

A EECBS Analysis

Apart from the actual high and low-level searches, EECBS can utilize the many different CBS improvement mechanisms that have been robustly validated in 2D gridworlds as listed in Section 5.2. Table 2 shows a large ablation study on the base map where we remove or add a single CBS optimization to analyze its effect on performance. We ran the same experiments on a 2D warehouse map (warehouse-10-20-10-2-1) and briefly verified similar results except for SIPP described later. For EECBS with $w_{so} = 5$, we found that SIPP+BP+Target performed the best and used it as the reference method accordingly (top row), and for $w_{so} = 1.02$, we found that all improvements enabled performed the best. Each row consists of experiments done over 4 different scenario settings. The first “start-goal” column denotes results run on start-goal problems, while the rest three are g start-goal-start problems which denote that g agents share the same intermediate goal locations. For each of these problem settings, we ran an increasing number of agents in step-sizes of 50 for $w_{so} = 5$ and step-sizes of 10 for $w_{so} = 1.02$ and reported two aggregate statistics. The “Max #” column is the maximum number of agents that could be run with $\geq 50\%$ success rate over 10 seeds within the 60-second timeout. The “Slowdown” column is the relative median slowdown of the addition/removal of a single optimization with respect to the referenced best configuration.

We also provide an EECBS parameter improvement deep-dive in Table A1 for $w_{so} = 5, 1.02$. Each row describes one of the improvements with its associated numerical insight in the right 3 columns, e.g. Prioritizing Conflict takes an average of 140 attempts where only 1 of which succeeds and the total overhead occupies 9.25% of the total runtime when running EECBS $w_{so} = 5$ (Best+PC). We follow the results from Tables 2 and A1 with an individual EECBS parameter analysis.

SIPP—Removing SIPP dramatically hurts performance by lowering the max # of agents and drastically slowing down runtime. Additionally, SIPP is more impactful in the high suboptimality than the low suboptimality instances as using SIPP in EECBS with $w_{so} = 5$ gets speedups of $20\times$ and above while in EECBS with $w_{so} = 1.02$ reports the speedup is below $2\times$. This is because the high suboptimality problems shift attention from the high level to the low level, thus, having a fast low-level planner can boost performance significantly. SIPP’s percent overhead seems large but is misleading. SIPP’s overhead ranges between 0.1 to 2 seconds, therefore SIPP’s percent overhead is large on easy-to-solve problems (e.g. total runtime less than 2 seconds) but is small on difficult problems with more agents.

Bypass—Bypassing conflicts is another generally beneficial optimization, especially in high suboptimality problems. This is because the high suboptimality allows a higher chance for a child CT node to satisfy as a valid BP conflict, producing a high success rate (59 out of 482). In low suboptimality, BP makes much fewer successful attempts (15 out of 7151) due to the lower chance of being pruned. Moreover, BP only requires a negligible detection time, which is mainly why it’s beneficial for different suboptimalities.

Prioritized Conflicts—Adding prioritized conflicts *hurts*

w_{so}	Improvements	# atp	# succ	% Overhead
5	SIPP	N/A	N/A	24.67
	Bypass	482	59	N/A
	PC	140	1	9.25
	CR (≤ 650)	257	0	5.17
	CR (700)	690	17	1.42
	Target (≤ 300)	39	0	0.7
	Target (350-700)	381	35	1.6
1.02	WDG	268	N/A	6.55
	SIPP	N/A	N/A	12.1
	Bypass	7151	15	N/A
	PC	3615	1951	6.72
	CR	3637	275	32.16
	Target	3637	21	0.63
	WDG	7120	N/A	19.59

Table A1: Each CBS improvement is associated with 3 metrics, the number of attempts made to detect/build (# atp), the number of successful activations from those attempts (# succ), and the corresponding percentage of runtime with respect to the total runtime (% Overhead). We present two groups of statistics one for each suboptimality aggregated across all successful runs on the base map to reveal the change of each parameter.

