Voxel Benchmarks for 3D Pathfinding: Sandstone, Descent, and Industrial Plants

Thomas K. Nobes[1], **Daniel D. Harabor**[1], **Michael Wybrow**[1], **Stuart D.C. Walsh**[2]

[1]Faculty of Information Technology, Monash University, Melbourne, Australia
[2]Faculty of Engineering, Monash University, Melbourne, Australia
{thomas.nobes, daniel.harabor, michael.wybrow, stuart.walsh}@monash.edu

Abstract

Voxel grids are an increasingly common enabler for pathfinding in 3D spaces. Currently in this area there exists only a limited number of publicly available benchmarks. This makes it difficult to establish state-of-the-art performance and to compare the strengths and weaknesses of competing search techniques. In this work, we introduce three new and diverse sets of voxel benchmarks intended to help fill this gap. We further describe our methodology for generating and selecting a representative set of pathfinding queries. Our dataset comprises 46 distinct voxel maps and 92,000 problem instances. The data is drawn from distinct application domains: computer video games, industrial plant layouts and sandstone porosity scans. Featuring distinctive geometric properties and a variety of challenging query types, these new datasets allow practitioners to evaluate algorithmic performance across a variety of settings encountered when pathfinding in practice.

Introduction

Computing a shortest path from A to B is a common problem in many practical contexts, such as robotics, pipe routing, and computer video games. While pathfinding systems in two dimensions (2D) have been studied extensively, pathfinding in three dimensions (3D) remains relatively unexplored. As computing optimal paths in true-3D has been shown to be NP-hard (Canny and Reif 1987), the typical strategy is to approximate the 3D space by constructing a graph and then applying a conventional search algorithm, such as A*.

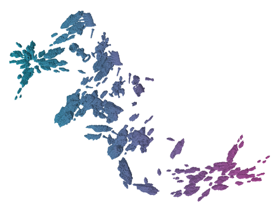
Voxel grid maps are a simple approximation technique where 3D space is partitioned into unit-sized cubes. Each voxel has up to 26 neighbours, and the total number of voxels is the width $\times$ height $\times$ depth of the map. Among the advantages of voxel grids is that they enable high-quality solutions and allow direct reasoning about the height and vertical positions of 3D agents. This means voxel-based planners can solve a broader range of 3D pathfinding queries than some competing techniques; e.g., *pseudo-3D* methods where 3D agents plan on a 2D navigation mesh (Noonchester 2019). Another competing approach, Sparse Voxel Octrees (SVO), recursively partition 3D space into octants of increasingly smaller size (Schwarz and Seidel 2010). Paths in the SVO structure are defined in terms of octant centres, which means they may introduce long detours and have higher cost than equivalent solutions computed with voxel grids. Such considerations are important for a variety of 3D applications, such as computer video games where agents need to appear intelligent (Beig et al. 2019).

Recent works which have had success searching on voxel grids include studies in aerial drone navigation (Liu et al. 2017; Zhang 2021; Zhang, Zhang, and Low 2021), pipe routing (Min, Ruy, and Park 2020) and pathfinding for 3D computer video games (Brewer and Sturtevant 2018; Min 2019; Nobes et al. 2022). Yet despite a variety of ideas and perspectives, the state-of-the-art in this area is unclear. The only publicly available voxel dataset that we are aware of is the Warframe benchmark (Brewer and Sturtevant 2018), a collection of 44 maps drawn from the epoynmous space combat game by developer Digital Extremes. This benchmark features large voxel worlds with complex geometry. However, the game setting and generated problem sets reflect only a narrow range of practical interests.

For a contrast consider the related area of 2D pathfinding, where a variety of publicly available benchmarks have been proposed (Demetrescu, Goldberg, and Johnson 2009; Sturtevant 2012; Sturtevant et al. 2019; Harabor, Hechenberger, and Jahn 2022). Some 2D domains are drawn from real applications (several types of game worlds, rasterised maps of city streets and real-world road networks) and these are designed to test practical performance. Other 2D domains are synthetically generated testbeds (maze, room, random), and these are designed to understand pathological behaviour of search algorithms. These benchmarks help to establish clear performance indicators and they serve to highlight the different strengths and weaknesses of competing algorithms.

To fill the gap between 2D and 3D pathfinding, we propose three new voxel benchmarks. Featuring a wide a variety of geometrical characteristics, our maps are drawn from diverse applications: (i) geological rock samples (Sandstone); (ii) industrial plant layouts (Plants); (iii) 3D games with indoor navigation (Descent). There are a total of 46 maps and 92,000 problem instances, with map sizes ranging from several million voxels to over one billion. The voxel maps and sets of problem instances are available for download[1]. We

[1]https://bitbucket.org/shortestpathlab/benchmarks/


(a) Warframe: *FA2*
826 × 493 × 574

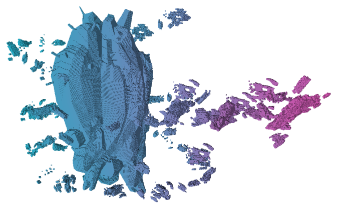

(b) Warframe: *C2*
1093 × 647 × 725

Figure 1: Voxel maps from the Warframe dataset (Brewer and Sturtevant 2018): (a) FA2; and (b) C2, a C-map variant.

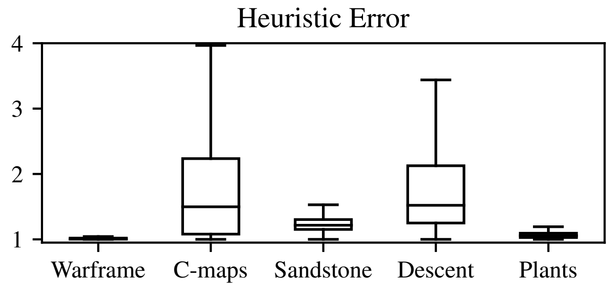


Figure 2: Heuristic error across each benchmark data set.

describe the origins of each data set and present a novel methodological approach for generating a representative and unbiased set of pathfinding queries. We then compare the performance of two voxel-based search algorithms to highlight differences across the three domains.

