Revisiting the Complexity Analysis of Conflict-Based Search: New Computational Techniques and Improved Bounds

Ofir Gordon, Yuval Filmus, Oren Salzman
Technion–Israel Institute of Technology, Computer Science Dept., Haifa, Israel
ofirgo@cs.technion.ac.il, filmus.yuval@gmail.com, osalzman@cs.technion.ac.il

Abstract

The problem of Multi-Agent Path Finding (MAPF) calls for finding a set of conflict-free paths for a fleet of agents operating in a given environment. Arguably, the state-of-the-art approach to computing optimal solutions is Conflict-Based Search (CBS). In this work we revisit the complexity analysis of CBS to provide tighter bounds on the algorithm's run-time in the worst-case. Our analysis paves the way to better pinpoint the parameters that govern (in the worst case) the algorithm's computational complexity. Our analysis is based on two complementary approaches: In the first approach we bound the run-time using the size of a Multi-valued Decision Diagram (MDD)—a layered graph which compactly contains all possible single-agent paths between two given vertices for a specific path length. In the second approach we express the running time by a novel recurrence relation which bounds the algorithm's complexity. We use generating functions-based analysis in order to tightly bound the recurrence. Using these technique we provide several new upper-bounds on CBS's complexity. The results allow us to improve the existing bound on the running time of CBS for many cases. For example, on a set of common benchmarks we improve the upper-bound by a factor of at least 2^{10^7}.

1 Introduction

The *Multi-Agent Path Finding problem* (*MAPF*) is a well-studied problem, which attracts high interest among the robotics and AI community. It can be used to model many real-life applications, from automated warehouses (Wurman, D'Andrea, and Mountz 2008), through computer games (Sturtevant 2012) and to autonomous vehicles (Pallottino et al. 2007). Therefore, many efforts are invested in order to solve the problem as efficiently as possible, under different models and for different scenarios.

In the general version of MAPF (Stern et al. 2019) we are given a graph $G = (V, E)$ with n vertices and a set of k agents $A = \{a_1, a_2, ..., a_k\}$. Each agent a_i is provided with a start and a goal location, (s_i, g_i) s.t. $s_i, g_i \in V$. Time is discretized and at every time-step an agent can either *wait* in its current location or *move* across an edge to an adjacent vertex. A *feasible* solution is a paths set $\mathcal{P} = \{p_1, p_2, ..., p_k\}$ such that p_i is a path for agent a_i from s_i to g_i, and there is no conflict between any two paths in $\mathcal{P}$. We consider two types of conflicts—a *vertex-conflict*, in which two agents occupy the same vertex at the same time-step, and an *edge-conflict*, in which two agents traverse the same edge from opposite sides at the same time-step. An *optimal* solution is a paths set $\mathcal{P}$ which also optimizes some objective function. Arguably, the most common objective functions used for MAPF are:

1. *Makespan*–where we want to minimize the time in which the last agents arrives to its goal.
2. *Sum-of-Costs*–where we want to minimize the combined time it took for all agents to arrive at their goals.

The task of finding an optimal solution is known to be NP-hard (Yu 2016) for both aforementioned objectives. The problem remains NP-hard even when G is a sub-graph of a planar grid graph (Banfi, Basilico, and Amigoni 2017). Nevertheless, state-of-the-art optimal algorithms are able to effectively solve many non-trivial instances. Arguably, the most commonly-used algorithm for solving MAPF optimally is *Conflict-Based Search* (CBS) (Sharon et al. 2015). CBS first plans an initial (possibly infeasible) solution, and then systematically identifies and resolves conflicts. After CBS finds a conflict it applies a constraint that prohibits a conflicted agent from being in the location of the conflict at that certain time-step. Studies that followed, introduce different techniques that allow to empirically improve the algorithm's run-time (Boyarski et al. 2015; Felner et al. 2018; Li et al. 2019a; Zhang et al. 2020).

Despite the ability of those techniques to cope with a wide range of non-trivial instances, there are many cases where CBS and its improvements cannot solve the problem even when allowed very long running times (Kaduri, Boyarski, and Stern 2020). Interestingly, many such empirically-hard instances do not exhibit notable differences from easy ones and identifying the exact source of (theoretical and empirical) hardness is an open question (Salzman and Stern 2020).

The original exposition of CBS (Sharon et al. 2015) presented a (loose) upper-bound on the algorithm's complexity that is exponential both in the problem's parameters (number of agents k and the number of vertices n in the graph G) and in the cost of an optimal solution. In this work we tighten this upper-bound and provide a new point-of-view on the analysis of a worst-case scenario for the algorithm. We believe

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

that this is a first step towards improving our understanding on the problem's hardness which, in turn, will allow to design algorithms that can solve a wider range of instances.

We suggest two novel approaches to improve the algorithm's worst-case complexity analysis. In the first approach we improve the (existing) upper-bound on the number of possible constraints that CBS might need to apply in order to find a solution. We do it by bounding the size of a *Multi-valued Decision Diagram* (MDD)—a layered graph which compactly contains all possible paths between two vertices for a specific path length (Sharon et al. 2013). In the second approach we express the run-time of CBS using a novel recurrence relation which bounds the algorithm's complexity. We compute the *generating function* (Wilf 2006) of the recurrence and use it to tightly bound its value in order to obtain a tighter bound on the complexity of CBS.

Combining the results from both approaches provides us with new tighter upper-bounds for the worst-case complexity of CBS. Beyond the new bounds, we anticipate that the computational tools we introduce will allow to obtain future improvements to the upper-bound. For example, this can be obtained via tighter bounds on the recurrence relation, or by improving the analysis of an MDD size.

2 Setting and Background

Given an optimal solution $\mathcal{P}$, denote by $\mathcal{T}(p)$ the time that a single-agent's path $p \in \mathcal{P}$ terminates (note that wait moves are counted as a timestep in a path p). Now, set C to be the latest time that a single-agent's path $p \in \mathcal{P}$ terminates. Namely, $C = \max_i \{\mathcal{T}(p_i)\}$.

