The Factored Shortest Path Problem and Its Applications in Robotics

Zhi Wang   Liron Cohen   Sven Koenig   T. K. Satish Kumar
University of Southern California, Los Angeles, California 90089, USA
{zhiwang,lironcoh,skoenig}@usc.edu tkskwork@gmail.com

Abstract

Many real-world combinatorial problems exhibit structure in the way in which their variables interact. Such structure can be exploited in the form of "factors" for representational as well as computational benefits. Factored representations are extensively used in probabilistic reasoning, constraint satisfaction, planning, and decision theory. In this paper, we formulate the *factored shortest path problem* (FSPP) on a collection of constraints interpreted as factors of a high-dimensional map. We show that the FSPP is not only a generalization of the regular shortest path problem but also particularly relevant to robotics. We develop factored-space heuristics for A* and prove that they are admissible and consistent. We provide experimental results on both random and hand-crafted instances as well as on an example robotics domain to show that A* with factored-space heuristics outperforms A* with the Manhattan Distance heuristic in many cases.

Introduction

Real-world combinatorial problems exhibit structure. Algorithms for solving combinatorial problems that exploit the specific nature of interactions between variables are computationally much more efficient than generic algorithms. Examples include polynomial-time convex programming solvers (Nesterov and Nemirovskii 1994), the polynomial-time algorithm for 2-SAT (Aspvall, Plass, and Tarjan 1979), and the Kalman filter (Kalman 1960).

Even when the structure of how variables interact is not apparent, one can still exploit which variables interact with each other, without heeding to the exact nature of these interactions. Typically, each variable interacts with a limited set of other variables and maintains "locality." A set of interacting variables is referred to as a "factor." This kind of structure has been exploited in almost all areas of Artificial Intelligence (AI). Popular examples include Belief Networks in probabilistic reasoning (Cooper 1990), Markov Random Fields in computer vision (Li 1994), constraint networks in constraint reasoning (Dechter 2003), causal graphs in domain-independent planning (Brafman and Domshlak 2006; 2008), and factored Markov Decision Processes in learning and decision theory (Boutilier, Dean, and Hanks 1999; Guestrin, Koller, and Parr 2002).

However, not all aspects of structure have been fully exploited in some fundamental combinatorial problems. In particular, the factored form of the well-known shortest path problem is understudied despite its common occurrence in robotics and other domains.

In this paper, we therefore formulate the *factored shortest path problem* (FSPP) on a collection of constraints interpreted as factors of a high-dimensional map. We show that the FSPP is not only a generalization of the regular shortest path problem but also particularly relevant to robotics. In fact, it arises frequently in configuration space planning and path planning in the presence of obstacles.

Current state-of-the-art methods for high-dimensional path planning in robotics are primarily sampling-based methods. These include rapidly-exploring random trees (LaValle and Kuffner 2001), probabilistic roadmaps (Kavraki et al. 1996), and more recent algorithms such as those in (Persson and Sharf 2014). Despite their success, sampling-based methods are only probabilistically complete, i.e., are complete only when the number of samples approaches infinity. Moreover, they do not produce the same results in different runs when randomization is used, which is undesirable in many application domains. Deterministic complete search methods address both drawbacks but are not computationally viable unless they are guided by strong heuristics.

In this paper, we show that the FSPP can model many high-dimensional path-planning problems. To solve it, we develop factored-space heuristics for A* and prove that they are admissible and consistent. We provide experimental results on both random and hand-crafted instances as well as on an example robotics domain to show that A* with factored-space heuristics outperforms A* with the Manhattan Distance heuristic in many cases. Our heuristics are presented mostly as a strawman technique for the purpose of gaining interest among AI and robotics researchers to study the FSPP more closely.

The Factored Shortest Path Problem

In this section, we formalize the FSPP based on the standard formalization of constraint satisfaction problems (CSPs). We then comment on why it is imperative for us to compare it with other combinatorial problems for which efficient algorithms that exploit structure have been developed.

