Diverse Shortest Paths in Game Maps: A Comparative User Study and Experiments

Lingxiao Li[‡], Muhammad Aamir Cheema[‡], Mohammed Eunus Ali[§], Hua Lu[†], Huan Li[*]

[‡]Faculty of Information Technology, Monash University, Australia,
[§]Bangladesh University of Engineering and Technology, Bangladesh
[†]Department of People and Technology, Roskilde University, Denmark
[*]Department of Computer Science, Aalborg University, Denmark
[‡]{lingxiao.li, aamir.cheema}@monash.edu, [§]eunus@cse.buet.ac.bd,
[†]luhua@ruc.dk, [*]lihuan@cs.aau.dk

Abstract. Computing diverse shortest paths requires finding a set of k alternative paths (including the shortest path) between a given source s and a target t. Intuitively, these paths should be significantly different from each other and meaningful/natural (e.g., must not contain loops or unnecessary detours). While finding diverse shortest paths (also called alternative paths) in road networks has been extensively studied, to the best of our knowledge, we are the first to formally study alternative pathfinding in game maps which are typically represented as Euclidean planes containing polygonal obstacles. First, we adapt the existing techniques designed for road networks to find alternative paths in the game maps. Then, we design a web-based system that allows the users to visualise the alternative paths generated by these existing approaches in different maps. Finally, we use this web-based system to conduct a user study that shows that the existing road network approaches generate high-quality alternative paths when adapted for the game maps. Furthermore, we also evaluate the quality of alternative paths returned by existing approaches using some well-known quantitative measures on a widely used game maps benchmark.

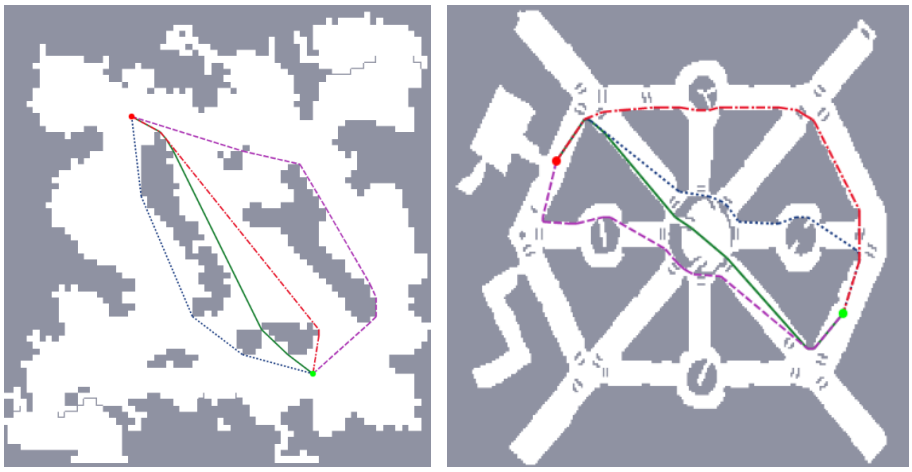
Keywords: Diverse shortest paths · Alternative pathfinding · Game maps

1 Introduction

Given a source s and a target t, a shortest path query requires finding the path from s to t with the minimal total cost. Finding shortest paths is a very well-studied problem and numerous techniques exist to find shortest paths in different settings such as in road networks [1], general graphs (i.e., social networks) [14], indoor venues [3], game maps [7] etc. In many applications, it is desirable to return not only the shortest path to the users/agents but also some alternative paths so that they can choose a path of their choice. Intuitively, these alternative paths must be short and sufficiently different from each other (i.e.,

diverse). Modern navigation systems such as Google Maps return several paths from source to target and the user can choose a path of their choice to travel on.

Inspired by the applications, computing alternative paths in road networks has received significant research attention, e.g., see [13,8,16] and referenes therein. However, to the best of our knowledge, computing alternative paths in game maps has received no research attention despite its applications. Typically game characters take shortest paths to reach the target location. However, in many cases, it is desirable to have more than one paths for the characters to choose from. For example, in real-time strategy (RTS) games, if the opponent character always takes the shortest path to the target, their movement/plan may become predictable. Therefore, it may be better to compute alternative paths and randomly assign one of the alternative paths to the character. Many RTS games allow the users to choose a waypoint to allow them choosing a path different from the shortest path. In such games, the users may be shown several alternative paths and asked to select a path of their choice. Computing alternative paths also has applications in indoor venues which are also typically modeled as Euclidean plane containing obstacles. For example, an indoor navigation system may show multiple alternative paths to the target location so that the user can choose a path of their choice. Figure 1 shows two examples where four alternative paths are reported on two different game maps.