Under the minimal-makespan objective, C constitutes the cost of the optimal solution $\mathcal{P}$. Thus, for the rest of this paper we will consider C as the cost of an optimal solution of the problem, to be used in the complexity analysis. Notice that kC constitutes an upper-bound on the cost of an optimal solution for the sum-of-costs objective. We further discuss the applicability of our results for the sum-of-costs objective in Sec. 5.

2.1 CBS and its Complexity Analysis

Conflict-Based Search (CBS) (Sharon et al. 2015) is a two-level search algorithm which works as follows—first it finds an optimal path for each agent independently using some single-agent search algorithm like A* (Hart, Nilsson, and Raphael 1968). CBS then works to resolve conflicts that occur in the solution: in the high-level search it preforms a best-first search upon a constructed conflicts-tree (CT). Each node in the CT consists of a *solution*, the solution's *cost* and a set of *constraints* imposed on the agents. A constraint is either a *vertex-constraint* of the form $\overline{\langle a, v, t \rangle}$, which prohibits agent a from being at vertex v at time-step t, or an *edge-constraint* of the form $\overline{\langle a, u, v, t \rangle}$, which prohibits agent a from crossing the edge (u, v) at time-step t. We refer to such constraints as *negative constraints*.

In each iteration, CBS selects an unexpanded CT node with a lowest cost. It then finds a conflict that occurs between two agents in the node's solution. It splits the CT node into two child-nodes, each with a constraint on one of the agents that were involved in the conflict. It then runs the low-level search to construct a new solution in each child node, that does not violate the new constraint, by running a single-agent search algorithm like A*.

The basic version that we just presented was recently improved using *positive constraints* (Li et al. 2019b). In a positive constraint $\langle a, v, t \rangle$, agent a is required to be at vertex v at time-step t. When using positive constraint with CBS, a CT node is split into two child nodes using a positive and negative constraints forcing and forbidding the conflicted agent to be at a vertex or edge at a certain time-step, respectively.

An analysis of the worst-case time-complexity of CBS was originally presented by Sharon et al. (2015). They show that CBS's complexity can be decomposed to bounding the size of the CT and the complexity of the single-agent search in each low-level iteration. We refer to the size of the CT in a worst-case scenario as the *high-level search complexity*. The low-level search complexity corresponds to running A* for a single agent. For the rest of this paper we focus on analyzing the high-level search complexity, thus, a complete upper-bound on CBS's complexity can obtained by simply multiplying any of the following results with the complexity of a single agent's A*-search.

The original analysis uses the assumption that each agent can potentially be in every vertex at every time-step. This bounds the number of (negative) constraints that CBS might need to apply by $\mathcal{O}(nkC)$. At each CT node exactly one constraint is added. Thus, the number of possible constraints bounds the depth of the CT, and gives an overall bound on CBS's running time of $\mathcal{O}(nC \cdot 2^{knC})$.[1] In the rest of this paper, we refer to this analysis and bound as the *original analysis* and *original upper-bound*, respectively.

It is important to note that the original analysis does not account for the possibility that CBS would apply edge-constraints (in order to resolve conflicts). It is possible that in a worst-case scenario the algorithm would require not only to prevent any agent from occupying any vertex in the graph at each time-step, but also from crossing each edge of the graph, in order to find the optimal solution. Therefore, accounting for edge constraint should further increase the theoretical upper-bound. For clarity of exposition, we present our tools for analysing CBS's complexity with the same assumption, i.e., that only vertex-constraints are considered. Nevertheless, we address this issue in Sec. 5 and show how to incorporate edge-constraints in the complexity analysis.

2.2 Multi-valued Decision Diagram (MDD)

The *multi-valued decision diagram* MDD_i^C is a layered graph that consists of C layers, which compactly contains all possible paths of agent a_i of cost at most C from s_i to g_i (Sharon et al. 2013). A vertex $v \in V$ appears at the t'th layer of MDD_i^C if it is reachable from s_i and g_i in t and $C - t$

[1]The original paper contains a minor error in the calculation of the upper bound. The bound presented here is the new bound whose validity was verified with one of the authors. Similarly, the oversight regarding not accounting for edge constraints (explained shortly) was also discussed and verified with one of the authors in the original CBS paper.

steps, respectively. Finally, the *size* $\mathcal{M}$ of an MDD, represents the total number of MDD nodes and the size $\mathcal{M}_t$ of the t'th layer is the number of MDD nodes in that layer.

MDD graphs are commonly-used for different purposes in MAPF algorithms, since they can be constructed efficiently for a given cost and their compact representation contains information that can help improve the identification and classification of conflicts (Li et al. 2019a; Zhang et al. 2020). In this work we use MDDs to bound the number of possible constraints that might need to be applied on a single agent during a CBS execution.

2.3 Generating Functions for Bounding Recurrence Relations

Generating functions are a well-known mathematical tool which, among other things, can be used to bound recurrence relations. Formally, a generating function of a sequence $a_0, a_1, a_2, \ldots$ with the general element denoted by a_r, is the function $F(x) = \sum_{r\geq 0} a_r x^r$, i.e., the sequence elements are the coefficients of the series expansion of $F(x)$. This notion can be extended for a sequence (or recurrence relation) with multiple variables. For instance, given a recursion $T(r,s)$ which defines a sequence, a possible generating function for it will be of the form $F(x,y) = \sum_{r,s\geq 0} T(r,s)\, x^r y^s$. For further details on generating functions see, e.g., the book by Wilf (2006).

Given a generating function for a specific sequence, there can be many methods which allow to utilize the function in order to bound the value of the sequence at a certain index. These different methods are dependent on the sequence and the obtained function and there is no guarantee that a certain method could always be applied for this purpose.

Pemantle and Wilson (2008) provide one approach for dealing with recursions of multiple variables which we briefly describe (additional details are presented mainly in Sec. 3 of the aforementioned paper) as it will be a key technique used to obtain our complexity bounds. Assume that we are given a recursion $T(r,s)$ and a matching generating function for it $F(x,y) = \sum_{r,s\geq 0} T(r,s)\, x^r y^s$ that can be expressed by the following form: $F(x,y) = \frac{G(x,y)}{H(x,y)}$. Denote by H_z the partial derivative of H for z (where z can be a sequence of x and y). The first step calls for finding *critical points*, which are given by the solutions in the positive quadrant (i.e., $x, y \geq 0$) for the following system:

$$\begin{cases} H = 0 \\ sxH_x = ryH_y. \end{cases} \tag{1}$$

Denote the critical points by $q_1, q_2, \ldots, q_m$. Each point $q_i = (x_i, y_i)$ *contributes* a certain factor to the approximation of $T(r,s)$, and this contribution can be calculated according to the point's multiplicity. The exact way each point contributes to the bound is detailed by Pemantle and Wilson (2008) and in the extended version of this paper (Gordon, Filmus, and Salzman 2021).

Assume that the contribution of q_i is given by $T_i(r,s)$ for each $1 \leq i \leq m$, then the analysis suggests that the asymptotic growth of $T(r,s)$ can be tightly approximated by one of the factors which is given by the critical point's contribution.

3 CBS's Complexity Analysis using MDDs

Recall that the original analysis was obtained by bounding the number of possible (vertex) constraints that CBS may apply. In addition, as explained in Sec. 2.1, the original analysis did not account for edge constraints as a possible mean that can be used by the algorithm. We temporarily limit our analysis to account for vertex constraints only and defer handling edge constraints to Sec. 5.

We suggest a new approach to bound this number of possible constraints that CBS may apply, using the following observation:

Observation 1. *Given an agent a_i and an optimal solution's cost C, the maximal number of negative constraints that CBS may apply on a_i is bounded by the size of MDD_i^C.*

Obs. 1 holds as CBS may only apply a constraint on agent a_i at vertex v for time-step t if a_i can reach v from s_i within t time-steps and still reach g_i in $C - t$ time-steps, which is the exact definition of an MDD node.

From Obs. 1 we obtain the following corollary:

Corollary 1. *Let $\mathcal{M}$ denote the maximal size of an agent's MDD in a given instance. The size of CBS's conflict-tree is bounded by $\mathcal{O}(2^{k\mathcal{M}})$ for any execution of the algorithm on this instance. This implies a similar bound on the algorithm's high-level search complexity.*

Cor. 1 can be used to recover the original analysis of CBS—a (loose) bound on $\mathcal{M}$ can be obtained by bounding the size of any MDD layer by n. Thus $\mathcal{M} = \mathcal{O}(nC)$ which gives us the original bound of $\mathcal{O}(2^{knC})$.

We present two (tighter) bounds on $\mathcal{M}$. The first (Sec 3.1) removes the number of environment vertices from the complexity analysis, eliminating the possibility to deem a problem computationally hard just by adding inconsequential vertices to the environment.

The second bound accounts for the structure of G. In addition to the obtained bound, it demonstrates a complexity analysis restricted to a specific setting. This allows to (potentially) obtain tighter bounds on the size of an MDD which, in turn, provides tighter bounds on CBS's complexity for a given setting of interest.

In the following sections, we assume that G is a full $\sqrt{n} \times \sqrt{n}$ grid with no blocked vertices and $s_i = g_i$ for some agent a_i (this serves as an upper bound on the size of MDD_i^C for any other instance).

3.1 Upper-Bound on the Size of an MDD

For the simplicity of the exposition, we restrict the discussion in this section to MAPF instances on infinite 4-connected grids (Banfi, Basilico, and Amigoni 2017; Stern et al. 2019). That is, we consider the setting where an agent can move in four directions from any vertex in the graph. Nonetheless, we emphasize that the technique we use to bound CBS's complexity can be used to bound the size of an MDD for any environment.

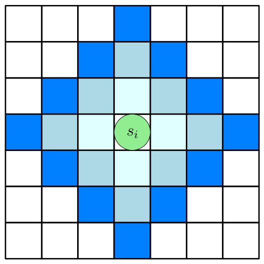


Figure 1: Illustration of vertices reachable for an agent a_i located at s_i within 1-3 time-steps, on a 4-connected grid.

For any optimal path, an agent a_i can't be located at any vertex within distance larger than $C/2$ from its start s_i or goal g_i. This implies a symmetry on the structure of MDD_i^C—the last $\lfloor C/2 \rfloor$ layers form a mirror-image of the first $\lfloor C/2 \rfloor$ layers. The number of vertices on a grid which are reachable from s_i within exactly t steps is $4t$ (see Fig. 1). At time-step t, a_i can be located at any vertex within distance at most t from s_i. Therefore, we sum the number of reachable vertices in the range from one to t (excluding s_i). For any $t \leq C/2$ the size of the t'th layer in MDD_i^C is:

$$\mathcal{M}_t \leq \sum_{i=1}^{t} 4i = 2t(t+1). \tag{2}$$

Given the aforementioned symmetry:

$$\mathcal{M} \leq 2 \cdot \sum_{t=1}^{C/2} 2t(t+1) = \frac{C^3 + 6C^2 + 8C}{6} = \mathcal{O}(C^3). \tag{3}$$

We assume for simplicity that C is even, if C is odd then the size of the middle layer ($\mathcal{O}(C^2)$ according to Eq. 2) needs to be added to the result of Eq. 3.

By placing the result from Eq. 3 in Cor. 1 we obtain that:

Claim 1. *The high-level search complexity of* CBS *on grid graphs is bounded by* $\mathcal{O}\left(2^{kC^3}\right)$.[2]

The bound in Claim. 1 provides a tighter estimation for the algorithm's complexity for any instance where: $C^3 < n \cdot C \Rightarrow C < \sqrt{n}$. In addition, this result removes n from the bound expression.