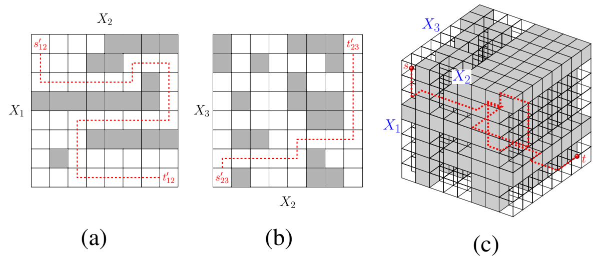

Figure 1: Illustrates an FSPP formed with two factors. Factors (a) and (b) are constraints $C_1(X_1, X_2)$ and $C_2(X_2, X_3)$, respectively. The rows and columns indicate different domain values of the corresponding variable; and the shaded cells correspond to infeasible assignments. (c) shows the implicit global constraint $\Omega = C_1 \wedge C_2$. s and t denote the two feasible assignments in Ω between which a shortest path is to be found. The dotted red line in (c) shows the shortest path between them, which is not a simple combination of the shortest paths between their projections onto the factors (a) and (b).

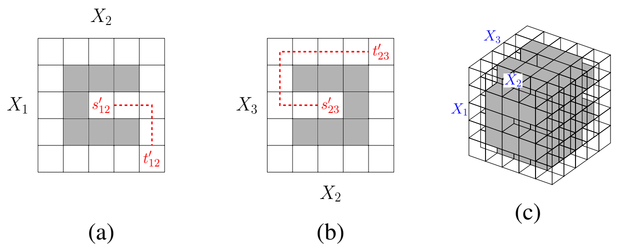

Figure 2: Illustrates an FSPP where a path between s and t does not even exist in Ω but exists in each of its factors. Here, (a) and (b) are the factors of (c).

A CSP is defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is a set of N variables, $\mathcal{D} = \{D(X_1), D(X_2), \ldots, D(X_N)\}$ is a set of N domains with discrete values, and $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ is a set of M constraints. Each C_i consists of a subset $S(C_i)$ of $\mathcal{X}$ and a list of allowed assignments of values to these variables chosen from their domains. In the CSP, the task is to find an assignment of values to all variables in $\mathcal{X}$ such that all constraints are satisfied, i.e., allow the assignment.

In the FSPP, given the same triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, we are interested in the global constraint Ω defined implicitly to be $C_1 \wedge C_2 \wedge \ldots \wedge C_M$. Here, $S(\Omega) = S(C_1) \cup S(C_2) \cup \ldots \cup S(C_M) = \mathcal{X}$, assuming that each variable in $\mathcal{X}$ appears in at least one constraint. Given two satisfying assignments $s = \langle X_1 = x_1^s, X_2 = x_2^s, \ldots, X_N = x_N^s \rangle$ and $t = \langle X_1 = x_1^t, X_2 = x_2^t, \ldots, X_N = x_N^t \rangle$, where $\forall i : x_i^s, x_i^t \in D(X_i)$, the goal is to find the shortest path from s to t in Ω. Assume that the domain $D(X_i)$ of each variable X_i is ordered as follows: $D(X_i) = \langle d_{i_1}, d_{i_2}, \ldots, d_{i_{|D(X_i)|}} \rangle$. In the search space defined by Ω, a state s_1 corresponds to a satisfying assignment of Ω, and a "move" operator from state s_1 to state s_2 changes the value of only one variable, say X_k, from d_{k_l} to $d_{k_{l \pm 1}}$, and is defined to be of unit cost.

It is worth noting that, given a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, the FSPP asks a different question than the CSP. While the task in the CSP is to find a satisfying assignment, the task in the FSPP is to find the shortest path between two given satisfying assignments. $C_1, C_2, \ldots, C_M$ are interpreted as the factors of the implicitly defined global constraint Ω, which in turn is also interpreted as a $2N$-neighbor map. Figure 1 shows two constraints C_1 and C_2, the global constraint Ω, and an instance of the FSPP.

While it is well known that the shortest path problem can be solved in polynomial time, the size of Ω is typically exponential in N. The idea is to develop more efficient algorithms for solving the FSPP that exploit the factored form of Ω. Such algorithms are expected to exploit structure in the *variable-interaction graph*, i.e., a graphical representation of which variables interact with which other variables.

The variable-interaction graph, also known as the constraint network, is an undirected graph that represents each variable as a vertex and the participation of two variables in a common constraint as an edge. A plethora of combinatorial problems that are otherwise NP-hard can be solved using dynamic programming in time that is exponential only in the treewidth of their variable-interaction graphs instead of the total number of variables. Examples include bounded-treewidth CSPs (Dalmau, Kolaitis, and Vardi 2002) and polytrees in probabilistic reasoning (Pearl 1986).

Following the same line of thought, we would like to interest AI and robotics researchers in developing efficient algorithms for the FSPP that exploit the structure of the variable-interaction graph. Such algorithms are expected to enhance heuristic search with elements of dynamic programming. Figure 1 shows that the shortest path in Ω may not be a simple combination of shortest paths in the individual factors. Moreover, Figure 2 shows that there may be constraints such that Ω does not admit a path from s to t, although paths between the projections of s and t exist in the individual factors. In general, therefore, a search algorithm for the FSPP has to exploit the structure of the variable-interaction graph facing the challenges shown in Figures 1 and 2.

Related Work and Comparisons

Factored representations and abstractions have been exploited in planning (Knoblock 1994; Brafman and Domshlak 2006; 2008). They have also been used to design causal-graph-based heuristics in (Helmert 2004) and merge-and-shrink abstractions in (Helmert et al. 2014). (Wehrle, Sievers, and Helmert 2016) provides graph-based factorizations of classical planning problems. (Fabre and Jezequel 2009; Fabre et al. 2010) provide an automata calculus for factored planning, which is extended to Petri nets in (Jezequel, Fabre, and Khomenko 2015). Message passing algorithms have also been used for factored planning in (Jezequel and Fabre 2015). Factored planning using decomposition trees is done using an algorithm called dTreePlan in (Kelareva et al. 2007).

Unfortunately, none of the above methods are applicable to the FSPP, because the logical connective between the factors in the FSPP is an $\wedge$ operation. For example, consider two constraints $C_1(X_1, X_2)$ and $C_2(X_1, X_3)$ that yield $\Omega = C_1 \wedge C_2$. Suppose $(X_1 = d_1, X_2 = d_2)$ and $(X_1 = d_1 + 1, X_2 = d_2)$ are both allowed by C_1. Then a move operator with precondition $(X_1 = d_1) \wedge (X_2 = d_2) \wedge C_1(X_1 = d_1 + 1, X_2 = d_2)$ and effect $(X_1 = d_1 + 1)$ can be created. However, the FSPP is not amenable to reasoning with such move operators because, although transitioning from $(X_1 = d_1, X_2 = d_2)$ to $(X_1 = d_1 + 1, X_2 = d_2)$ is allowed in C_1, it may not be allowed in Ω, because, while $(X_1 = d_1, X_3 = d_3)$ may be allowed by C_2, $(X_1 = d_1 + 1, X_3 = d_3)$ may not be.

In robotics, extending upon the work in (Amir and Engelhardt 2003), (Choi and Amir 2007) presents a factor-guided motion planning algorithm for robotic arms. However, the runtime of this algorithm is exponential in the number of obstacle islands. Factored planning approaches therefore have not been very successful in their applicability to robotics, even though the FSPP potentially applies to robotics domains.

In distributed and multi-agent settings, (Jezequel and Fabre 2012) provides a version of A* for factored planning. (Kvarnström 2011) uses partial-order forward-chaining for loosely coupled agents. When interactions between agents are limited, (Crosby, Rovatsos, and Petrick 2013) provides an automated decomposition method.

In natural language processing, factored versions of A* have been used for many tasks (Klein and Manning 2003; Haghighi, DeNero, and Klein 2007). However, they have been developed to exploit the factored representations of the objective function. They do not address the factorization of the search space itself as required in the FSPP.

Casting planning as constraint satisfaction (Do and Kambhampati 2001) or satisfiability (Kautz 2006) is not directly related to the FSPP although we use the CSP framework for its formalization. (Lozano-Pérez and Kaelbling 2014) uses CSPs to reason about geometric decisions in sequential manipulation planning problems, but we focus on the shortest path problem instead.

Factored-Space Heuristics

In this section, we present a strawman technique for the FSPP. Although this technique does not exploit the factorization of the search space in the search framework, it exploits the factorization for providing heuristic guidance to A*. Given two satisfying assignments s and t in Ω, the primary idea is to compute the shortest paths between their projections in each of the individual constraints (factors). The lengths of these shortest paths are added up to yield a heuristic estimate of the length of the shortest path between s and t in Ω. Although this heuristic estimate is not admissible, a simple modification to this strategy revives the admissibility and consistency required for A* producing shortest paths without re-expansions.