(a) Warcraft III (b) Baldurs Gate II

Fig. 1. Four alternative paths on two different game maps.

Although some open-source game development projects[1] have included support for computing alternative paths in game maps, the efficiency and effective-

[1] For example, see a game development project with Unity3D at `https://arongranberg.com/astar/docs/alternativepath.html`

ness of the proposed alternative pathfinding algorithms are not clear. In this paper, we fill this gap and formally study alternative pathfinding in game maps. We make the following contributions.

To the best of our knowledge, we are the first to study alternative pathfinding in game maps. First, we adapt some of the most well-known techniques for computing alternative paths on road networks (namely Penalty [2,4,11], Plateaus [9] and Dissimilarity [6,15]) for the game maps. However, it is not clear whether these techniques generate alternative paths of good quality when extended for the game maps. To this end, we created a web-based demonstration system and conducted a user-study on 9 diverse game maps selected from a widely used game maps benchmark. In total, we received 472 responses and the user study shows that the three approaches generate high-quality alternative paths in game maps as perceived by the users. We have made the source code[2] of the web-based demonstration system publicly available so that it can be reused and/or extended as needed. Furthermore, we also evaluate the quality of alternative paths returned by these algorithms on a widely used game maps benchmark using some well-known quantitative measures such as *path similarity*, *bounded stretch* and *local optimality* [10]. Our experimental study shows the paths returned by the existing approaches are comparable to each other in terms of quality metrics. However, the results show that the computation cost of these approaches is up to two orders of magnitude higher than the computation cost of the state-of-the-art shortest path algorithm, Polyanya [7], which computes only the shortest path, i.e., the overhead to compute alternative path is too high. Therefore, as a future work, there is a need for developing efficient techniques to compute diverse shortest paths in game maps.

2 Preliminaries

2.1 Problem Formulation

We assume a 2D Euclidean plane containing a set of polygonal obstacles. A **convex vertex** is a vertex that is located at a convex corner of the polygon. A **non-convex vertex** is located at a concave corner. We use V to denote the set of all convex vertices in the plane. Two points are **visible** from each other (also called **co-visible**) if there is a straight line between the two points not passing through any obstacle. A **path** P between a source s and a target t is an ordered set of points $\langle p_1, p_2, \cdots, p_n \rangle$ such that, for each p_i $(i < n)$, p_i and p_{i+1} are co-visible where $p_1 = s$ and $p_n = t$. **Length** of a path P is the cumulative Euclidean distance between every successive pair of points, denoted as $|P|$, i.e., $|P| = \sum_{i=1}^{k-1} EDist(p_i, p_{i+1})$ where $EDist(x, y)$ is the Euclidean distance between x and y. The shortest path $sp(s, t)$ is a path between s and t with the minimum length. The shortest distance between s and t is denoted as $d(s, t)$, i.e., $d(s, t) = |sp(s, t)|$.

[2] `https://bitbucket.org/lingxiao29/customized/src/master/`

Given a positive integer k, we are interested in finding k **alternative paths** (including the shortest path $sp(s,t)$) between s and t such that each alternative path is no longer than $d(s,t) \times \epsilon$ where $\epsilon \geq 1$ is a user-defined parameter. Intuitively, the k alternative paths must be diverse (e.g., should have small overlap with each other) and each path must be a "reasonable" path, e.g., should not contain unnecessary detours and loops etc. Diversity can be quantified by defining a similarity function based on the overlap between paths and requiring the paths to have similarity with each other less than a given threshold. The previous works in road networks (e.g., see [10]) have defined several measures to quantify whether a set of alternative paths is "reasonable" or not. We formally define these measures in the Experiments section.