Existing Benchmarks

In this section, we summarise the position of existing voxel benchmarks in the literature, and draw attention to key characteristics that we identify are missing from current testing suites available to the search community.

Warframe (Brewer and Sturtevant 2018) is a set of 44 publicly available voxel grid maps, taken from the eponymous online space combat game. Maps range in size from several million voxels to hundreds of millions of voxels. Each map typically features asteroids of different sizes floating in space. A small number of maps also have spaceships, which have internal structure that can be navigated. Figure 1a and Figure 1b show examples of each type. For performance evaluation, each map comes with a set of 10,000 pre-generated problem instances (referred to equivalently as *queries* or start-target pairs).

The Warframe benchmark has appeared in a variety of papers. Some authors evaluate only a single map named "Complex" (Muratov and Zagarskikh 2019; Saha et al. 2022; Tabacaru, Atzmon, and Felner 2022) , a small test case of around 8 million voxels, while others (Nobes et al. 2022) report results for a large majority (40) of maps. In each case Warframe provides a valuable reference point for testing and comparing 3D search algorithms. The benchmark is also valuable for the community as it documents the main challenges that arise in a specific application setting. Yet in the broader context of 3D pathfinding, the Warframe problems are narrowly focused. In particular, there is a need for a greater variety of maps, so that benchmark data more closely reflects the diverse range of real-world problems that motivate works in this area. There is also a need for more diverse problem instances, so that researchers can explore and evaluate the full range of planner performance in each type of application. We now explore these issues in turn.

(1) Problem Variety: Each map in the Warframe benchmark is composed of sparsely scattered asteroids, debris and sometimes spaceships, floating in open space. Each problem instance is is a start-target pair of voxels, both within 5 voxels of an obstacle. Being set in space, there usually exist many valid approaches to solving each problem (e.g., over, under or around obstacles). These attributes provide an interesting dataset for researchers and practitioners interested in open-world 3D computer game traversal.

Yet works on 3D pathfinding are motivated by a variety of diverse applications which often do not share strong similarities to open video-game traversal. Practitioners working on e.g., indoor (Li et al. 2018) and outdoor (Zhang, Zhang, and Low 2021) drone navigation or the design of chemical processing plants (Belov et al. 2020) often consider more "grounded" queries (from building-to-building or module-to-module) where paths cannot go underneath obstacles and cannot always bend easily around. Even for computer games, developers typically distinguish between the types of queries presented by "outdoor" 3D navigation (which Warframe closely resembles) and "indoor" 3D navigation (which Warframe does not).

In other words, effective performance on Warframe is not necessarily indicative of effective performance elsewhere. Having a diverse set of benchmarks, including ones drawn from real applications, helps the community to more clearly identify the strengths and weaknesses of competing planners. For example, there are several initiatives in the AI community that track progress on a variety of problems and application domains (Long and Fox 2006; Stuckey et al. 2014; Sturtevant et al. 2014; Shen et al. 2023). These works help to establish the state-of-the-art, they indicate where the remaining challenges are, and they stimulate further research.

(2) Problem Difficulty: When evaluating and comparing pathfinding algorithms, it is important to understand performance across the full range of query-availability inside a problem setting; i.e., not only on average but also in the worst case. This is because practical applicability is often dependent on the worst case. In the context of video games for example, developers create customised benchmarks that automatically generate and test up to millions of problems instances (Gillberg 2019). Analysing worst-case results from such experiments helps developers identify problems in map or game-play design, and can lead to changes in the pathfinding system itself in order to guarantee a "quality of service" for players. However, continuously generating and solving millions of test problems is prohibitive. For this reason, publicly available benchmarks

Maps	#	Instances	A*	JPS-3D
Warframe				
Test	2	20,000	0.01	0.01
C-maps	4	40,000	640.95	296.91
Standard	38	380,000	15.53	10.64
New				
Sandstone	11	22,000	8.44	6.06
Descent	30	60,000	48.17	11.90
Plants	5	10,000	4.65	1.78
Totals	46	92,000	61.26	19.83

Table 1: Cumulative runtime (hours) for A* and JPS-3D across each benchmark. Totals exclude Warframe.

such as Warframe are highly valuable: they provide a smaller but still representative test set that highlights the main challenges in a specific domain (i.e., the problems range in difficulty from very easy to very hard).

When discussing the difficulty of solving a given pathfinding problem it is common to refer to heuristic error and baseline expansion cost. Heuristic error is calculated as the ratio between the true cost of the optimal path, from start to target, and the estimated heuristic cost at the start node. We use the *voxel heuristic* estimator, which returns the minimum (26-connected) distance between two voxels while ignoring all obstacles. Baseline expansion cost meanwhile is the number of node expansions required by a comparison algorithm to solve a given problem. Here we use A* search.

In Figure 2 we measure the heuristic error for all problem instances on 38 of the 44 maps Warframe maps (excluded maps are discussed next). We observe that the median error is 1.007 on these 38 maps; i.e., the cost increase, from start node estimate to true optimal, is only 0.7%. In general, smaller heuristic error indicates problems are easier to solve. Table 2 confirms this—we see that the median runtime for our baseline A* solver is just 2.58ms. We observe similar trends with low A* node expansion cost in Figure 8. Despite Warframe maps having hundreds of millions of traversable voxels (see Figure 4), A* must expand a median of only thousands of nodes during search. In other words, when comparing across Warframe data, there are limited opportunities for new and emerging pathfinding algorithms to demonstrate substantial improvements. This is because the baseline A* algorithm is already competitive in most cases.

A significant exception to these observations is a set of 4 maps from Warframe which we refer to as the *C-map* variants. These maps feature a massive *Fomorian* capital ship, which has a complex interior that players are required to fly inside (Figure 1b). This provides distinctive *interior-to-exterior* (and vice versa) queries. Due to the size of the capital ship, these maps are several factors larger than the rest. The median heuristic error for these maps is 1.5; which indicates that these problems are substantially harder to solve. The two remaining maps (“Simple” and “Complex”) are an order of magnitude smaller than the next largest map. We refer to these maps as *Test* variants, as they feature only a single floating object. Queries on these maps are very simple and much easier to solve than other problems in the Warframe set. In Table 1 we report cumulative runtimes for A* search when solving each of these 3 distinct subsets. We see that solving C-maps instances take orders of magnitude longer than for the 38-map majority set (up to *hundreds* of seconds for the very hardest instances), while the test instances are orders of magnitude faster than the 38-majority set. In other words, aggregating across all of Warframe, or only testing only specific subsets, may produce misleading performance indicators. We suggest that performance measures should be reported separately on the 38-map majority, the 4 C-maps, and the 2 Test maps in the Warframe benchmark, due to their distinctive characteristics.

Another way to assess the difficulty of a given problem is to measure the amount of *directional bias*. This occurs when a “hard” start-to-target instance becomes an “easy” target-to-start problem. Problem instances which suffer from directional bias may be challenging but in uninteresting ways. Figure 5 measures the ratio of forward expansions to backward expansions across the 42 considered Warframe maps. We see that directional bias is prevalent. In other words, drawing comparisons between algorithms that solve these instances may lead to incorrect conclusions about strengths and weaknesses of competing techniques.

New Diverse Voxel Benchmarks

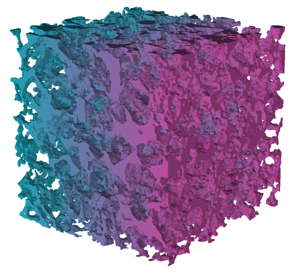
In the following sections, we outline three new benchmarks for evaluating three-dimensional pathfinding algorithms. Each benchmark involves separate geometric characteristics that test distinct aspects of the three-dimensional pathfinding problem.

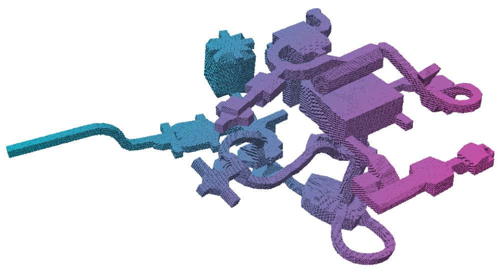
1. The first set of benchmarks are derived from tomographic scans of natural sandstone samples. These maps consist of densely packed multiply-connected percolating pathways through the rock samples.
2. The second set of benchmarks are obtained from the *Descent* video game. These examples consider long three-dimensional corridors and tunnels representative of interior spaces and mine-shafts.
3. The final set contains industrial plant layout benchmarks based on real pipe routing problems. These problems involve pathfinding through combinations of large and narrow voids around regular geometric obstacles—representative of factory layouts or cityscape structures.

The following sections provide more information on each of these benchmark geometries. Later, we describe how instances of pathfinding problems with a representative range of difficulties are obtained from these maps.

Sandstone Porosity Scans

The first set of pathfinding maps is derived from X-ray tomographic computed images (XRCT) of natural sandstone samples. The scenarios are based on scans of 11 sandstone plugs originally provided by Kocurek Industries (Kocurek Industries Inc. 2021), which were imaged and processed by Neumann et al. (2021). The samples include Bandera Gray, Parker, Kirby, Bandera Brown, Berea Sister Gray, Berea Upper Gray, Berea, Castlegate, Buff Berea, Leopard and Bentheimer sandstones.


(a) Sandstone: *Berea*
400 × 400 × 400


(b) Descent: *Level03*
1042 × 854 × 637

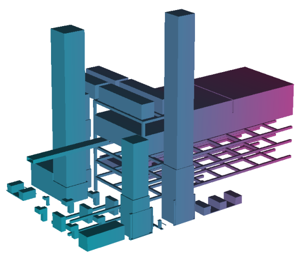

(c) Industrial Plants: *Plant03*
731 × 445 × 477

Figure 3: Examples of voxel maps from each of the new benchmark data sets: (a) sandstone, (b) Descent, and (c) industrial plants. For visual clarity, we draw free voxels rather than obstacles for sandstone maps (∼80% of the domains are filled-in).

Neumann et al. (2021) produced publicly-available binary images of the connected pore space within each sample for the purpose of numerical flow simulations and permeability predictions. Each of the raw binary images has voxel dimensions of 1000 × 1000 × 1000. To keep the size and the computational cost of these maps comparable to existing benchmarks, we take a 400 × 400 × 400 subset of the original image, resulting in maps with 64 million voxels each. We construct the voxel maps directly from the binary data, treating the sandstone itself as an obstacle, and the pore space as traversable terrain. Traversable voxels compose a single connected region, assuming a 6-connected neighbourhood. We define 11 maps with 2,000 pathfinding instances each, for a total of 22,000 problems. The pore-space geometry is characterised by winding narrow corridors and natural non-symmetric shapes resulting in an environment reminiscent of random grid maps.

Descent

The second set of scenarios is derived from in-game levels from *Descent*, a video game developed by Parallax Software and released by Interplay Productions in 1995. It is recognised as the earliest FPS to feature entirely true-3D graphics, and is now open source (Guinness World Records 1995). Descent is well-suited for search as each level has been designed specifically for 3D maneuverability and traversal; featuring rooms and branching corridors. We adapt the 27 in-game levels and 3 secret levels into voxel-world maps. The benchmark consists of 30 maps (one for each level) with 2,000 instances each, for a total of 60,000 problem instances.