performance in high suboptimality $w_{so} = 5$, but benefits performance in low suboptimality $w_{so} = 1.02$. Prioritized conflicts requires constructing an MDD which in 3D is a large overhead (9.25% of runtime) that outweighs its benefits as it is barely used (1 out of 140 is successful) in high suboptimality problems. However, PC is still useful in small suboptimality problems due to a much higher activation rate (1951 out of 3615) so that the benefits outweigh the overhead (6.72%). Similar behavior was observed in the 2D warehouse, but not investigated in depth. In some particular problems in 3D with a large number of agents, creating the MDD takes an incredible amount of time shown in our experiments that are not shown in the tables.

Corridor Reasoning—Adding corridor reasoning hinders performance to a small extent in high suboptimality $w_{so} = 5$, but greatly benefits performance in low suboptimality $w_{so} = 1.02$. Even though the 3D warehouse seems to have many corridors, Table A1 shows they are barely used (0 corridor conflict is detected until reaching 700 agents) in $w_{so} = 5$. This occurs due to EECBS’s high suboptimality which allows agents to traverse other corridors and prevent conflicts. However, this improvement does show effectiveness in low suboptimality 3D scenarios (275 out of 3647 conflicts are activated), especially in the start-goal problems. This is because the occurrence of corridor conflicts takes place more often when more agents are present. Thus, an easier problem (start-goal) enables a higher allowance of agents for which corridor reasoning can be effective. In addition, there exist some particular hard problems for which detecting corridor conflicts causes a drastic runtime slowdown. These outliers drag the average overhead to 32.16% in $w_{so} = 1.02$.

Target Reasoning—Removing Target Reasoning hurts

NodeID	X	Y	Z
0	1.2	3.5	2.2
1	1.2	4.4	2.2
2	1.2	4.4	1.1
3	3.1	3.5	2.2

EdgeID	Start	End	Bidirectional
0	0	1	True
1	0	2	False
2	1	2	True

Table A2: Examples of node and edge information that define a 3D scenario. Edges are defined by their start and end NodeIDs. Notice that in our warehouse scenario, all edges are grid-aligned but this is not a requirement for arbitrary 3D scenes.

performance, especially seen in smaller group-size problems. Table A1 shows that when we have a problem with ≤ 300 agents (Target (≤ 300)), 0 out of the 39 attempts are successful. On the other hand, when we have a problem with 350-700 agents (Target (350-700)), 35 out of the 381 attempts are successful on average. We deduce that similar to corridor conflicts, target conflicts are heavily distributed on the problems that contain more agents.

Weighted Dependency Graph Heuristic—Adding the WDG heuristic decreases performance under high suboptimality. This is because there is no need to build more informative heuristic information when the suboptimality is large. In addition, there exists some particular complex problem for which to build a WDG that causes a drastic runtime slowdown.

One interesting observation (not shown in the table) relevant to all EECBS runs is that there seem to be some “easy” and “hard” problems within the same scenario family. Specifically, EECBS with $w_{so} = 5$ on start-goal problems with 700 agents solved 5 seeds in 6 seconds and timed out (over 1 min) on the rest 5 seeds. This behavior was the main reason we needed to use 10 seeds (compared to most prior work which uses 3 or 5). This has been reported in 2D warehouse maps (Cohen et al. 2017) with ECBS without improvements, however, our 2D runs with EECBS with improvements did not encounter such behavior (i.e. runtimes are pretty consistent within instances).

B Formulation of 3D Maps

Unlike 2D environments which are typically represented by 2D voxels (i.e. grid cells), the 3D environment is represented via an adjacency list due to its relatively sparse nature. Edges in the 3D graph additionally can be specified as uni-directional, which although possible is practically never done in 2D. Each node has a unique id and (x, y, z) location, while each edge has a (startNodeId, endNodeId) and a boolean specifying whether it is a bi-directional edge or not. Table A2 shows an example of some grid-aligned nodes and edges which can define a 3D scenario. Our 3D warehouses have all bidirectional grid-aligned edges which are unit-length.