2.2 Related Work

While finding shortest paths in game maps has been extensively studied (e.g., see [7,18] and references therein), alternative pathfinding has only been studied in road networks. Below, we briefly describe three of the most popular approaches to compute alternative paths in road networks (we assume undirected road networks for simplicity).

Penalty: This approach [2,4,11] iteratively computes the shortest paths from s to t and, after each iteration, it applies a penalty to each edge on the shortest path found in the previous iteration by increasing its edge weight by a certain penalty factor (e.g., by multiplying the current edge weight with 1.5). Since the edge weights on the shortest path are increased, the approach is likely to choose a significantly different shortest path in the next iteration. The algorithm stops when k unique paths are found by the algorithm or when the length of the shortest path found in this iteration is longer than $d(s,t) \times \epsilon$.

Plateaus: This approach [9] was designed by Cotares Limited for their routing engine Choice Routing. First, two shortest path trees T_s and T_t are computed rooted at s and t, respectively. Then, the two tree are *joined* and common branches in the two trees are found. Each common branch is called a plateau. More formally, given a branch $\langle s, \cdots, u_1, u_2, \cdots, u_n, \cdots, y\rangle$ in T_s and a branch $\langle t, \cdots u_n, u_{n-1}, \cdots, u_1, \cdots, x\rangle$ in T_t, the common part of the two branches $\langle u_1, \cdots, u_n\rangle$ is a plateau, denoted as $pl(u_1, u_n)$. Note that the shortest path between s and t is a plateau with length $d(s,t)$. Let $pl(u,v)$ be a plateau such that u is the end closer to s and v is the end closer to t. The plateau is used to obtain an alternative path $sp(s,u) \oplus pl(u,v) \oplus sp(v,t)$ where $\oplus$ is the concatenation operation. It was observed in [9] that longer plateaus typically generate better alternative paths. Thus, the algorithm selects k longest plateaus and generates alternative paths for each of the plateaus. We show an example of how Plateaus computes alternative paths in game maps in the next subsection.

Dissimilarity: Techniques in this category specifically define a function that computes dissimilarity between two paths. Given a dissimilarity threshold θ, the goal is to return the shortest alternative paths such that the dissimilarity between any pair of returned paths is at least θ. It was shown that this problem is NP-hard [6]. Therefore, several approximate algorithms [6,15] have been proposed

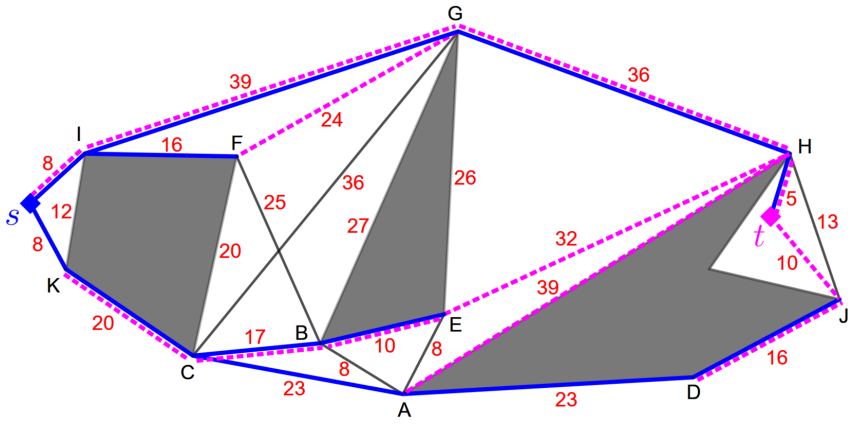


Fig. 2. Three alternative paths generated by Plateaus are $\langle s, I, G, H, t\rangle$, $\langle s, K, C, B, E, H, t\rangle$ and $\langle s, K, C, A,\ D, J, t\rangle$ with lengths 88, 92 and 100, respectively.

in the past. Below, we describe an algorithm [5] which was shown in [13] to generate high-quality alternative paths in road networks.

