Using Databases to Improve the Efficiency of Computations for Sub-Optimal Pathfinding in Video Games

Katrina Ziebarth
Earlham College
Richmond, Indiana
kszieba19@earlham.edu

ABSTRACT

This study will examine the feasibility of more closely integrating databases with video games, with motivations being to take advantage of increased data locality, optimizations already implemented in database management systems, and preexisting solutions to a number of problems encountered by modern video games. Using O'Grady's "Bringing Database Management Systems and Video Game Engines Together" as a base, this study will investigate how his approach can be both extended to the use of Weighted A* for sub-optimal pathfinding and applied to pathfinding problems derived from commercial games.

ACM Reference Format:
Katrina Ziebarth. 2022. Using Databases to Improve the Efficiency of Computations for Sub-Optimal Pathfinding in Video Games. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The use of databases to improve the efficiency of pathfinding in video games has implications for the potential complexity of video games. It offers a way to reduce the cost of pathfinding, freeing resources for the other components of a video game.

Perhaps the most relevant work within this field is O'Grady's 2021 dissertation. The need for this research can be seen in O'Grady's observation that "it is not uncommon for business logic of any kind (including video games') to read all data from the database, compute a subset of the data or transformation thereof in an imperative programming language, and then write it back periodically." As O'Grady notes, increasing data locality could potentially allow for much greater efficiency. O'Grady also observes that "DBMSs have decades of development and optimisation under their belt" Additionally, O'Grady argues that modern video games, particularly large multiplayer ones that take place in persistent worlds, confront issues already addressed by database management systems, such as "what does it mean when two players want to manipulate the same part of the game world at the same time?", "Can we efficiently find a subset of game elements that are relevant for a singular calculation, such as the objects that are currently visible to one player?", and "When we need to do these operations over many game objects at once, how do we avoid swapping parts of the huge game world in and out of the main memory?" [11].

O'Grady attempts to demonstrate the feasibility of more closely incorporating relational database management systems by implementing standard components of game engines, including pathfinding, in SQL. His 2021 work includes implementations in SQL of A*, a booking system for A*, and an iterative pathfinding method that O'Grady links to what he describes as "the set-based philosophy behind SQL, which excels at specifying operations on many elements at once." O'Grady evaluates the performance of the first and third of these implementations using pathfinding which uses pgRouting, an extension for PostgreSQL, as a baseline. He specifically notes that his thesis "refrains from introducing a *Domain Specific Language (DSL)* to achieve the outlined tasks and uses only plain SQL without utilizing any imperative extensions" [11].

This work departs from O'Grady's in its examination of sub-optimal pathfinding. The reason for this is the lack of necessity of optimal paths in video games under most circumstances. As Botea et al. note, "In games, the optimality of solutions is not seen as a must-have feature. Suboptimal paths that look reasonable to a user are acceptable" [4]. Similarly, Gao et al. write "finding an optimal path is usually too costly for real-world applications, an alternate is to find some acceptable path by sacrificing some quality of the result. Such ideas are widely used in video games and serving robots" [6]. The admission of sub-optimal pathfinding algorithms allows for a wider range of approaches, potentially leading to increased efficiency in the production of reasonable solutions.

Another choice made by this work is to use various pathfinding problems derived from commercial games for evaluation of performance, where O'Grady used a single unidentified "map of a skirmish" [11]. This decision offers the potential to provide further evidence for the feasibility of closer incorporation of database management systems in commercial game development.

2 RELATED WORK

2.1 Utilization of Databases to Improve Efficiency

Previous work exists concerning the utilization of databases to improve the efficiency of other systems. Bendre et al. explore how a spreadsheet can be unified with a relational database while preserving advantages of both [3].

Similarly, in his dissertation, O'Grady explores the implementation in SQL of not only pathfinding, but also of AI and map generation [11]. O'Grady's 2019 work also investigated the implementation of map generation in SQL [10].

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 Pathfinding in Video Games

A vast amount of relevant work exists in the field of pathfinding in video games. For this reason, this literature review will focus mainly on surveys and overviews of pathfinding in video games, going into greater depth only where necessary.