C Map Visualization

Visualizing 2D grid-world scenarios is taken for granted as packages like Matplotlib (Hunter 2007) can easily render them as 2D images. However visualizing large 3D scenar-

ios is non-trivial; specifically, Matplotlib cannot handle 3D plots with thousands of vertices and edges and becomes unusable fast. Luckily, we found Plotly’s (Plotly Technologies Inc. 2015) 3D graphing capabilities substantially better and sufficient for visualizing our 3D warehouses with more than 10K nodes and edges. An additional benefit of Plotly is its ability to create and save 3D plots/animations as HTML files which easily allows others to interactively visualize the scene on their browser without extra installations. Readers are encouraged to interactively view the corresponding visualization files in the supplementary material.

Finally, we present a graphical visualization of all 15 maps in our benchmark suite. The description for each map is formatted as follows: aisleLength-numAisles-aisleGap-numLevels-levelGap-numElevators. For example, 50-10-5-10-5-2 can be interpreted as an aisle length of 50, 10 aisles, aisle gap of 5, 10 levels, level gap of 5, and 2 elevators. We also show the number of nodes and edges for each map.

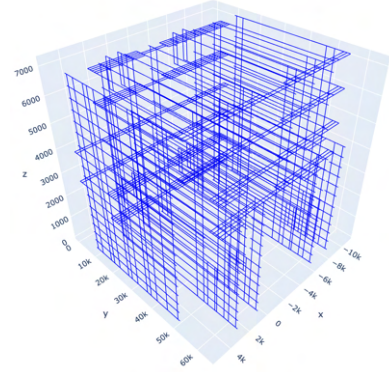


Figure A1: **Real-world map; Nodes: 12272; Edges: 13307.** This is a visualization of the real-world map provided by an industrial warehouse company. Note that it shares numerous similarities to our base map: the number of nodes and edges are roughly similar, both maps have a cubicle shape, and the structural arrangement of elevators, margins, etc. shows a high resemblance.