For a vertex v on the road network, a via-path passing through v is $sp(s, v) \oplus sp(v, t)$. Similar to the Plateaus approach, two shortest path trees T_s and T_t are computed and $d(s, v)$ and $d(v, t)$ are stored for each vertex v on the trees. Given these trees, the via-paths and their lengths can be efficiently computed. The algorithm iteratively accesses vertices on the road network in ascending order of the lengths of their via-paths. An accessed via-path is added to the set of alternative paths if its dissimilarity to the already added alternative paths is at least θ. The algorithm stops when k alternative paths are found or when the current via-path is longer than $d(s, t) \times \epsilon$.

3 Adapting Existing Techniques for Game Maps

We adapt Penalty, Dissimilarity and Plateaus for the game maps as follows. We create a visibility graph $G = \{V, E\}$ where V is the set of convex vertices in the game map and E is the set of edges connecting each pair of co-visible vertices (u, v) with edge weight corresponding to the Euclidean distance between them $EDist(u, v)$. For a given query, s and t are added to G by adding edges between s (resp. t) and the vertices visible from s (resp. t). Consider the example of Figure 2 containing three grey polygonal obstacles. The visibility graph G consists of all the edges shown in the figure. Once G is constructed, each of the approach described earlier can then be applied on G to generate the alternative paths. We prune every *non-taut* path explored by these approaches. A taut path is a path which, when treated as a string, cannot be made "tighter" by pulling on its ends [17], e.g., the path $\langle s, I, F, G\rangle$ is non-taut because string-pulling results

in a shorter path $\langle s, I, G\rangle$. We use Dijkstra's algorithm to compute the shortest path trees.

Example 1. Figure 2 shows an example of the paths generated by Plateaus. The convex vertices are A to K. The source and target are connected to their respective visible convex vertices (I and K for s and H and J for t). Plateaus then computes two shortest path trees T_s rooted at s (see the tree shown in blue edges) and T_t rooted at t (see the pink edges shown in broken lines). The common branches in the two trees are the plateaus. The three longest plateaus are $\langle s, I, G, H, t\rangle$, $\langle K, C, B, E\rangle$ and $\langle D, J\rangle$ with lengths 88, 47 and 16, respectively. Thus, three alternative paths are generated connecting s and t to the end of each plateau closer to them. The three alternative paths are $\langle s, I, G, H, t\rangle$, $\langle s, K, C, B, E, H, t\rangle$ and $\langle s, K, C, A, D, J, t\rangle$ with lengths 88, 92 and 100, respectively.

4 User Study

A recent work [13] presented a user study conducted on Melbourne, Dhaka and Copenhagen road networks which shows that Penalty, Dissimilarity and Plateaus generate alternative paths comparable to those generated by Google Maps. To compute alternative paths in game maps, we adapt these existing techniques originally designed for road networks as described in the previous section. However, the first question we ask ourselves is whether these techniques are able to generate high-quality alternative paths in game maps? To answer this, we conducted a user study and present the details in this section.

Based on our previous work [12], we developed a web-based system containing 9 diverse game maps selected from a widely used benchmark[3]. We created a webpage[4] containing the instructions for the participants and the links to the web-based system. The participants were asked to complete two types of surveys:

- **Pre-selected:** They were shown 3 pairs of pre-selected source-target pairs for each of the 9 game maps.
- **User-selected:** For each of the 9 maps, the participants could select any source-target pair of their choice by clicking on the map (they could select as many source-target pairs as they wanted but at least one for each map).

Once the source and target are selected, the system generates up to 4 alternative paths by each approach which can be viewed by clicking the radio button next to the approach's name. In the system, Plateaus, Dissmilarity and Penalty are named A, B and C, respectively, to avoid any potential preconceived biases. The participants were asked to enter a rating for each approach from 1-5 (higher the better). The participants were given a brief background on the alternative paths in road networks and game maps along with some applications. They were

[3] `https://movingai.com/benchmarks/grids.html`
[4] `http://aamircheema.com/paths_games/`

asked to rate the alternative paths generated by each approach based on how good they thought the paths generated by the approach were considering that the alternative routes should be significantly different from each other but meaningful/natural. People are used to see alternative paths in road networks (due to popular navigation applications), but they most likely have not seen alternative paths in game maps. Therefore, we selected the participants who were familiar with computer games or had research background either in pathfinding on game maps or alternative pathfinding in road networks.

