OPEN: New Path-Planning Algorithm for Real-World Complex Environment

J.I. Olszewska and J. Toman

Abstract This paper tackles with the single-source, shortest-path problem in the challenging context of navigation through real-world, natural environment like a ski area, where traditional on-site sign posts could be limited or not available. For this purpose, we propose a novel approach for planning the shortest path in a directed, acyclical graph (DAG) built on geo-location data mapped from available web databases through Google Map and/or Google Earth. Our new path-planning algorithm we called OPEN is run against this resulting graph and provides the optimal path in a computationally efficient way. Our approach was demonstrated on real-world cases, and it outperforms state-of-art, path-planning algorithms.

Keywords Path planning · Graph · Algorithms · Real-world navigation · Google Map · Google Earth

1 Introduction

In natural environment, finding the suitable/shortest path is a difficult task. Indeed, in ski resorts, traditional resources, e.g. ski piste maps or sign posts, could be limited [1] or even non-existent as illustrated in Fig. 1, while current mobile and web applications such as Navionics Ski [2] or [3] could lack of accuracy, flexibility or adaptability to environmental changes. On the other hand, many path-planning algorithms and their derivations exist [4–7], but usually restricted to artificial environments or optimised only for situations such as network traffic flow or social media. This reveals the need to deploy new path-planning approaches.

In this paper, we focus thus on the study and the development of an appropriate approach for path planning in a natural domain such as a ski area, using web available geo-location data, in order to help the growing number of winter-sport users. This

J.I. Olszewska (✉) · J. Toman
University of Gloucestershire, Cheltenham, UK
e-mail: joanna.olszewska@ieee.org

M. Bramer and M. Petridis (eds.), *Research and Development in Intelligent Systems XXXIII*, DOI 10.1007/978-3-319-47175-4_17

Fig. 1 Natural environment consisting of a ski area without sign posts

also implies the design of an adapted description and modelling of the application domain, i.e. the ski domain.

In particular, the contribution of this paper is a novel path-planning algorithm called *OPEN* based on the integration of multiple planning algorithms. Our approach provides an *O*ptimal *P*ath within a minimized *E*xecution time and a reduced number of explored *N*odes, i.e. with a capped memory size. It relies on the parallel computing of a set of path-finding algorithms embedded into our OPEN algorithm finding a triply optimised solution against three constraints such as the path length (P), the execution time (E), and the number of explored nodes (N).

The paper is structured as follows. In Sect. 2, we present our new OPEN path-planning approach. Results and discussion are presented in Sect. 3, while conclusions are drawn up in Sect. 4.

2 Proposed Method

Our method consists of two main steps: (i) the processing of the geo-location data and the building of the corresponding graph which is the core of the ski domain; (ii) the computing of the single-source shortest path to help skiers and surfers to evolve safely through natural environment.

Firstly, to apply a path-planning algorithm, we need to study the underlying graph data, its density, its size, and its environment, and select suitable data structures [8, 9] that make the appropriate compromise between speed and memory constraints and that allow to meet the planned requirements. Hence, our path-planning algorithm is run against a directed, acyclical graph (DAG) built on publicly available, real-time 3D geographical data representing geo-location coordinates of downhill ski runs. Access to geo-spatial databases has been made easier than in the past [10, 11] by using online databases such as Google Map [12] and/or Google Earth [13].

Indeed, the online data extracted from Google Map/Google Earth complies with the Keyhole Markup Language (KML) 2.2 OpenGIS Encoding Standard and contains information such as piste name, piste description; including its difficulty, and

geo-location coordinates (longitude, latitude, altitude) describing the top-bottom path of the ski piste. However, data acquired from these online datasets must be pre-processed based on the analysis of geo-location data of ski resorts, before feeding graph-based operations such as finding the shortest path. Indeed, the online data could present problems like superfluous or inconsistent data and should be mitigated against. For this purpose, we apply the algorithm as presented in [14].

Secondly, to solve the single-source, shortest-path problem, we designed the OPEN algorithm. It consists in partial, parallel computing of a set of path-planning algorithms embedded into our OPEN algorithm (Algorithm 1), and in finding a triply optimised solution against three constraints which are the path length (P), the execution time (E), and the number of explored nodes (N) related to the memory size.

The OPEN algorithm relies thus on a portfolio of path-planning algorithms which are run in parallel and applied against the built network graph. In first instance, we chose five algorithms, namely, Dijkstra's, A*, Iterative Deepening A* (IDA*), and Anytime Repairing A* (ARA*) with two different inflation factors [5]. Indeed, many consider that Dijkstra's algorithm as a robust and appropriate algorithm for directed, acyclical graphs (DAG) [15] in opposite to the Bellman-Floyd-Moore algorithm which is the most appropriate algorithm for use in graphs with cycles. A* is the most widely used path-planning algorithm for both virtual and natural environments. IDA* algorithm has the smallest memory usage when run against test graphs. ARA* is suitable for a dynamic search algorithm and provides the optimal path when the inflation factors is equal to one; otherwise, it is sub-optimal in terms of path, but it is faster. Hence, our portfolio could contain heterogeneous path-planning algorithms; this design allowing the modularity of our approach, i.e. its 'openness'.

Next, the set of selected algorithms is computed through a competitive process, where the fastest solutions are processed within our OPEN algorithm (Algorithm 1) to find the optimal path, while keeping execution time and memory size low.

Thus, the OPEN path-planning algorithm can be applied to a graph built based on Google Map/Google Earth data to enable users to find the shortest path from one point to another, aiding the navigation across the ski pistes that make up a ski resort.

3 Experiments

We have carried out experiments to evaluate, test, and validate our novel approach. In particular, we have assessed OPEN's computational efficiency in terms of precision, speed, and memory size of our system implemented in Java using the Eclipse Java EE IDE for Web Developer and Google Earth version 7.1.2.41. The dataset used to test this system is freely available under the Creative Commons 3.0 licence and is in KML 2.2 format. It represents a contained area of Whistler Blackcomb (Fig. 2), the American largest Ski Area [16], with different difficulties of pistes and numerous intersection points between the pistes. A tolerance of 15 meters is set for the line

(a) (b)

Fig. 2 Data of Whistler Blackcomb ski area, available at: **a** Google Map and **b** Google Earth

intersections and all altitude coordinates are recorded as zero in the dataset. Consequently, the system will ignore in first instance the altitude when building a network graph representing this data. In a further step, the dataset could be enhanced with the altitude using services such as the Google Elevation API [17], which will return an altitude in meters above sea level for a given longitude-latitude coordinate.

In the experiments, six path-planning algorithms, i.e. Dijkstra's, A*, IDA*, ARA* with IFL = 10, ARA* with IFL = 1, and OPEN, were tested on five ski routes using the `WhistlerBlackcomb.kml` dataset. The results of the performance testing are reported in Tables 1 and 2. It is worth to note that the time measurements for the ARA* results are cumulative, whereas the node count for ARA* algorithms is not cumulative, as it shows the number of nodes expanded at each stage, each subsequent stage building on the nodes expanded in the previous stages.

As indicated in Tables 1 and 2, when searching for a non-existing path, e.g. from the start of The Saddle (TS) to the end of Lower Olympic (LO), all the algorithms find there is no path. However, IDA* occasionally results in a stack overflow exception.

For all the existing tested paths, the performance results show that Dijkstra, A*, ARA*, and OPEN algorithms have execution times within the millisecond range when applied to these graphs.

Table 1 Performance of Dijkstra, A*, and IDA* path planning algorithms, with p: the path length (in meter) computed by an algorithm, e: the execution time (in milliseconds) of an algorithm, and n: the number of explored nodes by an algorithm. The start-/end-nodes correspond to locations such as GL (Green Line), VR (Village Run), MT (Matthews Traverse), LF (Lower Franz), GR (Glacier Road), YB (Yellow Brick Road), PT (Pikas Traverse), LO (Lower Olympic), TS (The Saddle)

Start Location	End Location	Dijkstra			A*			IDA*		
		p	e	n	p	e	n	p	e	n
GL	VR	7292.3	27	1180	7292.3	12	813	7292.3	2241	8338709
MT	LF	6050.7	3	489	6050.7	2	201	6050.7	22	27259
GR	YB	8995.6	1	208	8995.6	1	208	8995.6	42	71861
PT	LO	8058.6	8	495	8058.6	8	374	8058.6	1768	6430540
TS	LO	–	–	–	–	–	–	–	–	–

Table 2 Performance of ARA* and our path planning algorithm (OPEN), with IFL: the inflation factor of the ARA* algorithm

Start Location	End Location	ARA* (IFL=10)			ARA* (IFL=1)			Our		
		p	e	n	p	e	n	p	e	n
GL	VR	7900.1	3	422	7292.3	12	743	7292.3	12	743
MT	LF	6828.2	1	70	6050.7	8	169	6050.7	2	201
GR	YB	9009.6	1	160	8995.6	10	90	8995.6	1	208
PT	LO	8119.5	1	231	8058.6	17	390	8058.6	8	374
TS	LO	–	–	–	–	–	–	–	–	–