2.2.1 The Role of A.* Kapi identifies A*, an optimal algorithm, as the most prominent algorithm in video game pathfinding [7]. Similarly, Abd Algfoor notes that A* is among the most well-known search algorithms in games and robotics and that it "inspired many modified and improved algorithms" [2]. For these reasons, it is unsurprising that O'Grady's 2021 investigation of pathfinding focuses on A* and variants based on it [11].

2.2.2 Optimization. Kapi offers an overview of various ways of optimizing pathfinding in video games, including choice of graph representation, modification of heuristic functions, and choice of data structure for implementation [7].

2.2.3 Sub-Optimal Pathfinding. In regards to sub-optimal pathfinding, Botea et al. identify as promising the idea of "combining compressed path databases with hierarchical abstraction" [4].

Returning to the subject of A*, Rabin and Sturtevant, in their list of ways to optimize A*, suggest using an inadmissible heuristic. It is their observation that "a small amount of overestimating has large benefits with very little noticeable nonoptimality" [12].

2.2.4 Heuristics. The three heuristics which O'Grady discusses for use with A* on grid-based maps are Manhattan distance for Von Neumann neighborhoods and Euclidean distance and Chebyshev distance for Moore neighborhoods [11].

Botea et al. note that Manhattan distance and Octile distance are "simple, fast to compute, and reasonably accurate on many map topologies" [4].

2.2.5 Environments. O'Grady's 2019 work uses the OpenRA engine [10], as does his 2021 dissertation [11]. In explaining the reasons for his choice of it for his dissertation, O'Grady gives as advantages that it is open source and under GNU General Public License, has a development community which can be contacted for inquiries, has all core components written in an object-oriented programming language, and has a "clear-cut set of mechanisms" for managing several actors due to its focus on real time strategy games [11].

Video games utilize a range of techniques for terrain discretization, with Nash and Koenig giving examples of games using regular grids, navigation meshes, and circle-based waypoint graphs [9].

Unfortunately, it was not possible to determine whether O'Grady's experiments to measure performance in OpenRA in his 2021 dissertation used Von Neumann neighborhoods or Moore neighborhoods. O'Grady discusses both types of neighborhoods and uses in his code the boolean pseudo-function neighboring(a, b) [11]. It seems probable that his goal in doing so was to make the code more general and avoid the need to present separate implementations for the two types of neighborhoods. However, O'Grady does state that "we assume maps to generally be decomposable into grids of arbitrary granularity" [11], confirming that these experiments dealt only with 2D maps.

2.2.6 Existing Benchmarks. A number of pathfinding benchmarks have been drawn from commercial games which utilize 2D grids [14]. For 3D voxel grids, a benchmark set from the game Warframe has been made available, as Brewer and Sturtevant explain. They note that "in comparison with 2D grid maps, relatively little work has been done on planning directly in a 3D space representation" [5]. For this reason, this project will focus on pathfinding on 2D grids.

2.3 Weighted A*

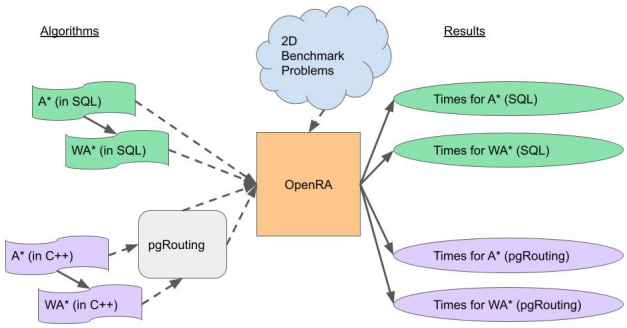
Rivera et al. note that Weighted A* (WA*) uses an evaluation function which is similar to A*'s, except for its incorporation of a weight greater than or equal to one by which the heuristic function is multiplied. They also refer to weighting the heuristic as a "simple but powerful technique" [13]. This relates also to Rabin and Sturtevant's suggestion that A* be optimized for video games through use of an inadmissible heuristic. In fact, their approach of multiplying the heuristic portion of the formula for A*'s evaluation by a weight is in practice identical to WA* [12]. It is proposed by Rivera et al. that "A possible reason that explains why Weighted A* finds solutions more quickly than regular A* is that in multiplying the heuristic by a factor $w \geq 1$, the heuristic becomes more *accurate*, in a significant portion of the search space" [13].