Let us interpret any constraint $C(X_{i_1}, X_{i_2}, \dots, X_{i_k})$ as a k-dimensional map of size $|D(X_{i_1})| \times |D(X_{i_2})| \times \cdots \times |D(X_{i_k})|$. A satisfying assignment s_1 is said to be adjacent to a satisfying assignment s_2 iff they differ in the value of only one variable, say X_{i_m}, and s_1 assigns $d_{i_{m_l}}$ to X_{i_m} and s_2 assigns $d_{i_{m_{l\pm 1}}}$ to X_{i_m}. The cost of moving from s_1 to s_2 in $C(X_{i_1}, X_{i_2}, \dots, X_{i_k})$ is set to $1/\gamma_{i_m}$. Here, γ_j is the number of constraints in which X_j participates. We now prove that this simple weighting of edges in the map interpretation of each individual constraint (factor) leads to cumulative heuristic estimates in Ω that are always admissible and consistent.

Theorem 1. *The factored-space heuristics are consistent.*

Proof. Let $\Omega = C_1 \wedge C_2 \wedge \cdots \wedge C_M$. In Ω, let a satisfying assignment t be the goal; and, for any satisfying assignment s, use $H(s)$ to denote the estimated cost of reaching the goal from s in Ω. $H(s) = 0$ for any goal s. Consider two satisfying assignments s_1 and s_2 that are adjacent. By definition, s_1 and s_2 differ in the value of only one variable, say X_k. Let X_k participate in the constraints $C_{j_1}, C_{j_2}, \dots, C_{j_{\gamma_k}}$. It suffices to prove that the estimated cost of reaching t from s_1 in Ω is no greater than the unit cost incurred in Ω for moving from s_1 to s_2 plus the estimated cost of reaching t from s_2 in Ω, i.e., $H(s_1) \leq 1 + H(s_2)$. In each individual constraint (factor) C_i of Ω, we use $h(s, C_i)$ to denote the length of the shortest path (with weighted edges) between the projections of s and t in C_i. By definition, $H(s_1) = \sum_{i=1}^{M} h(s_1, C_i)$ and $H(s_2) = \sum_{i=1}^{M} h(s_2, C_i)$. Since s_1 and s_2 differ only in the value of X_k, for each individual constraint (factor) C_w of Ω in which X_k does not participate, $h(s_1, C_w) = h(s_2, C_w)$. Now, consider the constraints $C_{j_1}, C_{j_2}, \dots, C_{j_{\gamma_k}}$ in which X_k participates: In each of these constraints, say C_{j_r}, $h(s_1, C_{j_r}) \leq 1/\gamma_k + h(s_2, C_{j_r})$, because $h(s, C_{j_r})$ denotes the length of the shortest path (an exact heuristic) between the projections of s and t in C_{j_r} and the cost of moving from s_1 to s_2 in C_{j_r} is $1/\gamma_k$. Therefore, $\sum_{r=1}^{\gamma_k} h(s_1, C_{j_r}) \leq \gamma_k \cdot 1/\gamma_k + \sum_{r=1}^{\gamma_k} h(s_2, C_{j_r})$. Hence, $\sum_{i=1}^{M} h(s_1, C_i) \leq \gamma_k \cdot 1/\gamma_k + \sum_{i=1}^{M} h(s_2, C_i)$, and $H(s_1) \leq 1 + H(s_2)$. □

Corollary 2. *The factored-space heuristics are admissible.*

Proof. Theorem 1 shows that the factored-space heuristics are consistent. Every consistent heuristic is admissible (Russell and Norvig 2009). □

Preliminary Experimental Results

In this section, we present experimental results for comparing A* with factored-space heuristics against A* with the Manhattan Distance heuristic. The experiments were run on a MacBook Pro with a 2.5GHz quad-core Intel Core i7 processor and 16GB RAM. We demonstrate that, although A* with factored-space heuristics is only a strawman solution to exploiting the factorization of a search space, it outperforms the baseline A* with the Manhattan Distance heuristic. This suggests that exploiting factorization further in the search framework itself could lead to significantly better search algorithms and results.

Table 1: Shows the performance of A* with factored-space heuristics against A* with the Manhattan Distance heuristic for the snake robot. The entries under the corresponding columns represent the numbers of A* node expansions. Test 2 is the result for the FSPP in Figure 3.

ID	Map Size	# Obstacles	**Manhattan Distance**	**Factored-Space**
1	5×10	9	13268	64
2	5×10	14	17978	4221
3	10×10	27	43430	9845

Table 2: Shows the performance of A* with factored-space heuristics against A* with the Manhattan Distance heuristic for random and hand-crafted instances. The entries under the corresponding columns represent the numbers of A* node expansions along with running times. Running times for computing factored-space heuristics (i.e., preprocessing for each high-dimensional map) are listed under "Preprocessing." The domain size of all variables is 10 in each instance, except in Test ID 3, where each variable has a domain size of 8. Test ID 3 is the result for the FSPP in Figure 1.

ID	# Variables	# Constraints	Arity of Each Constraint	**Manhattan Distance**	**Factored-Space**	Preprocessing
1	3	2	2,2	367 (0.011s)	153 (0.004s)	0.001s
2	3	2	2,2	266 (0.008s)	144 (0.004s)	0.001s
3	3	2	2,2	167 (0.006s)	141 (0.004s)	0.001s
4	4	3	2,2,2	2635 (0.112s)	627 (0.029s)	0.002s
5	4	2	3,3	403 (0.018s)	118 (0.007s)	0.024s
6	5	4	2,2,2,2	604 (0.042s)	45 (0.004s)	0.003s
7	5	3	3,3,2	22345 (1.350s)	14675 (1.016s)	0.020s

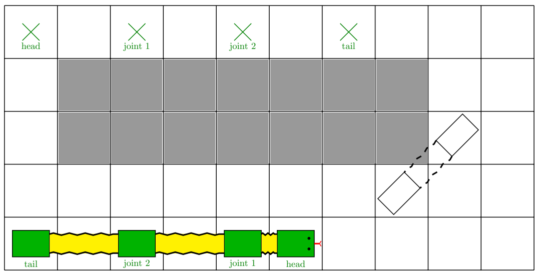

Figure 3: Illustrates the path-planning problem for a geometric agent, i.e., a snake robot, on a 4-neighbor 2D grid map with obstacles. The crosses specify the desired goal positions of the body parts. The figure also illustrates a configuration that is not allowed, because the segment intersects with an obstacle.

We performed experiments for two categories of problems. In the first category, we use a robotics domain. In this domain, we are given a 4-neighbor 2D grid map with obstacles; and we are required to move a geometric agent, instead of a point agent, from a starting configuration to a goal configuration. Our geometric agent of choice is a snake robot illustrated in Figure 3. It has four body parts—a head, a tail, and two joints—and a total of three segments between them. Each body part can move independently within the confines of the map in the horizontal or the vertical direction, but only one body part can move at a time. No segments can shrink or stretch beyond specified limits; the Manhattan Distance between two adjacent body parts can only be 1, 2, or 3. Segments cannot bend, and no segments or body parts can intersect with an obstacle. Body parts cannot occupy the same location at the same time.

The search space for this problem can be decomposed into three binary constraints, one for each pair of adjacent body parts. Table 1 compares the efficiency of A* with factored-space heuristics against A* with the Manhattan Distance heuristic. Here, we observe that the former outperforms the latter by more than an order of magnitude with respect to the number of node expansions.

In the second category, we use both random and hand-crafted instances. For a given number of variables, these instances are created by varying the number of constraints and the arity of each constraint. In generating the constraints, we either (a) first make random choices for allowing or disallowing combinations of values to the participating variables and then tweak the constraints until Ω allows for a path between s and t, where s and t are chosen to be two satisfying assignments such that the value of each variable is set to 0 for s and the largest value in its domain for t, or (b) handcraft adversarial constraints, such as in Test ID 3.

Table 2 compares the efficiency of A* with factored-space heuristics against A* with the Manhattan Distance heuristic. Here, too, we observe that the former outperforms the latter in many cases with respect to the number of node expansions as well as the running times (especially if one can amortize preprocessing). These cases typically have constraints of low arity and are therefore highly factored. Indeed, the performance of A* with factored-space heuristics may increase with increasing degree of factorization. These observations are important because there is no particular exploitable structure to the nature of the constraints themselves, and only the variable-interaction graph is available for exploitation. When the constraints (factors) have relatively high arities, however, such as in Test ID 5, we observe that the preprocessing time for A* with factored-space heuristics is high.