	#Responses	Average Rating		
		Plateaus	Dissimilarity	Penalty
All	472	**4.028**	3.998	3.852
Pre-selected	243	4.016	**4.025**	3.938
User-selected	229	**4.039**	3.969	3.760

Table 1. Results of the user study: Average rating, average and maximum $Sim(T)$, and average path length for Plateaus, Dissimilarity (shown as Dissim.) and Penalty. Best values for each category are shown in bold.

Table 1 shows the results of the user study. We received 472 responses from 9 participants. The results indicate that, on average, the three approaches received overall quite similar ratings and the participants rated the alternative paths quite highly (rated at around 4 on a scale of 1-5). We also conducted one-way repeated measures ANOVA test. Given a null hypothesis of no statistically significant difference in mean ratings of the three approaches, the results suggest that, at $p < 0.05$ level, there is no evidence that the null hypothesis is false, i.e., there is no credible evidence that the three approaches received different ratings on average. Next, we present a detailed experimental study comparing these approaches using some well-known quantitative measures.

5 Experiments

5.1 Settings

Similar to the existing studies on shortest pathfinding in game maps [7,18], we conduct experiments on the widely used grid map benchmarks[5], described in [19]. On a total of 298 game maps, 67 maps from Dragon Age II (DA), 156 maps from Dragon Age Origins (DAO) and 75 maps from Baldur's Gate II (BG) (see Table 2).

We compare Plateaus, Dissimilarity and Penalty shown as Pla, Dissim, and Pen in the results, respectively. For Penalty, the penalty factor was set to 1.4 and the dissimilarity threshold θ for Dissimilarity was set to 0.6 (we tried various penalty factors and dissimilarity thresholds and chose the best values). ϵ was set to 1.5 for each approach.

[5] `https://github.com/nathansttt/hog2`

Game	Benchmark Stats			
	#Maps	#Queries	#Vertices	#Convex Vertices
Dragon Age 2 (DA)	67	68K	1183	611
Dragon Age: Origins (DAO)	156	159K	1728	927
Baldurs Gate (BG)	75	93K	1295	668

Table 2. Benchmark stats include total number of maps and total number of queries in each benchmark, and average number of vertices and convex vertices per map.

We also include Polyanya [7] (shown as Poly) which is the state-of-the-art online algorithm for finding shortest paths in game maps. Although Polyanya only finds the shortest path, we compare against it to show the additional costs the alternative pathfinding algorithms pay to generate the alternative paths. We use the implementation of Polyanya provided by its authors[6]. For all algorithms except Polyanya, we vary k from 1 to 5 and the default value of k is 3.

5.2 Evaluation Measures

We evaluate the algorithms considering query processing time and quality of the alternative paths returned. To evaluate the quality of a set of alternative paths $\mathcal{P}$, we use bounded stretch, local optimality and similarity defined below.

Let $P = \langle p_1, p_2, \cdots, p_n \rangle$ be an alternative path between s and t such that $p_1 = s$, $p_n = t$, each p_i $(1 < i < n)$ is a vertex of an obstacle and for each p_i $(i < n)$, p_i and p_{i+1} are visible from each other. We use $P_{x,y}$ where $x < y$ to denote the subpath $\langle p_x, \cdots, p_y \rangle$ of P and denote its length as $d^P(p_x, p_y)$, i.e., $d^P(p_x, p_y) = \sum_{i=x}^{y-1} EDist(p_i, p_{i+1})$. Hereafter, whenever we use x and y, assume $x < y$.

Bounded Stretch [10]. Stretch of a path defines how long is the path compared to the shortest path. Formally, stretch of a subpath $P_{x,y}$ is defined as $S(P_{x,y}) = d^P(p_x, p_y)/d(p_x, p_y)$. For an alternative path P, its bounded stretch is the maximum stretch of any of its subpaths.

Given an alternative path P, we define its bounded stretch as follows.

$$BS(P) = \max_{\forall (x,y)} \frac{d^P(p_x, p_y)}{d(p_x, p_y)} \tag{1}$$

Consider a path P which has a bounded stretch 1.20, i.e., the maximum stretch of any of its subpath is 1.20. This means that there is no subpath of P which is more than 20% longer than the shortest distance between its end points. Note that an alternative path P with smaller bounded stretch is better. Also, if P is a shortest path, its bounded stretch is 1. Let $\mathcal{P}$ be a set of alternative paths returned by an algorithm. The bounded stretch of $\mathcal{P}$ is the maximum bounded stretch of any of the paths in $\mathcal{P}$, i.e., $BS(\mathcal{P}) = max_{\forall P \in \mathcal{P}} BS(P)$.