3.2 MDD Size Based on the Graph's Radius

Definition 1. *The* ***distance*** $dist(u, v)$ *between two vertices* u, v *in a graph is the number of edges on a shortest path between them. The* ***radius*** *of a graph is* $\rho = \min_{u \in V} \max_{v \in V} \{dist(u, v)\}$. *A vertex* u *for which it holds that* $\forall v \in V : dist(u, v) \leq \rho$ *is called a* ***center*** *vertex.*

In a complete square grid of size n, we have that $\rho = \sqrt{n} - 1$, with the center in the $\lceil \frac{\sqrt{n}}{2} \rceil$'th row and column for an odd value of $\sqrt{n}$. When $\sqrt{n}$ is even, there is no single center vertex. For simplicity, we assume that $\sqrt{n}$ is odd.

Observation 2. *A layer of size* n *exists in* MDD_i^C *only if* $C \geq 2\rho$ *(note that a layer's size can't exceed* n*).*

Obs. 2 allows us to characterize settings for which it is possible to refine our previous analysis. For the first and last ρ layers of MDD_i^C we bound a layer's size using Eq. 2. Using Obs. 2, the remaining layers are the only layers with size n. This gives us the following bound for $\mathcal{M}$, for cases where $C = 2\rho + \delta$ for some $\delta \in \mathbb{N}$:

$$\begin{aligned} \mathcal{M} &\leq \delta n + 2 \cdot \sum_{t=1}^{\rho} 2t(t+1) \\ &= \frac{4}{3} \cdot \rho(\rho+1)(\rho+2) + \delta n \end{aligned} \tag{4}$$

The expression $\frac{4}{3} \cdot \rho(\rho+1)(\rho+2)$ is smaller than $2 \cdot \rho^3$ for any $\rho \geq 7$. By placing the result from Eq. 4 in Cor. 1 we obtain that:

Claim 2. *The high-level search complexity of* CBS *on grid graphs with radius* $\rho \geq 7$ *where* $C = 2\rho + \delta$ *for some* $\delta \in \mathbb{N}$ *is bounded by* $\mathcal{O}\left(2^{k \cdot (2\rho^3 + \delta n)}\right)$.

Claim. 2 not only allows to express the bound in terms of a new (and arguably, more relevant) parameter (namely, the radius of a graph), it also provides a slightly tighter bound on the overall complexity for full grid graphs, as long as the graph's radius is not too small. The new bound is tighter than the original bound for cases where: $k \cdot (2\rho^3 + \delta n) < kn \cdot (2\rho + \delta) \implies \rho < \sqrt{n}$ (which, indeed holds for complete grids). The hidden constant in the Big-O notation in both the new and the original bounds is small and does not affect the asymptotic comparison between them.

4 Complexity Analysis for CBS using a Recurrence Relation

We introduce a novel recurrence relation which bounds the high-level search complexity. More precisely, it bounds the maximal number of CT nodes that might be generated during the high-level search. We provide an upper-bound on this recurrence relation that allows to improve the original bound on the run-time of CBS for many cases.

Our improved bound incorporates the fact that recent CBS variants use *positive constraints* (Sec. 2.1). This is in contrast to the original analysis that only considers negative constraints. However, the method in which the recursion is defined is not tied to positive constraints. We believe that tighter bounds may be obtained in the future by defining a similar recursion for alternative implementations of CBS (such as CBS with symmetry-breaking (Li et al. 2019c)).

4.1 Recurrence Relation which Bounds CBS's Worst-Case Complexity

Given a MAPF instance with k agents on a graph of size n where the optimal cost of a solution is C, our goal is to

[2]Note that in general it does not hold that $2^{\mathcal{O}(m)} = \mathcal{O}(2^m)$. The reason for which it does hold in this case is since the hidden constant in the Big-O notation in Eq. 3 is smaller than 1.

bound the maximal number of CT nodes that might be generated by CBS after applying a given number of positive and negative constraints. CBS will terminate if:

1. All possible negative constraints were applied (this assumption is similar to the one used for the original bound).
2. The algorithm applied C positive constraints on each of the k agents.

Note that any (positive or negative) constraint applied to a CT node cannot be applied to any of its children in the CT. In addition, if agents a_i and a_j were found to be in a conflict at vertex v at time-step t, applying a positive constraint on agent a_i implies that the negative constraints $\overline{\langle a_i, v, t\rangle}$ and $\overline{\langle a_j, v, t\rangle}$ cannot appear in the sub-tree of the CT node.

From the above we get the following recurrence relation:

Lemma 1. *Let r and s denote the maximal number of negative and positive constraints that CBS may apply before it is bound to terminate, respectively. Then, the high-level complexity of CBS is bounded by:*

$$T(r,s) \leq \begin{cases} 1, & r = 0 \text{ or } s = 0 \\ 3, & r = 1 \text{ and } s > 0 \\ T(r-1,s) + & \\ \quad T(r-2,s-1)+1, & \text{else.} \end{cases} \tag{5}$$

For $r = 0$ or $s = 0$ we get that one of the aforementioned conditions for termination holds, therefore, this respected node is a leaf node. For $r = 1$ there is still a single negative constraint left to apply, so the node can be split only one more time, creating two additional leaves (and the node itself is also counted). For any other inner-node, the algorithm would split it according to a conflict by applying a negative constraint on one branch, and a positive constraint for the other branch. Note that when applying a negative constraint (i.e., reducing r by one) it does not imply that a positive constraints has been applied (therefore, in the first component of the recurrence step s does not change).

We present two techniques for upper-bounding the recursion presented in Eq. 5, which in turns provide an upper-bound for CBS's complexity.