Conclusions and Future Work

In this paper, we formulated the FSSP on a collection of constraints interpreted as factors of a high-dimensional map. Our formalization uses the standard CSP framework but asks a question different from solution extraction. We showed that the FSPP is a generalization of the regular shortest path problem and is particularly relevant to robotics. Furthermore, we showed that the FSPP is not directly amenable to existing approaches in factored planning or A* search for factored objective functions. We developed a strawman technique for the FSPP using factored-space heuristics for A*. We proved that these heuristics are admissible and consistent. We provided experimental results on both random and hand-crafted instances as well as a snake robot domain. In future work, we are interested in exploiting the factorization of a search space in the search framework itself, which could lead to better search algorithms and results.

Acknowledgement

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1724392, 1409987, and 1319966.

References

Amir, E., and Engelhardt, B. 2003. Factored planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 929–935.

Aspvall, B.; Plass, M. F.; and Tarjan, R. E. 1979. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters* 8(3):121–123.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11(1):1–94.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 809–814.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 28–35.

Choi, J., and Amir, E. 2007. Factor-guided motion planning for a robot arm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 27–32.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial intelligence* 42(2-3):393–405.

Crosby, M.; Rovatsos, M.; and Petrick, R. P. A. 2013. Automated agent decomposition for classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 46–54.

Dalmau, V.; Kolaitis, P. G.; and Vardi, M. Y. 2002. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 310–326.

Dechter, R. 2003. *Constraint processing*. Morgan Kaufmann.

Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence* 132(2):151–182.

Fabre, E., and Jezequel, L. 2009. Distributed optimal planning: an approach by weighted automata calculus. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 211–216.

Fabre, E.; Jezequel, L.; Haslum, P.; and Thiébaux, S. 2010. Cost-optimal factored planning: Promises and pitfalls. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 65–72.

Guestrin, C.; Koller, D.; and Parr, R. 2002. Multiagent planning with factored MDPs. In Dietterich, T. G.; Becker, S.; and Ghahramani, Z., eds., *Advances in Neural Information Processing Systems 14*. MIT Press. 1523–1530.

Haghighi, A.; DeNero, J.; and Klein, D. 2007. Approximate factoring for A* search. In *Proceedings of Human Language Technologies: the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 412–419.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–16:63.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 161–170.

Jezequel, L., and Fabre, E. 2012. A#: a distributed version of A* for factored planning. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 7377–7382.

Jezequel, L., and Fabre, E. 2015. Factored cost-optimal planning using message passing algorithms. *Fundamenta Informaticae* 139(4):369–401.

Jezequel, L.; Fabre, E.; and Khomenko, V. 2015. Factored planning: From automata to Petri nets. *ACM Transactions on Embedded Computing Systems* 14(2):26:1–26:25.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* 82(1):35–45.

Kautz, H. 2006. Deconstructing planning as satisfiability. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1524–1526.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

Kelareva, E.; Buffet, O.; Huang, J.; and Thiébaux, S. 2007. Factored planning using decomposition trees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1942–1947.

Klein, D., and Manning, C. D. 2003. Factored A* search for models over sequences and trees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1246–1251.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Kvarnström, J. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 138–145.

LaValle, S. M., and Kuffner, Jr., J. J. 2001. Rapidly-exploring random trees: Progress and prospects. In Donald, B. R.; Lynch, K. M.; and Rus, D., eds., *Algorithmic and Computational Robotics: New Directions*. A K Peters. 293–308.

Li, S. Z. 1994. Markov random field models in computer vision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 361–370.

Lozano-Pérez, T., and Kaelbling, L. P. 2014. A constraint-based method for solving sequential manipulation planning problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3684–3691.

Nesterov, Y., and Nemirovskii, A. 1994. *Interior-point polynomial algorithms in convex programming*. Society for Industrial and Applied Mathematics.

Pearl, J. 1986. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence* 29(3):241–288.

Persson, S. M., and Sharf, I. 2014. Sampling-based A* algorithm for robot path-planning. *The International Journal of Robotics Research* 33(13):1683–1708.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition.

Wehrle, M.; Sievers, S.; and Helmert, M. 2016. Graph-based factorization of classical planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 3286–3292.