Local Optimality [10]. We say that a subpath $P_{x,y}$ is suboptimal if it is longer than the shortest distance between p_x and p_y, i.e., $d^P(p_x, p_y) > d(p_x, p_y)$. Given

[6] `https://bitbucket.org/mlcui1/polyanya`

an alternative path P between s and t, we use $minL(P)$ to denote the length of the shortest suboptimal subpath of P (if all subpaths are optimal, $minL(P)$ is assumed to be $d(s,t)$). Note that any subpath of P which is shorter than $minL(P)$ must be optimal. Thus, $minL(P)$ is a measure of optimality. The local optimality $LO(P)$ normalises this measure w.r.t. the shortest distance $d(s,t)$ between s and t.

Given an alternative path P, we define its local optimality as follows.

$$LO(P) = \frac{minL(P)}{d(s,t)} = \min_{\forall (x,y): d^P(p_x,p_y) > d(p_x,p_y)} \frac{d^P(p_x,p_y)}{d(s,t)} \tag{2}$$

Consider an alternative path P between s and t and assume that its shortest suboptimal path has length 20 and $d(s,t) = 100$. Its local suboptimality is $20/100 = 0.2$. This implies that every subpath of P which is shorter than 20% of the shortest path between s and t is guaranteed to be an optimal path. A path P with higher local optimality is better. Also, if P is a shortest path, its local optimality is 1. Let $\mathcal{P}$ be a set of alternative paths returned by an algorithm. The local optimality of $\mathcal{P}$ is $LO(\mathcal{P}) = min_{\forall P \in \mathcal{P}} LO(P)$.

Similarity [13]. Similarity $Sim(\mathcal{P})$ of a set of alternative paths $\mathcal{P}$ is

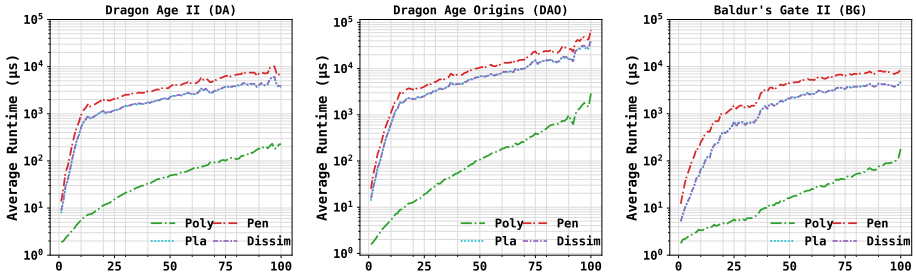
$$Sim(\mathcal{P}) = \max_{\forall (P_i,P_j) \in \mathcal{P} \times \mathcal{P}: i \neq j} \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \tag{3}$$

where $|P_i \cap P_j|$ (resp. $|P_i \cup P_j|$) denotes the total length of the overlap (resp. union) of two paths P_i and P_j.

In our experiments, we compute $BS(\mathcal{P})$, $LO(\mathcal{P})$ and $Sim(\mathcal{P})$ for each query and report average values. We also report the maximum of $BS(\mathcal{P})$ and $Sim(\mathcal{P})$ across all queries which correspond to the worst-case bounded stretch and similarity for an algorithm across all queries. We also report minimum of $LO(\mathcal{P})$ across all queries representing the worst-case for local optimality. If an approach is only able to generate less than k alternative paths, it may get better quantitative scores which is unfair (e.g., if an approach only returns the shortest path, it will receive the best possible score). Therefore, we only consider the queries where each approach generates exactly k alternative paths.