4.2 Induction-Based Bound

Claim 3. *For any $(r,\ s)$ s.t. $r \geq 1$ and $s \geq 1$ it holds that:*

$$T(r,s) \leq 3 \cdot r^s. \tag{6}$$

Which implies that $T(r,s) = \mathcal{O}(r^s)$.

Proof sketch. The proof is by induction over pairs (r,s), assuming an order where $(r_1,s_1) \geq (r_2,s_2)$ if $r_1 \geq r_2$ and $s_1 \geq s_2$ (there exists such an order on pairs where $r,s \in \mathbb{N}$).

Base:

$$T(1,s) \leq 3 \leq 3 \cdot 1^s.$$
$$T(r,1) \leq T(r-1,1) + T(r-2,0) + 1 = T(r-1,1) + 2$$
$$\leq \cdots \leq T(1,1) + 2(r-1) = 2r + 1 \leq 3 \cdot r^1.$$

Step: We assume that the claim holds for all pairs smaller than (r,s) and prove for (r,s):

$$\begin{aligned} T(r,s) &\leq T(r-1,s) + T(r-2,s-1) + 1 \\ &\underset{\text{i.h.}}{\leq} 3(r-1)^s + 3(r-2)^{s-1} + 1 \\ &\underset{*}{\leq} 3(r-1)^s + 3(r-1)^{s-1} \\ &= 3[(r-1)^{s-1} \cdot (r-1) + (r-1)^{s-1}] \\ &= 3(r-1)^{s-1} \cdot (r-1+1) \\ &= 3r \cdot (r-1)^{s-1} \leq 3r \cdot r^{s-1} = 3r^s. \end{aligned}$$

Where $*$ holds because $(r-1)^{s-1} \geq 1$ for $r > 1$ and $s > 1$. □

Recall that negative and positive constraints are bounded by $r = k\mathcal{M}$ and $s = kC$, respectively (where $\mathcal{M}$ is the size of an MDD graph of a single agent). Placing those values in Eq. 6 gives the following result:

$$T(k\mathcal{M}, kC) \leq \mathcal{O}\big((k\mathcal{M})^{kC}\big). \tag{7}$$

From Eq. 7 we obtain the following lemma:

Lemma 2. *The time-complexity of the high-level search of CBS is bounded by $\mathcal{O}\big((k\mathcal{M})^{kC}\big)$.*

By taking $\mathcal{M} = nC$ (i.e., the bound on an MDD size considered in the original analysis), we get an upper-bound of $\mathcal{O}\big((knC)^{kC}\big)$ in contrast to the original bound of $\mathcal{O}(2^{nkC})$. Here again, the hidden constant in the Big-O notation in both bounds is small, thus, we omit it in the upcoming comparison between them (see end of Sec. 4.3).

4.3 Generating Functions-Based Bound

In our second approach, we present an alternative approach to bounding the recursion (Eq. 5) using *generating functions*. This, in turn, will allow us to obtain a tighter bound on CBS's complexity. Due to lack of space we only outline the analysis and refer the reader to the extended version of this paper (Gordon, Filmus, and Salzman 2021).

We start by introducing the generating function for $T(r,s)$. We then continue to follow the steps outlined by Pemantle and Wilson (2008) to obtain a bound on $T(r,s)$. We stress that the suggested analysis method is not applicable for formally proving the bound's correctness, but we can use it to deduce an asymptotic upper-bound which we then support empirically for a variety of different values.

The method by Pemantle and Wilson (2008), as described in Sec. 2.3, calls for finding the contribution for the bound obtained by each critical point given from the solutions for Eq. 1. We first find the the contribution for each critical point. We then apply the following key observation which allows us to deduce a suggested upper-bound for CBS. The observation follows from our analysis of the size of an MDD (Sec. 3.1):

Observation 3. *For any MAPF instance, there is a linear dependency between r, the maximal number of negative constraints and s, the maximal number of positive constraints that CBS can apply. Specifically, $r = n \cdot (kC) = n \cdot s$.*

Benchmark Category	n	k	C	ORG	REC+IND	REC+GF	$\frac{\text{ORG}}{\text{REC+GF}}$
Warehouse	9,776	8	120	2^{10^7}	2^{10^5}	2^{10^5}	$\mathbf{2^{10^7}}$
Warehouse	9,776	64	140	2^{10^8}	2^{10^6}	2^{10^5}	$\mathbf{2^{10^8}}$
Warehouse	38,756	256	250	$2^{10^{10}}$	2^{10^7}	2^{10^5}	$\mathbf{2^{10^{10}}}$
Room	206,642	8	400	2^{10^9}	2^{10^6}	2^{10^6}	$\mathbf{2^{10^9}}$
Room	206,642	8	500	2^{10^9}	2^{10^6}	2^{10^6}	$\mathbf{2^{10^9}}$
Empty	2,304	64	70	2^{10^8}	2^{10^6}	2^{10^4}	$\mathbf{2^{10^8}}$
Empty	2,304	128	80	2^{10^8}	2^{10^7}	2^{10^4}	$\mathbf{2^{10^8}}$
Random	3,687	64	100	2^{10^8}	2^{10^6}	2^{10^4}	$\mathbf{2^{10^8}}$
Random	3,687	128	100	2^{10^8}	2^{10^7}	2^{10^4}	$\mathbf{2^{10^8}}$

Table 1: A comparison between the different upper-bounds obtained using the original analysis (ORG), Lemma. 2 (REC+IND) and Prop. 1 (REC+GF), on standard benchmarks (Sturtevant 2012; Stern et al. 2019). The last column presents a lower bound on the ratio between our improved bound and the original bound, which reflects the improvement. All bounds are calculated considering that $\mathcal{M} = nC$. Note that all actual bounds include a small constant multiplication factor, but the comparison in this table accounts only for the asymptotic factors.

By applying Obs. 3 we get that one of the contribution factors obtained from the analysis gives a tight upper-bound on $T(r,s)$.

