded-cost search algorithm. In *ICAPS*, 2011.

[Stern *et al.*, 2012] Roni Stern, Meir Kalech, Alexander Feldman, and Gregory Provan. Exploring the duality in conflict-directed model-based diagnosis. In *AAAI*, 2012.

[Stern *et al.*, 2014] Roni Stern, Ariel Felner, Jur van den Berg, Rami Puzis, Rajat Shah, and Ken Goldberg. Potential-based bounded-cost search and anytime non-parametric A*. *Artificial Intelligence*, 214:1–25, 2014.

[Stern *et al.*, 2019] Roni Stern, Gal Dreiman, and Richard Valenzano. Probably bounded suboptimal heuristic search. *Artificial Intelligence*, 267:39–57, 2019.

[Surynek, 2010] Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *AAAI*, 2010.

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

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

[Valenzano *et al.*, 2013] Richard Anthony Valenzano, Shahab Jabbari Arfaee, Jordan Tyler Thayer, Roni Stern, and Nathan R Sturtevant. Using alternative suboptimality bounds in heuristic search. In *ICAPS*, 2013.

[Van Den Berg *et al.*, 2011] Jur Van Den Berg, Rajat Shah, Arthur Huang, and Ken Goldberg. Anytime nonparametric A*. In *AAAI*, 2011.

[Veloso *et al.*, 2015] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal. Cobots: Robust symbiotic autonomous mobile service robots. In *IJCAI*, page 4423, 2015.

[Williams and Ragno, 2007] Brian C Williams and Robert J Ragno. Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics*, 155(12):1562–1595, 2007.

[Wilt and Ruml, 2016] Christopher Wilt and Wheeler Ruml. Effective heuristics for suboptimal best-first search. *Journal of Artificial Intelligence Research*, 57:273–306, 2016.

[Wurman *et al.*, 2008] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 2008.

[Yu and LaValle, 2013] Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, 2013.

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

[Zilberstein, 1996] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996.