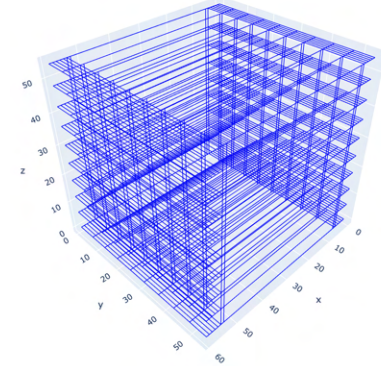


Figure A2: **Base map: 50-10-5-10-5-2; Nodes: 13200; Edges: 14540**

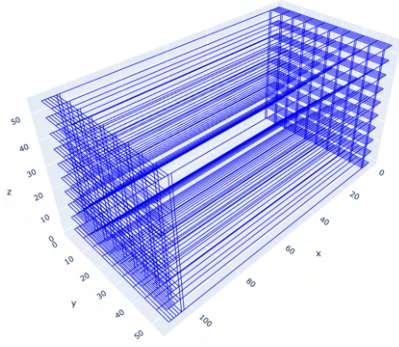


Figure A3: **Long aisles:** 100-10-5-10-5-2; **Nodes:** 18200;
Edges: 19540

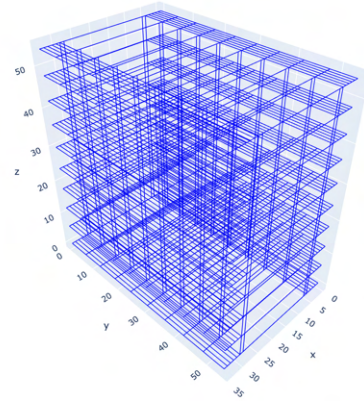


Figure A6: **Short aisles:** 25-10-5-10-5-2; **Nodes:** 10700;
Edges: 12040

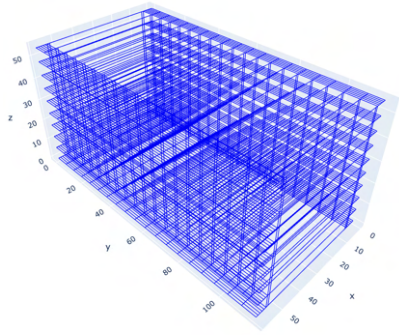


Figure A4: **More aisles:** 50-20-5-10-5-2; **Nodes:** 27000;
Edges: 29800

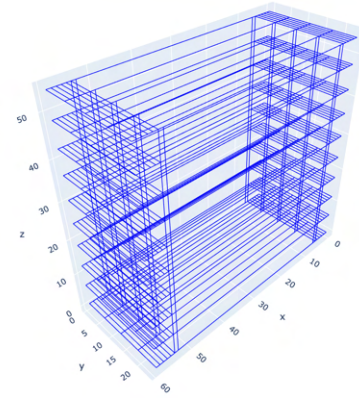


Figure A7: **Fewer aisles:** 50-5-5-10-5-2; **Nodes:** 6300;
Edges: 6910

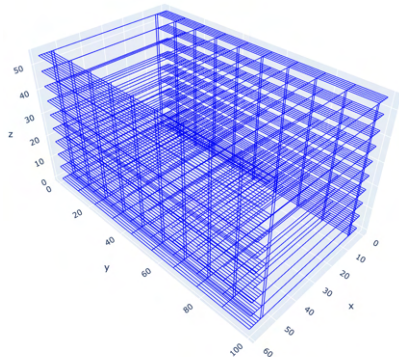


Figure A5: **Far aisles:** 50-10-10-10-5-2; **Nodes:** 18600;
Edges: 19940

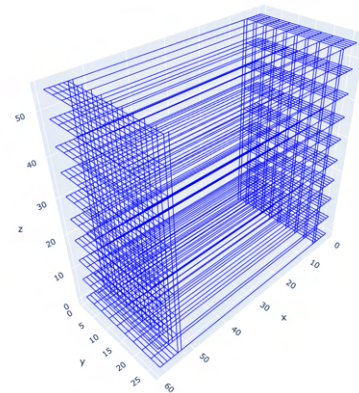


Figure A8: **Close aisles:** 50-10-2-10-5-2; **Nodes:** 9960;
Edges: 11300

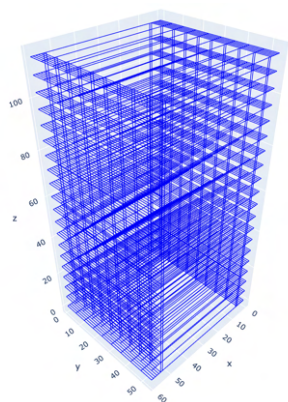


Figure A9: **More levels:** 50-10-5-20-5-2; **Nodes:** 26600; **Edges:** 29320

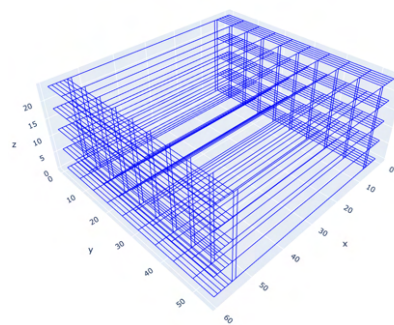


Figure A12: **Fewer levels:** 50-10-5-5-5-2; **Nodes:** 6500; **Edges:** 7150

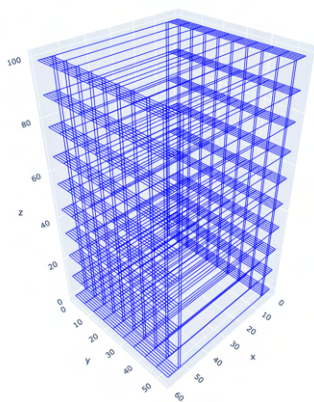


Figure A10: **Far levels:** 50-10-5-10-5-2; **Nodes:** 15000; **Edges:** 16340

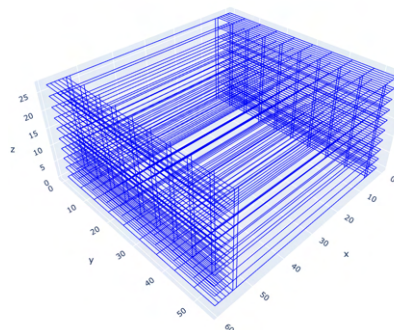