2.4 pgRouting

As O'Grady notes, pgRouting is an extension which can be used with PostgreSQL "to offer path finding capabilities through a variety of path finding algorithms, including A*, through calls to UDFs". In his 2021 thesis, O'Grady uses pgRouting as a baseline for his work on pathfinding within a database management system (DBMS). While he did not find it compatible with "the ambition to fully realize components of video games in SQL in order to not be tied down to a specific DBMS," due to its implementation in C++, he did mention that "it offers a way of keeping the computation of paths entirely in the world of the DBMS." [11]. pgRouting is also open source [1], which will allow for using a modified version of its implementation of A* to implement Weighted A* as well.

3 DESIGN AND IMPLEMENTATION

In accordance with the code given by O'Grady in his section on spatial A* [11], A* will be implemented in SQL. A modified version of the code will then be used in order to implement Weighted A* in SQL. This decision was made due to the dominance of A* in video game pathfinding, even in regards to optimization strategies. Similar to O'Grady's 2021 work, pgRouting's A* implementation will also be used as a baseline, with Weighted A* also implemented in pgRouting in order to allow for fairer comparison.



The performances of A* and Weighted A*, and pgRouting's implementations of A* and Weighted A* will then be compared on one or more set of 2D pathfinding problems within OpenRA. These problems will be drawn from Sturtevant's collection of existing benchmark sets derived from commercial games [14], with ones from real-time strategy games chosen preferably, due to the focus of OpenRA on real-time strategy. Several weights will be used in all experiments involving Weighted A*, in recognition of Rabin and Sturtevant's observation that "The correct weight for your game or parts of your game must be discovered experimentally" [12], which implies that each set of pathfinding problems will require a different weight to optimize performance.

O'Grady used one action per 150 milliseconds as an loose upper bound for "agreeable waiting time" for pathfinding, citing Lewis et al.'s Starcraft study as a source [11]. Lewis et al. assert that professional Starcraft players in South Korea "can execute over 400 actions per minute (APM) in the game. By contrast, a highly accomplished amateur in the United States would likely top out in the mid 200s." Lewis' study, which utilizes replay files from games played in tournaments outside South Korea, uses a bound of 250 actions per minute [8]. This suggests that outside of high-performance contexts, 250 APM would likely yield a more reasonable upper bound than the 400 APM O'Grady used. For that reason, this study will use metrics derived from both in evaluating algorithm performance.

All four algorithm implementations will be compared regarding time needed to calculate paths relative to path length as measured in nodes, similar to the comparison made by O'Grady between the pgRouting implementation of A* and his implementation of A* in SQL. In particular, emphasis will be placed on under what circumstances they remain within one or both of the upper bounds for waiting time. In recognition of O'Grady's observation that "In real-world scenarios, especially for real time strategy games, short path searches are the most time-critical ones, as they are part of the micromanagement of units that occurs during heated battles"[11], heavier weight will be given to the results on shorter paths.

The relationship between optimality of path and path length in nodes will also be examined for Weighted A*, since short path searches' place as "part of the micromanagement of units that occurs during heated battles" [11] suggests they may be not only time-critical, but important to achieve near-optimality for.

4 MAJOR RISKS

Understanding the structure of OpenRA, particularly in regard to how pathfinding is implemented within it, may be difficult, and it may prove necessary to contact its development community with inquiries. It is also currently uncertain whether benchmark problems will be in a format easily compatible with OpenRA's architecture and O'Grady's implementation of A* in SQL.

Other risks relate to possible difficulties in drawing conclusions based on results. If the performance of the implementation of Weighted A* in SQL is inferior to that of A* in SQL, it may be difficult to draw definite conclusions. Another possible difficulty may arise from the fact that while SQL is considered a declarative programming language, O'Grady's implementation of A* in it is imperative, which may lead to decreased efficiency. Lastly, O'Grady's experiments showed that the performance of pgRouting's implementation of A* was superior to that of his SQL implementation of A* in multiple ways, which may make it difficult to draw relevant conclusions regarding the performance of an implementation of Weighted A* in SQL.