5.3 Results

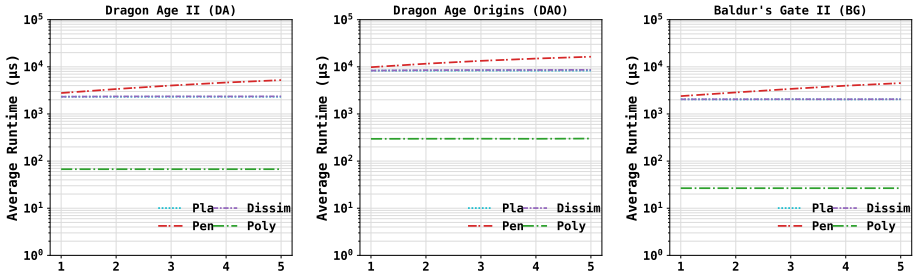
Query runtimes: Figure 3 shows query runtimes for different algorithms on DA, DAO and BG maps. Similar to the existing works on shortest pathfinding [7,18], we sort the queries by the number of nodes expansion required by the standard A* search to solve them (which is a proxy for how challenging a query is) and the x-axis corresponds to the percentile ranks of queries in this order. Figure 3 shows that Plateaus and Dissimilarity have almost the same query times as they both need to generate two shortest path trees which is the dominant cost. The penalty is more expensive as it needs to iteratively compute the shortest paths until k unique paths have been found. Polyanya is about two orders of magnitude faster than all three alternative pathfinding algorithms.



(a) DA (b) DAO (c) BG

Fig. 3. x-axis shows the percentile ranks of queries in number of node expansions needed by A* search to solve them.

Varying K: Figures 4(a), 4(b) and 4(c) show the result for varying k for the DA, DAO and BG maps, respectively. The cost of Plateaus and Dissimilarity do not change with k because they compute forward and backward shortest path trees regardless of the value of k which is the dominant cost. The cost of Penalty increases with increasing k because the algorithm needs to compute the results in at least k iterations. Polyanya was run only for $k = 1$ as it only generates the shortest path.



(a) DA (b) DAO (c) BG

Fig. 4. Effect of varying k.

Quality of alternative paths: Table 3 evaluates the quality of alternative paths generated by different algorithms. Regarding query runtimes, Plateaus and Dissimilarity are about two times faster than the Penalty approach. However, the Penalty method generates alternative paths with quality comparable to the Dissimilarity approach and better than Plateaus. The average bounded stretch of the Dissimilarity approach is better than Plateaus and Penalty. In terms of

average similarity, Penalty performs the best. However, Penalty performs quite poorly in terms of the worst-case (i.e., max) similarity (with maximum similarity around 0.9), whereas Dissimilarity guarantees that the maximum similarity is no more than 0.4 (recall dissimilarity threshold $\theta = 0.6$). Dissimilarity is comparable to Penalty, which outperforms the Plateaus in terms of local optimality. Dissimilarity is the best approach in terms of average path length.

Algorithm	$BS(\mathcal{P})$		$Sim(\mathcal{P})$		$LO(\mathcal{P})$		**Length**
	AVG	**MAX**	**AVG**	**MAX**	**AVG**	**MIN**	
DA							
Dissimilarity	**1.086**	**2.087**	0.188	**0.400**	0.352	0.008	**121.5**
Plateau	1.216	3.335	0.232	0.882	0.261	0.008	125.4
Penalty	1.157	7.000	**0.129**	0.953	**0.354**	0.008	124.1
DAO							
Dissimilarity	**1.050**	**3.481**	0.162	**0.400**	**0.336**	0.005	**124.9**
Plateau	1.207	6.859	0.184	0.931	0.190	0.005	132.7
Penalty	1.103	19.00	**0.083**	0.931	0.335	0.005	126.9
BG							
Dissimilarity	1.092	**2.903**	0.136	**0.400**	0.342	0.009	**282.1**
Plateau	1.224	4.454	0.137	0.876	0.252	0.009	298.0
Penalty	**1.088**	5.333	**0.089**	0.831	**0.343**	0.009	286.7

Table 3. Quality of alternative paths on DA, DAO and BG maps. We show $BS(\mathcal{P})$ (smaller the better), $Sim(\mathcal{P})$ (smaller the better), $LO(\mathcal{P})$ (larger the better) and average path length (smaller the better). Best values for each column are shown in bold.