Level descriptions are read from in-game files. Positional data from the game is continuous, and thus we convert each map into a discrete voxel-world equivalent. We use an online voxeliser (Westerdiep 2022) with 0 shell thickness and match the output voxel dimensions to the in-game units (as an integer). This conversion ensures that the interior traversable region of the original level remains one contiguous, enclosed region. The interior of the level only accounts for a small proportion of the total size of the map, where there is a significant exterior region of open space. A one-voxel border is added around all edges of the map so that the exterior is also one contiguous region (to enable exterior queries). We, however, generate instances solely from the internal traversable region which corresponds to the in-game playable area.

Industrial Plant Layouts

The final set of benchmarks is designed to mimic industrial plant design and pipe routing problems. Using real data from chemical processing plants, we construct voxel maps that feature layouts composed of many types and scales of equipment modules, safety and engineering zones, support units, and more. Some of these layouts have appeared previously in the literature (Belov et al. 2020), while others are entirely new. The benchmark data has been anonymised; it uses bounding boxes instead of original component geometry and it features layouts designed for a mix of both demonstration and practical purposes. The original geometry of each plant layout is described at millimetre precision. The corresponding size of the voxel grid maps at this resolution is prohibitive; up to 924 trillion voxels are required to encode the largest map, taking terabytes of space if stored explicitly. Instead, we construct geometry at 10cm resolution which produces map sizes in the hundreds of millions of voxels.

The benchmark consists of five unique plant layouts with 2,000 routing problems each for a total of 10,000 problem instances. Instances are generated by choosing at random start and target locations from among those voxels immediately adjacent to the surfaces of equipment modules. This simulates routing pipes between two fixed attachment points on corresponding modules. We exclude several areas such as safety or truck access zones, interiors of racks, and many miscellaneous equipment parts such as equipment skirts (lower supports for large vessels) from potential start/-target points. This decision better ties the resulting problem instances to valid pipe-routing scenarios. In addition, pathfinding between the often ground-attached cuboid obstacles is a close analogue to the problem of navigating aerial drones through cityscapes.

Size and Traversability

Figure 4 shows the distribution of voxel sizes of the maps and their traversable regions across each benchmark. Traversable regions are computed by performing a 6-connected (i.e., North/South, East/West, Up/Down) flood

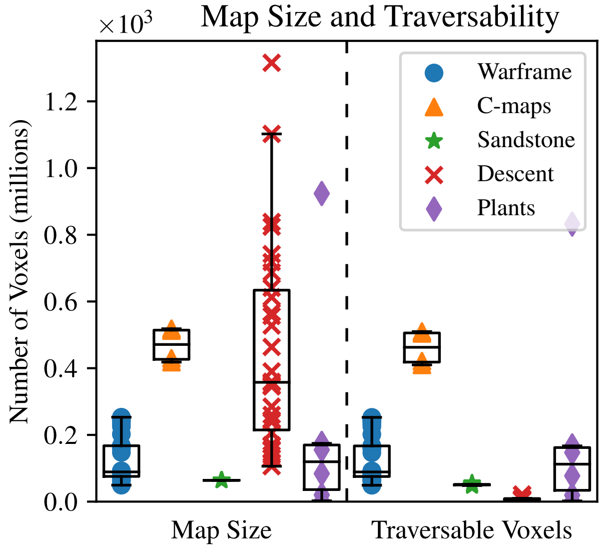


Figure 4: Size of maps and traversable regions across each benchmark. All Sandstone maps are 64M voxels. Warframe and C-maps are included for comparison, but are prior work.

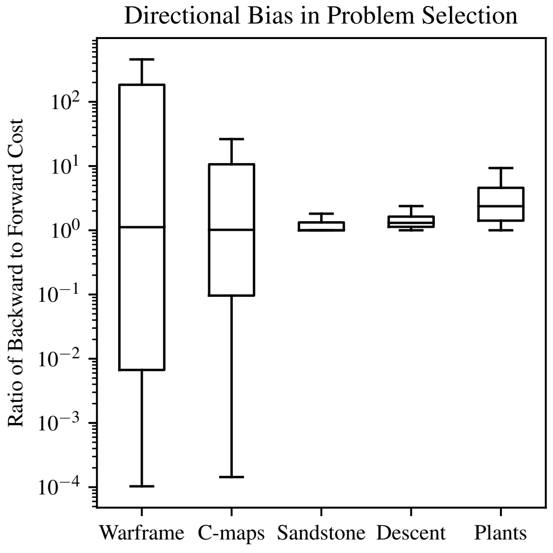


Figure 5: Directional-bias of problem selection in each data set. Cost refers to A* node expansions. Forward and backward refers to searching from start-to-target or vice versa, respectively. Ratios of directional-costs represent how many times harder (positive) or easier (negative) it is to solve the backward, target-to-start query. Note that the Sandstone, Descent and Plants benchmarks have been designed to control for the direction of bias.

filling procedure. Each voxel in the traversable region is reachable from any other voxel in the traversable region using either 6-connected or 26-connected movement. We assume optimal paths do not involve any diagonal moves through the corners or edges of a filled voxel (this is analogous to the "no-corner-cutting" rule in 2D grids). Note that by traversable space, we do not refer explicitly to the number of empty voxels. This is misleading as certain regions of free space are not reachable via search (i.e., are disconnected exterior, inside hollow obstacles, etc.).

By design, the sandstone maps have a uniform 64 million voxels. Obstacles make up between 73% and 86% of the maps. The remaining pore space is traversable, ranging from 9 to 17 million voxels. The Descent levels vary significantly in size, with the majority having hundreds of millions of voxels; the largest reaching 1.3 *billion* voxels. This size pertains to the required boundaries around the level's in-game geometry. Note that a small proportion of this space (only up to 17 million voxels) is part of the traversable interior. The industrial plant benchmarks also vary significantly in size, with the smallest having only 3 million voxels and the largest having 924 million. Notably, these maps have the greatest proportion of traversable space of the three benchmarks. This aligns with the geometric characteristics of sparse module-to-module routing across grounded vertical towers.