As explained, We start with presenting the generating function for this recursion, which is:

$$F(x,y) = \frac{1-x+2xy-x^2y}{(1-x)(1-y)(1-x-x^2y)}. \tag{8}$$

We denote $F = G/H$ where

$$G(x,y) = 1-x+2xy-x^2y,$$
$$H(x,y) = (1-x)(1-y)(1-x-x^2y),$$

and solve Eq. 1 to find the critical points. In this setting there are three such points:

$$q_1 := (x_1,y_1) = \left(\frac{-1+\sqrt{5}}{2}, 1\right),$$
$$q_2 := (x_2,y_2) = (1,1),$$
$$q_3 := (x_3,y_3) = \left(\frac{r-2s}{r-s}, \frac{s(r-s)}{(r-2s)^2}\right).$$

We denote the matching contribution factor of each point q_i by $T_i(r,s)$. Computing the exact contribution for each point is done according to Pemantle and Wilson (2008). This involves basic (yet daunting) algebraic manipulations and is summarized in Lemma. 3 (proof omitted).

Lemma 3.

$$T_1(r,s) = 1,$$
$$T_2(r,s) = \mathcal{O}(1)\cdot\left(\frac{1+\sqrt{5}}{2}\right)^r,$$
$$T_3(r,s) = \frac{(r-s)^{r-s}}{(r-2s)^{r-2s}\cdot s^s}\cdot\frac{2s}{r-2s}\cdot\sqrt{\frac{\alpha}{2\pi}},$$

where $\alpha = \mathcal{O}\left(\frac{r^2}{s}\right)$.

Following Pemantle and Wilson (2008), we can use Lemma. 3 to estimate the asymptotic growth of Eq. 5. Specifically, this growth is likely to be estimated by one of the three terms. Yet, using it to deduce an upper-bound for CBS's complexity is not straightforward.

Fortunately, by applying Obs. 3 we can obtain an estimated bound on Eq. 5 that is tighter than the one obtained using the induction-based analysis (Lemma. 2). We do it by restricting the recurrence to values of r and s that can be attained in our MAPF setting. Specifically, using $r = n\cdot s$ in Lemma. 3 we have that,

Proposition 1. *The high-level search complexity of CBS for instances with $n \geq 4$ vertices, k agents and an optimal solution cost C is bounded by $\mathcal{O}\big((en)^{kC}\big)$.*

We approximate the value of $T(ns,s)$ according to Lemma. 3 and have that

$$\begin{aligned} T_1(ns,s) &= 1,\\ T_2(ns,s) &= \mathcal{O}(1)\cdot\left(\frac{1+\sqrt{5}}{2}\right)^{ns},\\ T_3(ns,s) &= \left(\frac{(n-1)^{n-1}}{(n-2)^{n-2}}\right)^s\cdot\frac{2}{n-2}\cdot\sqrt{\frac{\beta}{2\pi s}}, \end{aligned} \tag{9}$$

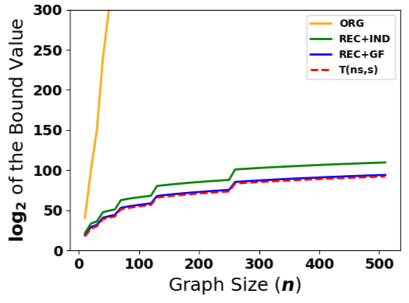
where $\beta = \mathcal{O}(n^2)$.

The contribution to $T(ns,s)$ from q_1 and q_2 (given by $T_1(ns,s)+T_2(ns,s)$) is identical to the contribution from q_3 (given by $T_3(ns,s)$) at $n_0 = \frac{\sqrt{5}+2}{2} \approx 3.618033$. In Sec. 4.4 we empirically demonstrate that $T_3(ns,s)$ indeed constitutes a tight bound for any $n > n_0$.

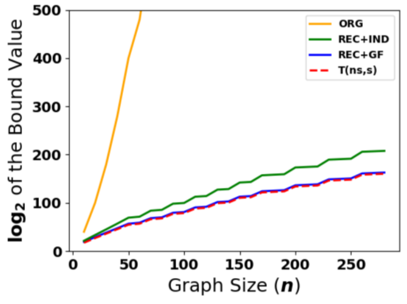
Therefore, we continue with the simplification of the expression given by $T_3(ns,s)$. Since $\frac{2}{n-2}\cdot\sqrt{\frac{\beta}{2\pi}} = \mathcal{O}(1)$ and $\left(\frac{(n-1)^{n-1}}{(n-2)^{n-2}}\right) < en$, we get the following result:

$$T(ns,s) = \mathcal{O}((en)^s),$$

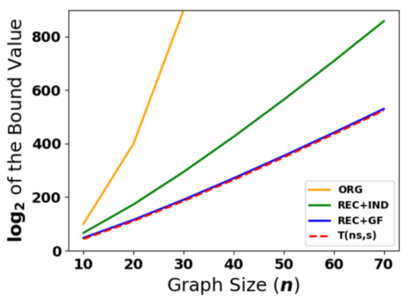
with a small hidden constant factor in the Big-O notation. By placing $s = kC$, which is the maximal number of positive



(a) $s = \log_2 n$



(b) $s = \sqrt{n}$



(c) $s = n$

Figure 2: The $\log_2$ of the bounds as a function of the graph size for different ratios between the graph's size (n) and the instance properties ($s = kC$). The two new bounds (REC+IND, REC+GF) are significantly lower than the original bound. Notice that the approximation obtained from the generating functions analysis (REC+GF) indeed tightly bounds the recurrence $T(ns, s)$.

constraints that needs to be applied by CBS in the worst case, we get the desired bound.

Prop. 1 improves the original known bound of $\mathcal{O}\big(2^{nkC}\big)$ for any set of values n, k and C. Moreover, it is also tighter than the already-improved bound presented in Lemma. 2. Notice that it allows to replace the asymptotic factor of kC in the base of the exponent with a constant (e), while also still eliminating the exponential dependency in n.