6 Conclusions and Future Work

We are the first to study alternative pathfinding in game maps. We adapt the existing techniques designed for road networks to find alternative paths in game maps. We present a user study conducted using a web-based system demonstrating that the existing approaches are capable of generating high-quality alternative paths in game maps. Furthermore, we also evaluate the quality of alternative paths returned by existing approaches using some well-known quantitative measures on a widely used game maps benchmark. We also compared the query processing costs of these algorithms with the state-of-the-art shortest path algorithm, Polyanya. The results show that Polyanya is almost two orders of magnitude faster than the existing alternative pathfinding techniques in terms of the query processing times. This implies that the overhead paid by these algorithms to compute alternative paths is quite high. Thus, there is a need to develop techniques to efficiently compute diverse shortest paths in game maps.

Acknowledgements. Muhammad Aamir Cheema is supported by ARC FT180100140.

References

1. Abraham, I., Delling, D., Goldberg, A.V., Werneck, R.F.: A hub-based labeling algorithm for shortest paths in road networks. In: International Symposium on Experimental Algorithms. Springer (2011)
2. Akgün, V., Erkut, E., Batta, R.: On finding dissimilar paths. European Journal of Operational Research **121**(2), 232–246 (2000)
3. Cheema, M.A.: Indoor location-based services: challenges and opportunities. SIGSPATIAL Special **10**(2), 10–17 (2018)
4. Chen, Y., Bell, M.G., Bogenberger, K.: Reliable pretrip multipath planning and dynamic adaptation for a centralized road navigation system. IEEE Transactions on Intelligent Transportation Systems (2007)
5. Chondrogiannis, T., Bouros, P., Gamper, J., Leser, U., Blumenthal, D.B.: Finding k-dissimilar paths with minimum collective length. In: SIGSPATIAL (2018)
6. Chondrogiannis, T., Bouros, P., Gamper, J., Leser, U., Blumenthal, D.B.: Finding k-shortest paths with limited overlap. The VLDB Journal pp. 1–25 (2020)
7. Cui, M., Harabor, D.D., Grastien, A.: Compromise-free pathfinding on a navigation mesh. In: IJCAI. pp. 496–502 (2017)
8. Häcker, C., Bouros, P., Chondrogiannis, T., Althaus, E.: Most diverse near-shortest paths. In: SIGSPATIAL. pp. 229–239 (2021)
9. Jones, A.H.: Method of and apparatus for generating routes (Aug 21 2012), uS Patent 8,249,810
10. Kobitzsch, M.: An alternative approach to alternative routes: Hidar. In: European Symposium on Algorithms. pp. 613–624. Springer (2013)
11. Kobitzsch, M., Radermacher, M., Schieferdecker, D.: Evolution and evaluation of the penalty method for alternative graphs. In: ATMOS. vol. 33, pp. 94–107 (2013)
12. Li, L., Cheema, M.A.: Alternative pathfinding in game maps and indoor venues. ICAPS (2021)
13. Li, L., Cheema, M.A., Lu, H., Ali, M.E., Toosi, A.N.: Comparing alternative route planning techniques: A comparative user study on Melbourne, Dhaka and Copenhagen road networks. IEEE TKDE (2021)
14. Li, Y., Yiu, M.L., Kou, N.M., et al.: An experimental study on hub labeling based shortest path algorithms. PVLDB **11**(4), 445–457 (2017)
15. Liu, H., Jin, C., Yang, B., Zhou, A.: Finding top-k shortest paths with diversity. IEEE Transactions on Knowledge and Data Engineering (2017)
16. Moghanni, A., Pascoal, M., Godinho, M.T.: Finding shortest and dissimilar paths. International Transactions in Operational Research **29**(3), 1573–1601 (2022)
17. Oh, S., Leong, H.W.: Edge n-level sparse visibility graphs: Fast optimal any-angle pathfinding using hierarchical taut paths. In: SoCS. pp. 64–72 (2017)
18. Shen, B., Cheema, M.A., Harabor, D.D., Stuckey, P.J.: Euclidean pathfinding with compressed path databases. In: IJCAI. pp. 4229–4235 (2021)
19. Sturtevant, N.R.: Benchmarks for grid-based pathfinding. IEEE Transactions on Computational Intelligence and AI in Games **4**(2), 144–148 (2012)