Generating Problems

We construct each map file in a unified format, consistent with existing 3D benchmarks (Brewer and Sturtevant 2018). To ensure the files are easy to read and parse by any language or application, each map file simply contains the type and dimensions of the map in plain text, followed by a list of voxels that are filled. Prior work uses only the type *'voxel'* to describe the map. For convenience, we introduce the type *'rev_voxel'*, which signifies that the voxels listed are instead empty rather than filled. This is relevant for dense benchmarks, such as the sandstone maps, where the majority of the domain is filled-in.

Our next goal is to generate a suite of pathfinding problem instances. Here, we hope to obtain a representative dataset: the problems are distributed evenly across potential problem difficulties introduced by unique voxel geometry. However, merely generating random start and target pairs from the voxel maps is known to yield biased samples that over-represent easy pathfinding problems (Harabor, Hechenberger, and Jahn 2022). A further challenge in 3D voxel grid maps is that it can be very expensive to generate and solve large numbers of random problems, due to the large domain sizes. It is therefore important to select a good subset of test problems: small enough so that evaluation is not prohibitive, but diverse enough to contain a large range of challenges from each map.

Our approach largely follows the scenario generation procedure presented by Harabor, Hechenberger, and Jahn (2022). This method combines large random sampling with k-means clustering to select a much smaller, but more representative test set. The same procedure is used to generate instances across all benchmarks. In this work, we randomly generate 100,000 unique problem instances, such that the standard error estimate of the mean is within ±1% of the sample mean, given a 99% confidence interval. Each problem instance is generated only from domain-dependent regions of voxels to support interesting types of queries. This region consists of all free space for the sandstone benchmarks, the playable interior of for each Descent level, and reachable voxels immediately adjacent to equipment modules within the industrial plants.

Next, we aim to select a representative sub-sample by ranking instances. In this work, we use the number of nodes expanded by A* as a measure of problem difficulty. This has been shown to better align with problems that other algorithms also find challenging than previous measures such as path length (Harabor, Hechenberger, and Jahn 2022). Each problem is solved with A* both forwards (start-to-target) and backwards (target-to-start). By solving each query in both directions, we can reduce misleading bias in problem difficulty (it is easier to search in one direction relative to the other) by taking the minimum cost as the ranking metric. This guarantees that the selected problems will be *at least* as hard as their stated difficulty. We then output each instance using the start-target ordering that corresponds to the minimum cost. We explore this issue further in the following section.

Returning to the problem of selecting a representative subset, once all sampled instances are ranked, we then group them into buckets. We perform k-means clustering on the buckets, taking the maximum cost bucket in each cluster and selecting instances with the highest expansion cost; all others are discarded. We use a bucket size of 200 and select the ten highest-cost instances, which yields a subset of 2,000 final instances. For more information, see Harabor, Hechenberger, and Jahn (2022).

We can see that the final set of problems has an even distribution of difficulty on a per map basis by comparing the bucket clustering approach to uniform grouping and selection (i.e., dividing the problems into 200 groups of roughly even size, and taking the 10 highest-ranked instances). Figure 7 shows the uniform strategy selects a majority of easy instances (low-difficulty), whereas k-means sampling returns a more even distribution across all levels of problem difficulty.

Directional Bias

Due to geometrical asymmetry, pathfinding queries can have significantly disproportionate difficulty depending on the direction of the search (see Figure 5); we call this property directional bias. Problems with large directional bias present a hollow challenge, whereby simply reversing the order of search can make the problem easy. In Harabor, Hechenberger, and Jahn (2022) problems are ranked according to the minimum cost of forward or backward search. This approach avoids classification errors for directionally-biased instances, in that it guarantees problems are at least as difficult as the minimum expansion cost. However, queries that have been ranked as easy may still be presented in a direction that is substantially harder, requiring many more expansions to solve (if the minimum cost was achieved by swapping the identity of the start and target). We demonstrate this problem in the following experiment.

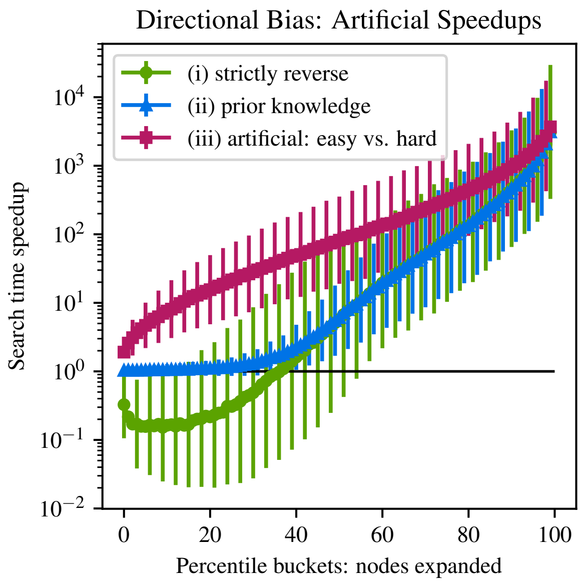


Figure 6: Artificial search time speedups of A* over *itself*, when informed vs. uninformed about directional bias (i.e., knows whether search is easier from start-to-target or vice versa). Results are reported on the Warframe benchmark. We plot the average speedup, bucketed by each percentile of node expansions by the baseline, uninformed A*. Error bars indicate standard deviation; for clarity, we show only every third.

Experiment 1 (Figure 6): Typically, we report runtime performance in terms of a given algorithm compared to a standard baseline. Here, we construct artificial comparisons where *both* the given and baseline algorithm is 3D A*. Both algorithms will solve the same problem instances, but we toy with the direction (start-to-target or target-to-start) that each algorithm searches. This emulates the advantages that can be gained from being informed about directional bias ahead of time. We report performance in terms of search time speedup: a speedup of 10^1 indicates that an algorithm is ten times faster than baseline A*; less than 10^0 indicates worse performance. We test three scenarios: (i) we strictly reverse (target-to-start) the search direction of all queries, irrespective of directional bias; (ii) we search in reverse only on advantageous instances based on prior knowledge of directional bias; (iii) we construct an artificial comparison where the given algorithm searches all queries in the easier direction, whereas the baseline instead searches in the hard direction (note: this is a different baseline to (i) and (ii)). Figure 6 shows that we are able to report significantly *different speedups* on the *same problems* while running the *same algorithm*, simply by playing with the direction of search. □

We note that it is not our suggestion to eliminate directionally-biased instances, for they are indeed important to evaluate for two reasons; (i) in practice, we do not know ahead of time which direction is easier or harder, and; (ii) practical performance can be significantly impacted by these types of queries. Instead, we identify that existing works do not consider this issue, and thus enable misleading conclusions by abusing information about directional bias. We hope to improve existing methodology by controlling for directional bias when generating benchmark instances.

Here, we deviate from prior work and contribute a novel method of problem generation: we provide problems that are explicitly biased in the forward direction of search (i.e., they are guaranteed to be at least as easy in the direction provided by the benchmark). The benefit of this method of problem selection lies in the nature of being informed *a priori* of inherent directional bias. This allows researchers to isolate this bias from aggregate performance evaluation, while retaining the ability to draw conclusions about practical performance on biased problems by simply running problems in reverse.

Evaluation

We test each new benchmark with pathfinding algorithms from the literature. For comparison, we also report results across the existing Warframe benchmark (Brewer and Sturtevant 2018). These experiments aim to provide a general indication of performance and examine trade-offs. We conduct experiments using the following algorithms:

- JPS-3D (Nobes et al. 2022). The leading planner for online grid search in 3D. We use a C++ implementation from the WARTHOG pathfinding library (Harabor 2015),
- Grid A* (3D). The standard baseline algorithm. Our C++ implementation shares a common base with JPS-3D.

Both algorithms use the *voxel heuristic*, which returns the minimum (26-connected) distance between two voxels while ignoring all obstacles (this is analogous to the well known octile distance heuristic from 2D grids). All experiments were performed on an Intel Core i7-8559U CPU with 32GB of RAM running Ubuntu 18.04.3 LTS.

Cumulative search time results are reported in Table 1. We further outline the distribution search time performance averaged across instances in Table 2. These provide an overview of the general performance across the benchmarks. We further observe that JPS-3D consistently outperforms A* across the problem sets. The cumulative search time across the four C-maps dwarfs the runtime of all other data sets. A similar observation, but in the other direction (smaller runtimes) is true for the set of Test maps. Noting substantial distinctions, we exclude the Test maps from further analysis, and report performance measures separately for the 4 C-map variants and 38 standard Warframe maps.

Figure 8 shows the nodes expanded by A* across the benchmark data sets. Here, we observe the greatest differentiation between the Descent and C-map problems: the upper 50% of the instances across the C-map variants expand significantly more nodes. Yet the traversable voxels in the Descent maps are an order of magnitude smaller than that of the C-maps. We observe that queries across the Sandstone, Descent and Industrial Plant maps consistently expand millions of nodes. A* expands substantially more nodes across the three new voxel benchmarks than the Warframe maps.

Figure 9 highlights the relative size of the frontier (the number of nodes generated during search) versus the number of nodes expanded during search. Results show that much of 3D search across the benchmarks expand a median of one node out of a 26-connected neighbourhood. The sandstone benchmarks are an exception, where there are substantially more nodes generated per node expanded. This suggests greater fill-in behaviour.

Conclusion

3D search using voxel grids is an increasingly important technological enabler in areas such as drone navigation, pipe-routing and video games. Yet, the research literature in this area contains few publicly-available data sets that can be used for evaluation purposes. Access to a diverse and challenging set of problems is necessary for understanding the strengths and weaknesses of competing planners, and for measuring the efficacy of these algorithms for solving the types of problems that appear in practice.

In this work, we introduce three new and diverse voxel benchmark data sets which are drawn from real applications. Each benchmark has distinct geometry and features unique types of problems that reflect the challenges of a specific domain. We produce 46 maps with a total of 92,000 problem instances. Our Sandstone benchmarks are not dissimilar to random obstacle maps and feature narrow and winding corridors with little symmetry. Our Descent benchmarks feature interior pathfinding in labyrinthine hallways and rooms designed specifically for 3D agent traversability. Our Industrial Plant layout benchmarks simulate real pipe-routing problems, and present similar geometry to aerial drone navigation through cityscapes.

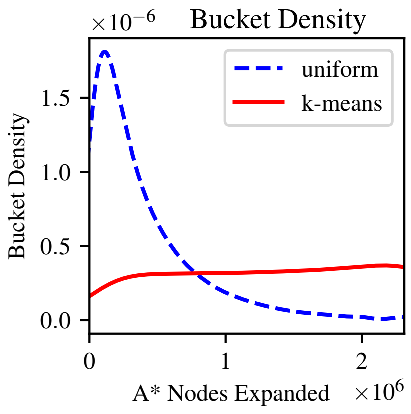
An important role of good benchmarks is to facilitate fair and informative comparisons between competing approaches. On the one hand, benchmark problems should represent the full spectrum of challenges in a target domain; on the other hand, benchmarks should control for confounding factors and biases that can produce misleading conclusions when making comparisons between competing techniques. In this work, we consider how to generate representative sets of benchmark problems, and we examine directional bias, a potential source of experimental error that can produce misleading conclusions. We show how directional bias arises, how it can influence experimental results, and we propose a novel strategy for controlling the bias so that practitioners can draw stronger conclusions.