In comparison to the other algorithms, the results from IDA* show a very slow algorithm that is very inefficient, exploring very large numbers of nodes. For example, on the route from Green Line (GL) to Village Run (VR), IDA* explores a total of 8338709 nodes, yet there are only 3349 nodes in the Whistler Blackcomb ski map. It is likely that this algorithm is not suitable for the underlying data. There are many nodes densely packed, and the edge cost between them is relatively small. For each iteration of the IDA* algorithm, the algorithm is likely only evaluating a single extra node. This is a worst-case scenario of IDA*, with the number of node expansion equal to $\Omega((N_{A*})^2)$, where N_{A*} is the number of nodes expanded by A* and Ω is the size of the subset of edges considered. Further testing looking at the memory usage profile is required to ascertain if using the IDA* algorithm is worthwhile, as well as looking at more advanced implementations of IDA* that reuse the previous iteration search results.

The use of a heuristic value in the A* algorithm provides improvements over Dijkstra's one in both execution time and number of nodes explored by the algorithm. For example, A* is far more efficient to compute the path starting at the beginning of the Green Line (GL) piste and finishing at the end of the Village Run (VR) piste than Dijkstra's. The analysis of this path shows that there are many decision points at which the algorithms need to make a decision to prioritise one route over another. More specifically, at a decision point, Dijkstra's algorithm explores additional nodes away from the destination as the cost of this path is less than the cost of the path heading toward the destination, whereas the heuristic value used in A* prioritises the path heading toward the destination and the additional nodes are not explored.

ARA* algorithm results highlight how the use of inflated heuristics produces sub-optimal results, but in less time than A* and explores fewer nodes. For example, for the path from Green Line (GL) to Village Run (VR), A* takes 12 ms and explores 813 nodes to produce the optimal route 7292.26 m long, while ARA* with inflation factor 10 takes 3 ms, explores only 422 nodes and produces a sub-optimal path. In the tests, there is no difference between using a heuristic inflation factor of 5 and 10. Further testing shows all inflation factors of 2 and greater produce the same results. As expected the inflation factor of 1 produces the optimal path.

The OPEN algorithm has computational performance at least as good as A* and in some cases even better, e.g. for the path Green Line (GL) to Village Run (VR). Moreover, the OPEN algorithm provides the optimal path unlike ARA* (IFL = 10), while OPEN outperforms the state-of-the art algorithms such as Dijkstra, IDA*, and ARA* (IFL = 1) in terms of both execution time and number of nodes explored.

4 Conclusions

In this paper, we proposed a new path-planning approach to aid people's navigation in outdoor, natural locations such as ski resorts. Indeed, automated path planning is of great utility in this novel application domain, because of the limits of traditional solutions, e.g. outdated ski resort maps or sign-post shortage in remote areas of large ski resorts, or engineering issues with current IT solutions such as native mobile apps provided by some ski resorts. Thus, we developed an original path-planning method which is focused on finding a global solution with an optimal path as well as a capped execution time and a reduced memory size, rather than on locally improving search algorithms. For this purpose, we introduced our OPEN algorithm integrating multiple path-planning algorithms. The OPEN algorithm is run on the ski domain modelled by directed, acyclic graphs based on processed online geo-location data from Google Maps/Earth. Our OPEN algorithm provides the optimal path, while it shows better computational performance than the well-established search algorithms such as A*, when tested for ski path-planning purpose. Considering the computational performance of our OPEN algorithm and the degree of generality of our proposed approach, our method could be useful for any application requiring single-source, path planning in real-world, changing 3D environments.

Appendix—OPEN Algorithm

Algorithm 1 OPEN Algorithm

Given a graph G, a start node S, an end (goal) node t, with $s \in G$ and $t \in G$.

Considering $i \in \mathbb{N}$ with $i \in I$, the algorithm set; IFL, the ARA* inflation factor;
p_i, the path length computed by an algorithm i;
e_i, the execution time of an algorithm i;
n_i, the number of node explored by an algorithm i.

Let us initialize
P, the set of computed paths by each algorithm, $P = \emptyset$;
E, the set of execution time of each algorithm, $E = \emptyset$;
N, the set of explored nodes by each algorithm, $N = \emptyset$.