Figure A13: **Close levels:** 50-10-5-10-2-2; **Nodes:** 12120; **Edges:** 13460

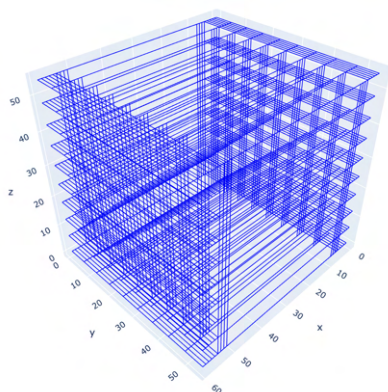


Figure A11: **More elevators:** 50-10-5-10-5-4; **Nodes:** 15000; **Edges:** 16700

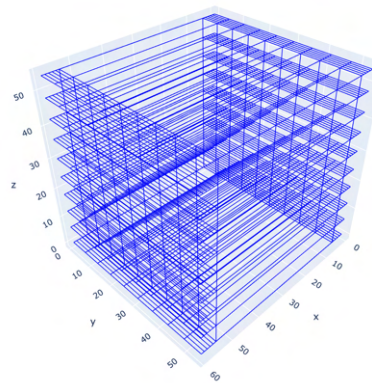


Figure A14: **Fewer elevators:** 50-10-5-10-5-1; **Nodes:** 12300; **Edges:** 13460

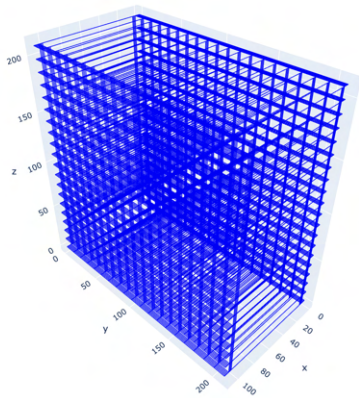


Figure A15: **All doubled:** 100-20-10-20-10-4; **Nodes:** 120000; **Edges:** 127200. The reason why this map does not look cubical is because the y and z axes are scaled up by a factor of 4: doubling both # of aisles and aisle gap results in a quadrupled scaling in the y dimension. Similarly, doubling # of levels and level gap results in a quadrupled z dimension. However, on the x dimension, only doubling the aisle length can affect the size, doubling # of elevators does not impact the size at all. Hence, the x dimension is only scaled up by 2 overall, resulting in a non-cubical look of the map.

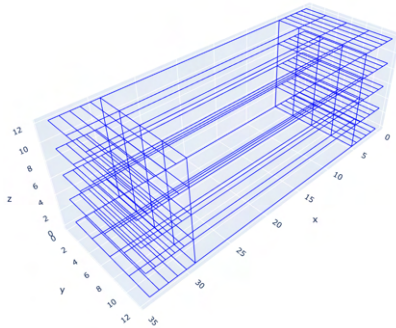


Figure A16: **All halved:** 25-5-2-5-2-1; **Nodes:** 1435; **Edges:** 1690. For a similar reason, as the “all-doubled” map, the y and z dimensions are scaled down by 4 and the x dimension is scaled down by 2, showing a non-cubical structure with slightly longer aisles.