One potential direction for future work is to generate alternate test sets that further differentiate problem types within a domain (e.g., queries with long paths, queries between specific areas, or queries intended to produce pathological behaviours in pathfinding algorithms). This would allow practitioners to more deeply investigate points of differentiation between algorithms. Another possibility is to optimise for the size of the test set: namely how large does a benchmark need to be to provide an indicator of performance on a specific map? This would reduce the computational burden required to make comparisons.

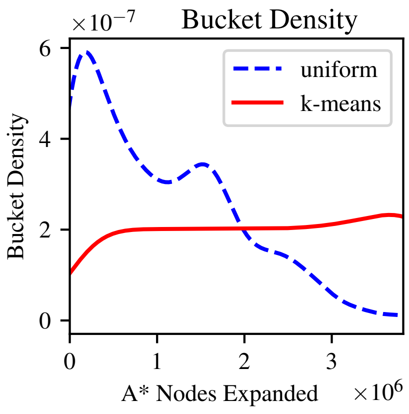
We hope that the benchmarks produced in this work will foster interest and facilitate the development of innovative methods for solving 3D search problems in voxel domains.

	Benchmark	#	A* Search Time (ms) Q1	Med	Q3	Max	JPS-3D Search Time (ms) Q1	Med	Q3	Max
Warframe	Test	2	6.71e-2	1.82e-1	1.15	6.12e1	1.13	1.67	2.03	1.05e3
	Standard	38	4.87e-1	2.58	4.03e1	1.65e4	1.37	5.23	2.20e1	1.82e4
	C-maps	4	3.35e2	**3.50e3**	**1.58e4**	**1.06e6**	1.36e2	**1.56e3**	**1.08e4**	**2.85e5**
New	Sandstone	11	5.50e2	1.21e3	1.91e3	1.17e4	**3.80e2**	8.33e2	1.37e3	8.99e3
	Descent	30	**9.19e2**	1.91e3	3.41e3	2.16e4	1.33e2	3.80e2	934e2	6.13e3
	Plants	5	6.92e1	5.52e2	2.05e3	4.19e4	1.99e1	1.48e2	7.67e2	1.47e4

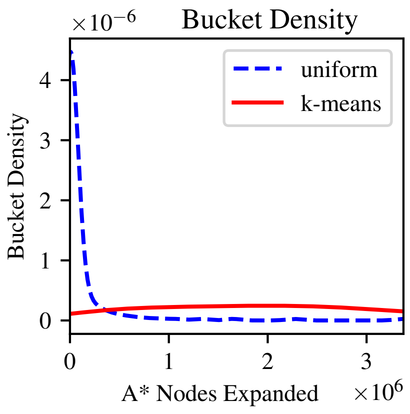
Table 2: Search time performance across each benchmark by A* and JPS-3D search. Column-highest values are bold.



(a) Sandstone: *BanderaGray*



(b) Descent: *Level17*



(c) Industrial Plants: *Plant04*

Figure 7: Representative problem selection across each benchmark. Problem instances are ranked according to A* node expansions and a sub-sample is selected via. either uniform sampling (dashed blue line) or k-means clustering (red line).

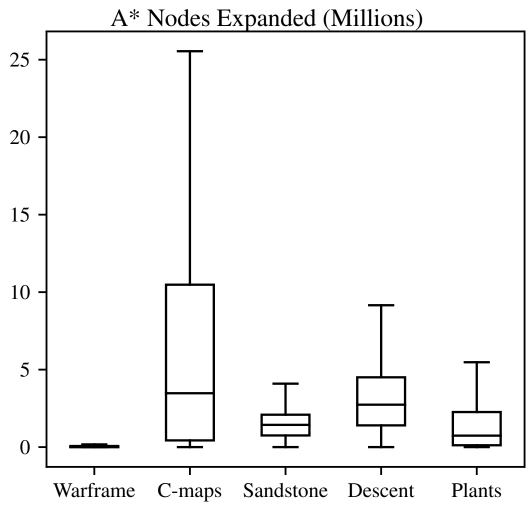


Figure 8: Box plots of nodes expanded by A*. This is a common measure of problem difficulty (higher means harder).

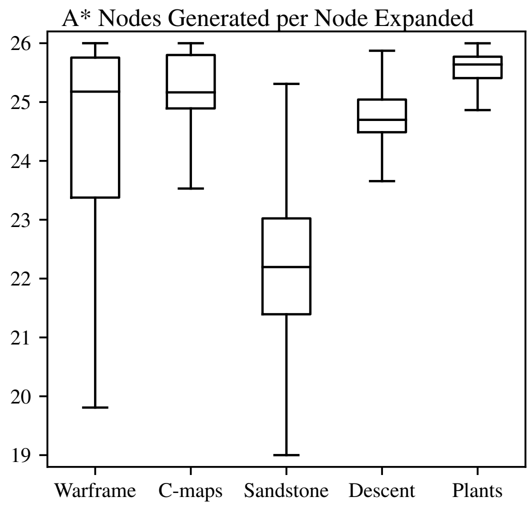


Figure 9: Box plots of nodes generated per expanded by A*. Lower values indicate greater fill-in behaviour in the frontier.

Acknowledgements

This research was funded by Woodside Energy Ltd. through the Monash University Human-in-the-Loop Analytics Graduate Research Industry Partnership. We thank our Woodside collaborators for useful discussions that led to this research. Daniel Harabor is partially supported by the Australian Research Council under grants DP190100013 and DP200100025, and by a gift from Amazon.

References

Beig, M.; Kapralos, B.; Collins, K.; and Mirza-Babaei, P. 2019. G-SpAR: GPU-Based Voxel Graph Pathfinding for Spatial Audio Rendering in Games and VR. In *2019 IEEE Conference on Games (CoG)*, 1–8.