Let us compute

pardo ▷ parallel computing of the algorithms
  Dijkstra(s,t); **return** $P = P \cup \{p_1\}, E = E \cup \{e_1\}, N = N \cup \{n_1\}$
  A*(s,t); **return** $P = P \cup \{p_2\}, E = E \cup \{e_2\}, N = N \cup \{n_2\}$
  IDA*(s,t); **return** $P = P \cup \{p_3\}, E = E \cup \{e_3\}, N = N \cup \{n_3\}$
  ARA*(s,t,IFL=10); **return** $P = P \cup \{p_4\}, E = E \cup \{e_4\}, N = N \cup \{n_4\}$
  ARA*(s,t,IFL=1); **return** $P = P \cup \{p_5\}, E = E \cup \{e_5\}, N = N \cup \{n_5\}$
until # E = # I - 1 ▷ eliminate the slowest algorithm

if $P \neq \emptyset$ **then**
  $p_o = \min(P)$
  for all $j = 1 : \#P$ **do** ▷ find the shortest path(s)
    if $p_j \neq p_o$ **then**
      $P = P \setminus \{p_j\}, E = E \setminus \{e_j\}, N = N \setminus \{n_j\}$
    end if
  end for

  $e_o = \min(E)$
  for all $k = 1 : \#E$ **do** ▷ find the fastest algorithm(s)
    if $e_k \neq e_o$ **then**
      $P = P \setminus \{p_k\}, E = E \setminus \{e_k\}, N = N \setminus \{n_k\}$
    end if
  end for

  $n_o = \min(N)$
  for all $l = 1 : \#N$ **do** ▷ find the most efficient algorithm(s)
    if $n_l \neq n_o$ **then**
      $P = P \setminus \{p_l\}, E = E \setminus \{e_l\}, N = N \setminus \{n_l\}$
    end if
  end for

else'no path'
end if

return head(P), head(E), head(N) ▷ find the optimal solution

References

1. Watts, D., Chris, G.: Where to Ski and Snowboard 2014. NortonWood Publishing, Bath (2014)
2. NavionicsSki: Navionics Ski (Version 3.3.2) for Android (Mobile Application Software) (2014). https://play.google.com/store
3. RTP LLC: Whistler Blackcomb Live for Android (Mobile Application Software) (2014). https://play.google.com/store
4. Baras, J., Theodorakopoulos, G.: Path Planning Problems in Networks. Morgan & Claypool Publishers, Berkley (2010)
5. Edelkamp, S., Schroedl, S.: Heuristic Search. Morgan Kaufmann, Waltham (2011)
6. Hernandez, C., Asin, R., Baier, J.A.: Reusing previously found A* paths for fast goal-directed navigation in dynamic terrain. In: Proceedings of the AAAI International Conference on Artificial Intelligence, pp. 1158–1164 (2015)
7. Uras, T., Koenig, S.: Speeding-up any-angle path-planning on grids. In: Proceedings of the AAAI International Conference on Automated Planning and Scheduling, pp. 234–238 (2015)
8. Nguyen, T.D., Schmidt, B., Kwoh, C.K.: SparseHC: a memory-efficient online hierarchical clustering algorithm. In: Proceedings of the International Conference on Computational Science, pp. 8–19 (2014)
9. Sedgewick, R., Wayne, K.: Algorithms, 4th edn. Addison-Wesley, New Jersey (2011)
10. Breunig, M., Baer, W.: Database support for mobile route planning systems. Comput. Environ. Urban Syst. **28**(6), 595–610 (2004)
11. Hulden, M., Silfverberg, M., Francom, J.: Kernel density estimation for text-based geolocation. In: Proceedings of the AAAI International Conference on Artificial Intelligence, pp. 145–150 (2015)
12. GoogleMaps: Google Maps Web Database (2015). http://maps.google.com/
13. GoogleEarth: Google Earth Web Database (2015). http://earth.google.com/
14. Toman, J., Olszewska, J.I.: Algorithm for graph building based on Google Maps and Google Earth. In: Proceedings of the IEEE International Symposium on Computational Intelligence and Informatics, pp. 80–85 (2014)
15. Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest paths algorithms: theory and experimental evaluation. Math. Program. **73**(2), 129–174 (1996)
16. WhistlerBlackcomb: Winter Trail Map of the WhistlerBlackcomb ski resort (2014). http://www.whistlerblackcomb.com/~/media/17fbe6c652bd4212902aa2f549a5df9f.pdf
17. Google: Google Elevation API (2016). https://developers.google.com/maps/documentation/elevation/