New bounds can also be obtained by combining the results from Sec. 3 that bound the size of an MDD. For example, using Eq. 3 we observe that there is a quadratic dependency between $r = kC^3$ and $s = kC$ on 4-connected grids. By simply substituting n with C^2 in the bound obtained by Prop. 1 we have that,

Corollary 2. *The high-level search complexity of CBS on 4-connected grids is bounded by* $\mathcal{O}\big((eC)^{2kC}\big)$.

4.4 Empirical Comparison of Bounds

To demonstrate the difference between the new bounds and the original one, we evaluated their ratio for commonly-used benchmarks. The results are summarized in Table 1, which presents the worst-case complexity of CBS's high-level search, for the original bound (ORG), the looser induction-based bound presented in Lemma. 2 (REG+IND) and the tighter generating functions-based bound presented in Prop. 1 (REC+GF). It also presents the ratio between the original bound and the best newly-obtained bound, to demonstrate the improvement.

For all benchmarks, the new bound improves the original one by a factor of at least 2^{10^7}. More precisely, for any instance we examined, we obtain a significantly-tighter bound on the high-level complexity of CBS.

To support the correctness of the bound obtained in Prop. 1, we evaluated and compared its value to the recurrence's value for large values of n and s. We also use this evaluation to further demonstrate the difference between the new bounds and the original one.

Fig. 2 presents a comparison (on a logarithmic scale) of the three bounds (ORG, REG+IND and REC+GF) and $T(r, s)$ for different graph sizes n for the setting $r = ns$ (see Obs. 3). The three different figures reflect different settings of MAPF, where the dependency between the graph's size and the instance properties (number of agents and optimal solution cost) changes. Indeed for all different cases the same trend can be observed where the gap between the presented functions demonstrate the magnitude of the improvement obtained using our analysis. In addition, the figures serve as an empirical validation of the generating functions-based analysis—the asymptotic bound (REC+GF) tightly approximates the recurrence relation (Eq. 5).

5 Summary and Discussion

5.1 Summary

We presented two novel approaches for analyzing the worst-case complexity of CBS's high-level search. In the first approach, by analysing the size of an agent's MDD graph, we provided two new upper-bounds for CBS's high-level search of $\mathcal{O}(2^{kC^3})$ and $\mathcal{O}(2^{k\cdot(2\rho^3+\delta n)})$. The latter is a bound for a setting where a solution's optimal cost is dependent on the radius ρ of G. Our approach allows to seamlessly obtain tighter bounds on CBS's complexity given tighter bounds on $\mathcal{M}$. Such bounds may be obtained either by (i) better analyzing the general structure of an MDD or (ii) restricting the analysis for a specific instance of interest.

In the second approach we presented the recurrence relation $T(r, s)$. An upper-bound on $T(r, s)$ constitutes a bound on the size of the CT of CBS, therefore bounding the complexity of CBS's high-level search. Using this approach we obtain a new general bound of $\mathcal{O}((k\mathcal{M})^{kC})$. When using $\mathcal{M} = nC$, we obtain the new induction-based bound of $\mathcal{O}((knC)^{kC})$.

Using a generating functions-based bound, we obtain a tighter bound on the recurrence which, in turn, provides a tighter bound for CBS. Observing that there exists a linear dependency between the number of negative and positive constraints, allows us to achieve further improvement, and eventually obtain a bound of $\mathcal{O}\big((en)^{kC}\big)$, which improves the original bound on the algorithm by a significant factor for a wide range of standard benchmarks.

We believe that the recurrence relation can be further improved, in order to better express the real conditions of a worst-case scenario. An immediate step would be to try and account for tighter dependencies between the number of constraints that are being eliminated once a single constraint is applied. In addition, revisiting the conditions on the recursion's parameters may allow to tighten the upper-bound on the recursion, and in turn, on the complexity of CBS.

It is important to note that the new bounds are still somewhat loose and present a worst-case analysis. However, our analysis paves the way to better pinpoint the parameters that govern (in the worst case) the algorithm's computational complexity as well as analyze the complexity when restricted to certain settings. Moreover it provides a general methodology that can be used to analyze different variants of the MAPF problem. For example, in the next sections we show how to seamlessly account for edge constraints (Sec. 5.2) as well as for settings that optimize the Sum-of-Costs objective (Sec. 5.3).

5.2 CBS with Edge-Constraints

Recall that the analysis we performed in Sec. 3 and 4 (as well as the original analysis) accounted for vertex constraints only. We now show simple approaches to account for edge constraints using the existing analysis. Accounting for edge constraints directly is left for future work.

Counting edge constraints as vertex constraints Notice that an edge constraint implicitly defines two vertex constraints (forcing agent a_i to traverse (u, v) at time t corresponds to the constraint that a_i has to be in vertex u and v at times t and $t + 1$, respectively). Thus, we can simply increase the number of vertex constraints by the number of possible edge constraints (which is twice the number of edges as each edge can be traversed in both directions). Specifically, on 2D-grids, each node has at most 4 outgoing and incoming edges, therefore, using the original analysis the number of negative constraints would increase to $nkC + 8nkC = 9nkC$ in the worst-case. For the general case where $|E| \leq n^2$, the number of negative constraints would increase to $(2n^2 + n) \cdot kC$ in the worst-case. We emphasize that while the increase may seem negligible, the actual worst-case complexity is exponential in the number of negative constraints and the additional constraints would appear in the exponent of the original bound. For example, the original bound of $\mathcal{O}(2^{nkC})$ would be $\mathcal{O}(2^{9 \cdot nkC})$.

A similar approach can be applied to the recursion-based bounds presented in Sec. 4. We can consider a total number of negative constraints of $r = 9nkC$, which also includes the negative edge-constraint. One can show that Eq. 5 still upper-bounds the number of expanded nodes in CBS's high-level search. Thus, by placing $r = 9nkC$ in Prop. 1 we obtain a bound of $\mathcal{O}\Big((9en)^{kC}\Big)$.

It is important to note that accounting for edge constraints increases the relative improvement of the bounds we present over the original bound. This is because in our bound the additional work is reflected by increasing the base of the exponent and not the exponent itself as in the original bound.

Counting MDD edges In Cor. 1 the size of an MDD is defined as the number of MDD vertices. However, if we wish to account for edge constraints, the same result holds simply by changing the definition of the size of an MDD to be the number of MDD vertices *and* edges. For instance, on a 4-connected grid, the maximal (outgoing) degree of each MDD node is five. Therefore, the total number of MDD edges is bounded by five times the number of MDD vertices. Using Eq. 3, the total number of vertex and edge constraints is bounded by $(1 + 5) \cdot \mathcal{O}\left(C^3\right)$. We then update Claim. 1 to incorporate edge constraints, and get a bound of $2^{\mathcal{O}\left(kC^3\right)}$ (with a small hidden constant factor slightly larger than 1).

5.3 Analysis for the Sum-of-Costs (SoC) Objective

The bounds we provided (Sec. 3 and 4) were obtained and expressed using C—an upper-bound on a single agent's path length in an optimal solution. In the setting where we seek to minimize the makespan, C is indeed an optimal solution's cost. Unfortunately, this is not the case for the SoC objective.

One way to use our results when using SoC as our optimization objective is to observe that kC is an upper-bound for the optimal solution's cost. Namely, denote $C' = kC$ and use C' instead of kC in all the bounds presented. For example, our generating function-based analysis for the SoC objectives yields the bound of $\mathcal{O}\Big((en)^{C'}\Big)$. This is a slightly looser bound expressed with an upper-bound on a single-agent's solution's optimal cost.

5.4 Future Improvements for CBS

Throughout this paper we presented several observations, which improve our understanding regarding the hardness of the MAPF problem. For instance, in Sec. 3 we discuss the dependency of the graph's size on the problem's complexity, and use it later to provide refined upper-bounds. We provide a new understanding that the worst-case complexity of CBS depends more on the graph's radius rather on the graph's size. Another such observation is reflected in the incorporation of positive constraints in our recurrence-based analysis. It focuses on the importance of positive constraints by demonstrating the huge reduction in the search tree's size.

We hope that these observations would allow, in addition to improving the theoretical analysis of CBS, to be used to improve the algorithm in practice. One possible way to approach this task is to use these observations to develop heuristics. For instance, by favoring expansion of CT nodes with a large number of positive constraints or with a minimal amount of approximated work that is remained to be done by the algorithm.

Acknowledgments

We wish to thank Roni Stern and the anonymous reviewers of earlier versions of this paper for helpful discussions and comments that contributed to this paper. This research was partially supported by grants No. 102583, 2028142 from the Israeli Ministry of Science & Technology (MOST), and by grant No. 1018193 from the United States-Israel Binational Science Foundation (BSF).

References

Banfi, J.; Basilico, N.; and Amigoni, F. 2017. Intractability of Time-Optimal Multirobot Path Planning on 2D Grid Graphs with Holes. *IEEE Robotics and Automation Letters* 2: 1941–1947.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, E. 2015. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. *Int. Joint Conf. on Artificial Intelligence, IJCAI* 2015-January: 740–746.

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. K.; and Koenig, S. 2018. Adding heuristics to conflict-based search for multi-agent path finding. In *International Conference on Automated Planning and Scheduling, ICAPS*, volume 2018-June, 83–87. AAAI press.

Gordon, O.; Filmus, Y.; and Salzman, O. 2021. Revisiting the Complexity Analysis of Conflict-Based Search: New Computational Techniques and Improved Bounds. *CoRR* abs/2104.08759.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics* 4: 100–107.

Kaduri, O.; Boyarski, E.; and Stern, R. 2020. Algorithm selection for optimal multi-agent pathfinding. In *International Conference on Automated Planning and Scheduling, ICAPS*, volume 30, 161–165. AAAI press.

Li, J.; Felner, A.; Boyarski, E.; Ma, H.; and Koenig, S. 2019a. Improved heuristics for multi-agent path finding with conflict-based search. *Int. Joint Conf. on Artificial Intelligence, IJCAI* 2019-August: 442–449.

Li, J.; Harabor, D.; Stuckey, P. J.; Felner, A.; Ma, H.; and Koenig, S. 2019b. Disjoint splitting for multi-agent path finding with conflict-based search. In *International Conference on Automated Planning and Scheduling, ICAPS*, 279–283. AAAI press.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019c. Symmetry-breaking constraints for grid-based multi-agent path finding. *Symp. on Combinatorial Search, SoCS* 184–185.

Pallottino, L.; Scordio, V. G.; Bicchi, A.; and Frazzoli, E. 2007. Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems. *IEEE Trans. Robotics* 23(6): 1170–1183.

Pemantle, R.; and Wilson, M. C. 2008. Twenty Combinatorial Examples of Asymptotics Derived from Multivariate Generating Functions. *SIAM Rev.* 50(2): 199–272.

Salzman, O.; and Stern, R. 2020. Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 1711–1715. International Foundation for Autonomous Agents and Multiagent Systems.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence* 195: 470–495.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symp. on Combinatorial Search, SoCS* 151–158.

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *Trans. on Computational Intelligence and AI in Games* 4: 144–148.

Wilf, H. S. 2006. *Generatingfunctionology*. A. K. Peters, Ltd.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *Artificial Intelligence* 29(1): 9–20.

Yu, J. 2016. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robotics and Automation Letters* 1: 33–40.

Zhang, H.; Li, J.; Surynek, P.; Koenig, S.; and Satish Kumar, T. K. 2020. Multi-agent path finding with mutex propagation. In *International Conference on Automated Planning and Scheduling, ICAPS*, volume 30, 323–332. AAAI press.