Belov, G.; Du, W.; Garcia de la Banda, M.; Harabor, D.; Koenig, S.; and Wei, X. 2020. From Multi-Agent Pathfinding to Pipe Routing. *Proceedings of the 13th International Symposium on Combinatorial Search*, 11(1): 11–19.

Brewer, D.; and Sturtevant, N. R. 2018. Benchmarks for Pathfinding in 3D Voxel Space. *Proceedings of the 11th International Symposium on Combinatorial Search (SoCS)*, 9(1): 143–147.

Canny, J.; and Reif, J. 1987. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science (SFCS)*, 49–60.

Demetrescu, C.; Goldberg, A. V.; and Johnson, D. S. 2009. *The shortest path problem: Ninth DIMACS implementation challenge*, volume 74. American Mathematical Soc.

Gillberg, J. 2019. AI for Testing: The Development of Bots that Play 'Battlefield V'. https://www.gdcvault.com/play/1026308/AI-for-Testing-The-Development. Accessed: 2023.

Guinness World Records. 1995. First fully 3-dimensional FPS. https://www.guinnessworldrecords.com/world-records/first-fully-3-dimensional-fps. Accessed: 2022.

Harabor, D. 2015. The Warthog Pathfinding Library. https://bitbucket.org/dharabor/pathfinding/src/master/warthog/. Accessed: 2023.

Harabor, D.; Hechenberger, R.; and Jahn, T. 2022. Benchmarks for pathfinding search: Iron Harvest. *Proceedings of the 15th International Symposium on Combinatorial Search (SoCS)*, 15(1): 218–222.

Kocurek Industries Inc. 2021. Details For Our Sandstone. https://kocurekindustries.com/products-category/sandstone-cores. Accessed: 2022.

Li, F.; Zlatanova, S.; Koopman, M.; Bai, X.; and Diakité, A. 2018. Universal path planning for an indoor drone. *Automation in Construction*, 95: 275–283.

Liu, S.; Watterson, M.; Mohta, K.; Sun, K.; Bhattacharya, S.; Taylor, C. J.; and Kumar, V. 2017. Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments. *IEEE Robotics and Automation Letters*, 2(3): 1688–1695.

Long, D.; and Fox, M. 2006. The International planning competition series and empirical evaluation of AI planning systems. *Proceedings of Workshop on Empirical Methods for the Analysis of Algorithm.*

Min, J.-G.; Ruy, W.-S.; and Park, C. S. 2020. Faster Pipe Auto-Routing Using Improved Jump Point Search. *International Journal of Naval Architecture and Ocean Engineering*, 12: 596–604.

Min, K. 2019. Voxel-Based Navigation in Black Desert for Increased Efficiency and Flexibility. https://www.pearlabyss.com/en-US/Company/About/Lab. Accessed: 2023.

Muratov, T.; and Zagarskikh, A. 2019. Octree-Based Hierarchical 3D Pathfinding Optimization of Three-Dimensional Pathfinding. *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*, 1–6.

Neumann, R. F.; Barsi-Andreeta, M.; Lucas-Oliveira, E.; Barbalho, H.; Trevizan, W. A.; Bonagamba, T. J.; and Steiner, M. B. 2021. High accuracy capillary network representation in digital rock reveals permeability scaling functions. *Scientific Reports*, 11(1).

Nobes, T.; Harabor, D.; Wybrow, M.; and Walsh, S. 2022. The Jump Point Search pathfinding system in 3D. *Proceedings of the 15th International Symposium on Combinatorial Search (SoCS)*, 15(1): 145–152.

Noonchester, A. 2019. 'Marvel's Spider-Man' AI Postmortem. https://gdcvault.com/play/1025828/-Marvel-s-Spider-Man. Accessed: 2023.

Saha, I.; et al. 2022. It Costs to Get Costs! A Heuristic-Based Scalable Goal Assignment Algorithm for Multi-Robot Systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 2–10.

Schwarz, M.; and Seidel, H.-P. 2010. Fast Parallel Surface and Solid Voxelization on GPUs. *ACM Transactions on Graphics*, 29(6).

Shen, B.; Chen, Z.; Cheema, M. A.; Harabor, D. D.; and Stuckey, P. J. 2023. Tracking Progress in Multi-Agent Path Finding. arXiv:2305.08446.

Stuckey, P. J.; Feydy, T.; Schutt, A.; Tack, G.; and Fischer, J. 2014. The MiniZinc Challenge 2008–2013. *AI Magazine*, 35(2): 55–60.

Sturtevant, N.; Traish, J.; Tulip, J.; Uras, T.; Koenig, S.; Strasser, B.; Botea, A.; Harabor, D.; and Rabin, S. 2014. The Grid-Based Path Planning Competition: 2014 Entries and Results. *Proceedings of the 8th International Symposium on Combinatorial Search (SoCS)*, 6(1): 241–250.

Sturtevant, N. R. 2012. Benchmarks for Grid-Based Pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2): 144–148.

Sturtevant, N. R.; Sigurdson, D.; Taylor, B.; and Gibson, T. 2019. Pathfinding and abstraction with dynamic terrain costs. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 15: 80–86.

Tabacaru, A.; Atzmon, D.; and Felner, A. 2022. Meeting at the Border of Two Separate Domains. *Proceedings of the 15th International Symposium on Combinatorial Search*, 15(1): 244–248.

Westerdiep, A. 2022. Online HTML5 voxelizer. https://drububu.com/miscellaneous/voxelizer/. Accessed: 2023.

Zhang, N.; Zhang, M.; and Low, K. H. 2021. 3D Path Planning and Real-Time Collision Resolution of Multirotor Drone Operations in Complex Urban Low-Altitude Airspace. *Transportation Research Part C: Emerging Technologies*, 129: 103–123.

Zhang, S. 2021. *Trajectory Planning Based on Optimized Jump Point Search Results Using Artificial Potential Field in 3-D Environments*. Master's thesis, University of California, Santa Cruz.