5 TIMELINE

Week 1

Research algorithm implementation in SQL and type up O'Grady's implementation of A* in SQL. Identify specific pathfinding problems. Begin constructing implementation of Weighted A* in SQL, using O'Grady's implementation of A* as a base. Begin constructing implementation of Weighted A* for pgRouting using existing implementation of A* as a base.

Week 2

Finish constructing implementation of Weighted A* in SQL, using O'Grady's implementation of A* as a base. Finish constructing implementation of Weighted A* for pgRouting using existing implementation of A* as a base.Construct data architecture diagram. Begin integrating algorithm implementations with OpenRA.

Week 3

Review draft of paper for inconsistencies with implementation plan. Continuing integrating algorithm implementations with OpenRA.

Week 4

First draft of paper due (introduction, literature review, v1 of data architecture diagram, methods (data set(s), analysis plan)). Finish integrating algorithm implementations with OpenRA.

Week 5

First draft of software due. Integrate benchmark problem format with algorithm implementation and OpenRA. Complete experiment setup.

Week 6

Run initial experiments. Revise paper to resolve inconsistencies with implementation. Begin writing up initial results.

Week 7

Finish writing up initial results. Begin creating visualizations for initial results.

Week 8

Finish creating visualizations for initial results. Resolve any formatting errors in paper. Begin planning demonstration video.

Week 9

Second draft of paper due (+initial results, +initial visualizations). Finish planning demonstration video. Begin planning poster.

Week 10

First draft of demonstration video due. Finish planning poster. Create poster. Work on analysis for paper.

Week 11

First draft of poster due. Complete analysis for paper. Check paper for errors and typos.

Week 12

Third draft of paper due. Revise demonstration video. Create new draft of demonstration video. Plan revision of poster.

Week 13

Second draft of demonstration video due. Revise poster.

Week 14

Second draft of poster due. Check paper, poster, and demonstration video for errors, inconsistencies, and typos.

Finals Week

Final versions of paper, poster, and demonstration video due.

ACKNOWLEDGMENTS

I would like to thank David Barbella and Sofia Lemons for their assistance in constructing this proposal.

REFERENCES

[1] 2020. *pgRouting Project.* Retrieved December 9, 2022 from https://pgrouting.org/#
[2] Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. 2015. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology* 2015 (2015).
[3] Mangesh Bendre, Bofan Sun, Ding Zhang, Xinyan Zhou, Kevin ChenChuan Chang, and Aditya Parameswaran. 2015. Dataspread: Unifying databases and spreadsheets. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 2000.
[4] Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, and Dana Nau. 2013. Pathfinding in games. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[5] Daniel Brewer and Nathan R. Sturtevant. 2018. Benchmarks for Pathfinding in 3D Voxel Space. *Symposium on Combinatorial Search (SoCS)* (2018).
[6] Zhipeng Gao, Junmeng Huang, and Chen Zhao. 2021. A double-phase search algorithm for sub-optimal path finding. In *2021 the 5th International Conference on Innovation in Artificial Intelligence*. 190–195.
[7] Azyan Yusra Kapi, Mohd Shahrizal Sunar, and Muhamad Najib Zamri. 2020. A review on informed search algorithms for video games pathfinding. *International Journal* 9, 3 (2020).
[8] Joshua Lewis, Patrick Trinh, and David Kirsh. 2011. A corpus analysis of strategy video game play in starcraft: Brood war. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 33.
[9] Alex Nash and Sven Koenig. 2013. Any-angle path planning. *AI Magazine* 34, 4 (2013), 85–107.
[10] Daniel O'Grady. 2019. Database-Supported Video Game Engines: Data-Driven Map Generation. *BTW 2019* (2019).
[11] Daniel O'Grady et al. 2021. *Bringing Database Management Systems and Video Game Engines Together.* Ph. D. Dissertation. Eberhard Karls Universität Tübingen.
[12] Steve Rabin and Nathan R Sturtevant. 2013. Pathfinding architecture optimizations. *Game AI Pro: Collected Wisdom of Game AI Professionals* 1 (2013), 241–252.
[13] Nicolás Rivera, Jorge A Baier, and Carlos Hernández. 2013. Weighted real-time heuristic search. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. 579–586.
[14] Nathan R. Sturtevant. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games* 4, 2 (2012), 144 – 148. http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf