Generalized GETSTARS is NP-Complete

Jack Eisenmann

1. Definition of Generalized GETSTARS

The standard version of GETSTARS involves controlling a ghost to collect ten stars in an enclosed 10 by 10 board. The board may also contain three 2 by 2 block barriers through which the ghost may not pass. Barriers must have at least one block of space between each other. The ghost is only able to move in one orthogonal direction at a time.

In this document, we will use a generalized version of GETSTARS with these modifications:
- The board is unbounded
- There may be any number of stars
- There may be any number of barriers

We will say that the GETSTARS language consists of ⟨B, t⟩ such that:
- B is a board containing star positions, barrier positions, and initial player position
- t is a non-negative integer
- It is possible for the player to collect all stars in B by making t or fewer moves

2. Overview of proof

We will prove that GETSTARS is NP-complete in the following way:
1. Show that GETSTARS is NP-hard by reduction from planar HAMPATH (Hamiltonian path problem)
2. Show that GETSTARS is in NP
3. Conclude that GETSTARS is NP-complete because it is NP-hard and in NP

3. GETSTARS is NP-hard

Planar HAMPATH (Hamiltonian path problem) is NP-complete[1]. Therefore if we can poly-time reduce planar HAMPATH to GETSTARS, we can conclude that GETSTARS is NP-hard.

HAMPATH is the language of ⟨G, a, b⟩ such that there exists a Hamiltonian path in graph G between the two given vertices a and b. A hamiltonian path is a path which visits every vertex of the graph exactly once.

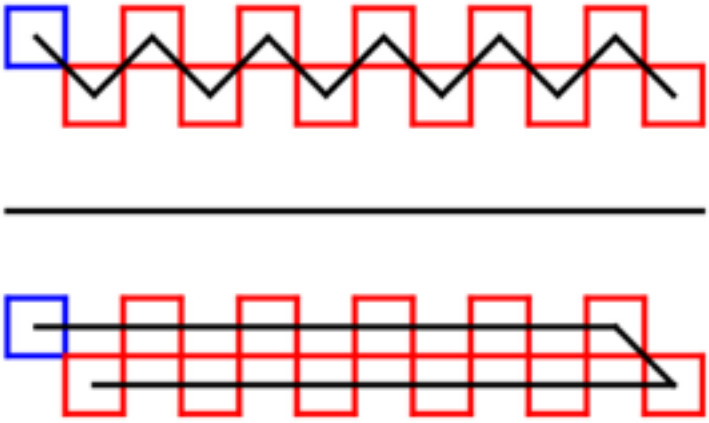
Planar HAMPATH adds the restriction that the graph must be able to be drawn on a 2D surface so that no two edges cross.

We want to find some poly-time computable function f such that x $\in$ planar HAMPATH iff f(x) $\in$ GETSTARS.

Our reduction f from planar HAMPATH to GETSTARS will involve the following steps:

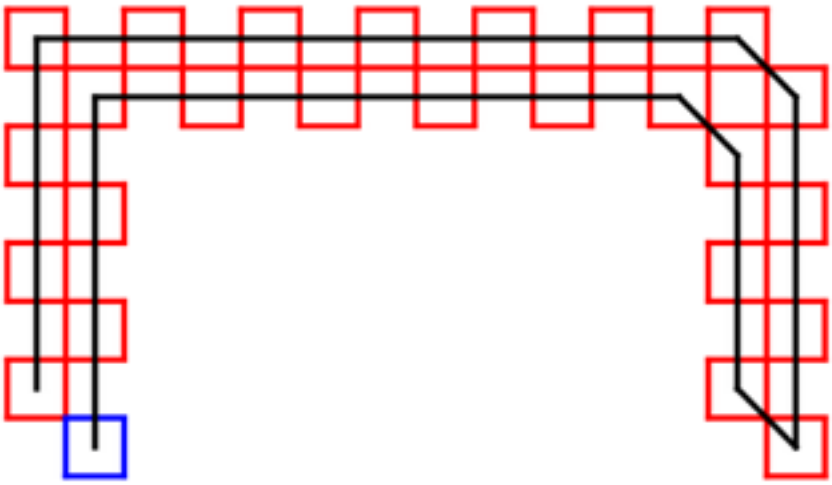
1. Convert each vertex in the HAMPATH graph into a super vertex composed of stars
2. Convert each edge in the HAMPATH graph into a super edge in GETSTARS
3. Do not add any barriers
4. Start the player two spaces away from the start super vertex
5. Create a super vertex of degree one connected to the end vertex b of the HAMPATH graph
6. Set t equal to [number of stars * 2 + (number of super vertices - 1) * 2]

A super vertex will contain channels with a checkerboard structure. The diagram below shows two channels with two different pathways through them. Blue squares represent starting positions. Red squares represent stars. Black lines represent paths.

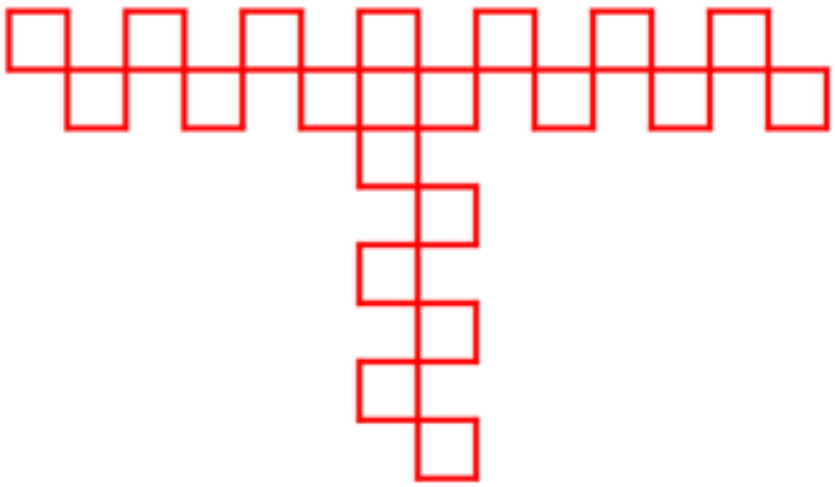


In both paths, every star is visited. The distance travelled between each star is always 2 spaces. In the top channel, the end position is on the other side of the channel from the start position. In the bottom channel, the end position is next to the start position.

It is possible for a channel to bend and still maintain both types of paths:

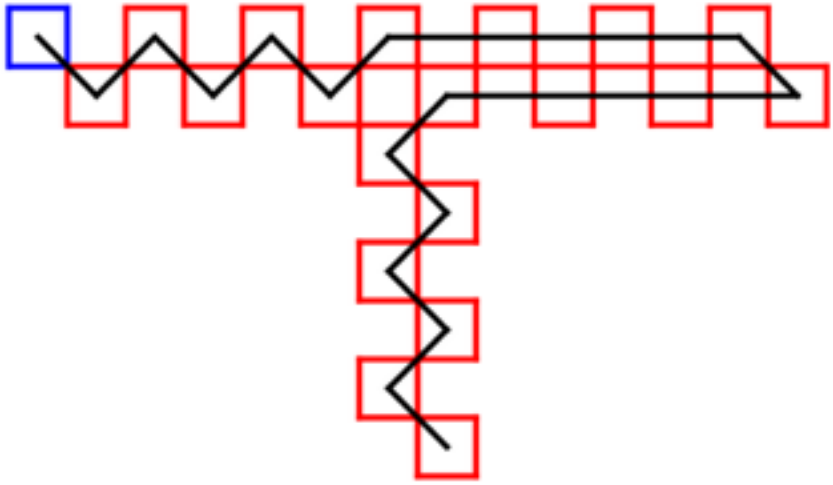


A supervertex may contain junctions of 3 channels with the following structure:

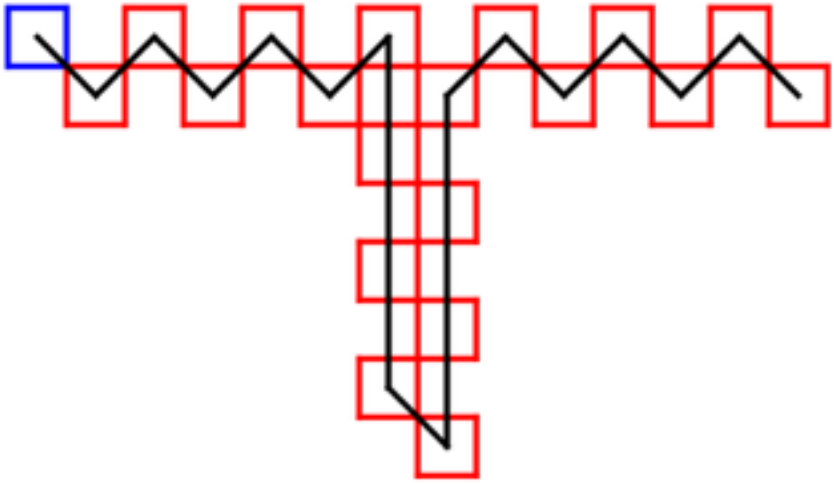


Given any start channel, it is possible to completely traverse all stars (2 spaces at a time) and exit the end of either other channel. There are six possible cases for this.

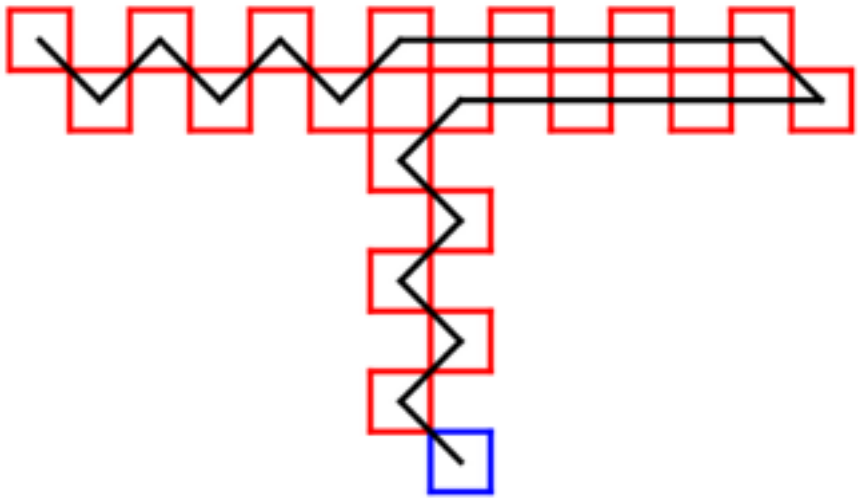
Case 1: West → East → South



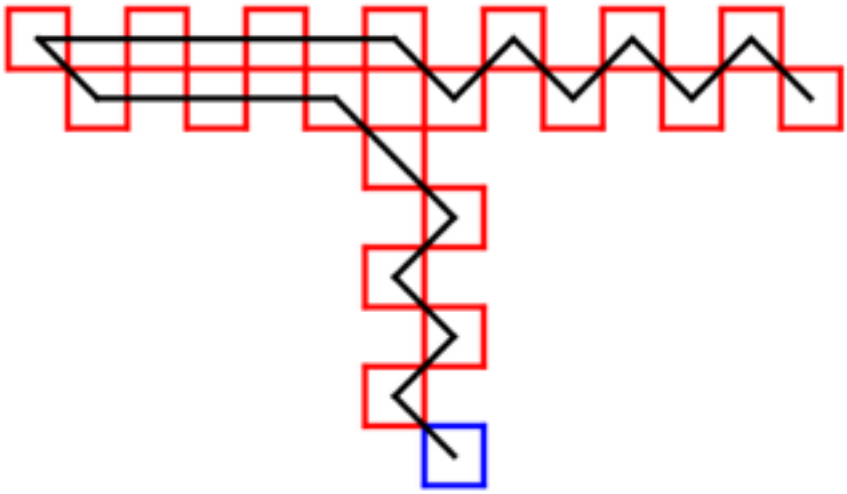
Case 2: West → South → East



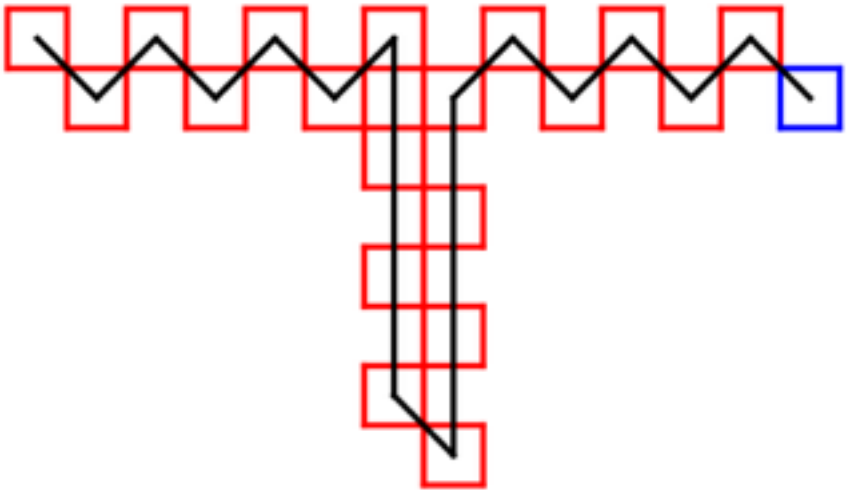
Case 3: South → East → West



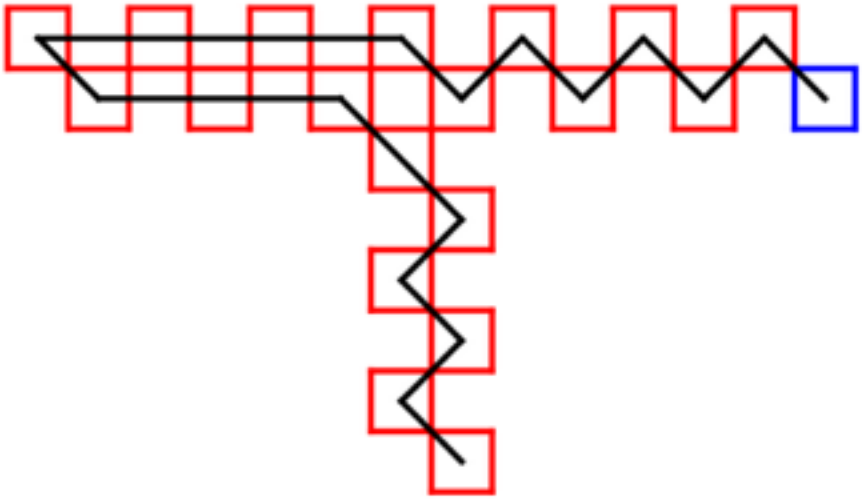
Case 4: South → West → East



Case 5: East → South → West

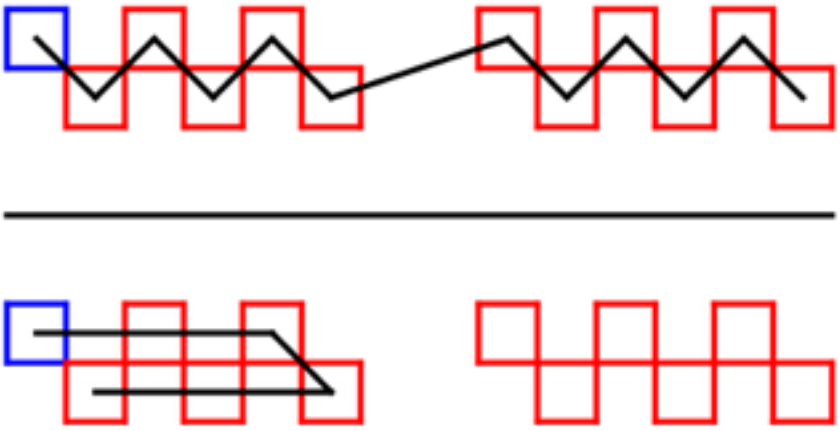


Case 6: East → West → South



We can include many junctions in a super vertex. Provided that there are no cycles, it will always be possible to traverse all stars in the super vertex (2 spaces at a time) while travelling from one endpoint to another. This allows a super vertex to have an arbitrarily large number of interconnected channels.

A super edge consists of channels of two super vertices which meet within exactly 4 spaces of each other:



In the top case, the path traverses across the super edge from one super vertex to another. In the bottom case, the path does not traverse across the super edge.

Generating the super graph is a poly-time operation because, in general, drawing any planar graph is a poly-time operation[2]. Adding the extra super vertex after the end node in HAMPATH is also a poly-time operation. Therefore the reduction f from planar HAMPATH to GETSTARS may be performed in polynomial time.

Suppose that x $\in$ planar HAMPATH. This means there is a path in G from a to b which visits every vertex in G exactly once. This means there exists a corresponding path through B which visits every super vertex exactly once. This means that the number of spaces moved is equal to [number of stars * 2 + (number of super vertices - 1) * 2], because there will be [number of super vertices - 1] transitions across super edges. Therefore f(x) $\in$ GETSTARS.

Suppose that f(x) $\in$ GETSTARS. This means that all of the stars in B may be collected in [number of stars * 2 + (number of super vertices - 1) * 2] moves or fewer. The number of moves required to collect stars within the same supervertex is [number of stars * 2]. The number of moves for each super edge traversed is 4. Therefore the number of super edges traversed must be [number of super vertices - 1]. This means that each super vertex was only visited once. The end super vertex must be the extra super vertex added after b, because there is only one edge connected to the extra super vertex. Therefore there exists a corresponding path in G which is a Hamiltonian path between a and b. Therefore x $\in$ planar HAMPATH.

Therefore GETSTARS is NP-hard.

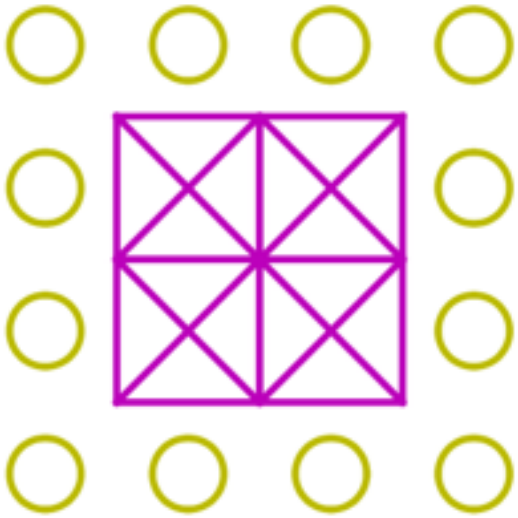
4. GETSTARS is in NP

We may employ the following non-deterministic algorithm to solve GETSTARS:
1. Non-deterministically choose an ordering of vertices to visit
2. Find the total number of moves required to traverse the vertices in the given order
3. If the number of moves is less than or equal to t, accept. Otherwise reject.

This algorithm will accept iff there exists a way to collect all stars in t moves or fewer.

Without barriers, it is trivial to find the distance between any two stars. When barriers are included, the algorithm must find an optimal path around them.

In order to find an optimal path around barriers, we will employ Dijkstra's algorithm. We will create a vertex for the start star and end star. In addition, we will create 12 vertices around every barrier. The barrier blocks shown below are purple, and the vertices are yellow:



In total the number of possible edges to add is polynomial in the number of barriers. We will only add edges whose orthogonal paths do not intersect with barriers. The weight of each edge will be the number of spaces required to move from one vertex to another.

The runtime of Dijkstra is polynomial in the number of edges and vertices[3]. The number of edges and vertices in this case are both polynomial in the number of barriers, so the path finding step will require polynomial time.

Therefore GETSTARS may be solved in non-deterministic polynomial time.

5. GETSTARS is NP-complete

Because GETSTARS is NP-hard and is in NP, GETSTARS is NP-complete.

6. Bibliography

[1] Garey, M. R.; Johnson, D. S.; Stockmeyer, L. (1974), "Some simplified NP-complete problems", Proc. 6th ACM Symposium on Theory of Computing (STOC '74), pp. 47–63, doi: 10.1145/800119.803884.

[2] Martinet, Lucie (2010), "Drawing Planar Graphs" http://perso.ens-lyon.fr/eric.thierry/Graphes2010/lucie-martinet.pdf

[3] Fredman, Michael Lawrence; Tarjan, Robert E. (1984). Fibonacci heaps and their uses in improved network optimization algorithms. 25th Annual Symposium on Foundations of Computer Science. IEEE. pp. 338–346. doi:10.1109/SFCS.1984.715934.