



DAA Assignment PHASE-2

Design and Analysis of An Algorithm to Generate A Random Board for Settlers of Catan

Abhigyan Gandhi, 2018A7PS0168U, f20180168@dubai.bits-pilani.ac.in

Submitted to: Ma'am A Razia Sulthana, Instructor-In-Charge,
CS F364 Design and Analysis of Algorithms

Abstract: This document is for the assignment report for the DAA course. I have created the problem statement on my own for a board game that I play with my family. Settlers of Catan is an award-winning multiplayer board game developed by Klaus Teuber, it was released in 1995 in Germany. It was named the game of the century at GamesCon, Vegas in 2015. It can be played by four players in the normal version and six players by using an extension pack. It is a very interesting and intense game. An average game lasts for about an hour. The game is played on a board which consists of 19 different hexagonal tiles arranged in a random order. The game is set in the village of Catan where the players compete to build settlements and earn points. The first player to reach 10 points is declared the winner. You get a point for each of your settlements, and you can also gain additional points by different ways. First thing to do is to setup the board by placing the hexes randomly, then each hex is given a token randomly. There are 6 different types of hexes, namely ore, wheat, brick, wood, wool, and desert. These are the commodities in the game except the desert. The players use these commodities to trade with each other and from the bank to build settlements and roads. There are 4 hexes of wheat, wood, and wool each, 3 hexes of ore and brick and 1 hex for the desert. Each player takes turn rolling the two dices to get a number. Now the 18 tokens are used to specify each number that can come up on rolling the dices. There are 2 tokens each for the numbers 3, 4, 5, 6, 8, 9, 10 and 11, and 1 token each for 2 and 12. The number 7 has a special use which I will explain later. So, to begin the game each player takes turn to place their starting two settlements and roads. You can place settlements only on the vertices of the hexes and roads on the edges. There needs to be at least two roads in between two settlements. Now while placing the settlements the players need to strategize as to on which hexes to place their settlements taking into consideration the probability of the numbers that come more often on the dices. The highest probability is of the numbers 8 and 6 after 7. Then 9 and 5, then 10 and 4, then 11 and 3, then 12 and 2. When the dices are rolled, the players

get commodity cards of the resources on which they have placed their settlements, this is the most basic way to progress in the game, to place settlements on good hexes (high probability hexes), get more resources and use them to build and trade. When a 7 is rolled on the dices, the robber comes into play who resides on the desert hex. The player who has rolled a 7 gets to move the robber to any hex of his choice to block players from getting the resource on that hex when its token number is rolled.



Figure 1: Settlers of Catan

Another aspect of the game are the development cards which are like action cards having special abilities. These cards will help you progress faster and get ahead of your opponents. One of the most important development cards is the knight card. When a player uses the knight card they get to move the robber to any hex of their choice. You can also get victory points as a development card which is just an extra point. For building settlements, roads, cities and buying development cards you need to give specific resources to the bank. Like to build a settlement you need to give a wood, a brick, a wool and a wheat. During a players' turn they can trade with

other players to exchange resources as per their needs. If no one wants to trade with you, you can do maritime trading with the bank which means that you can give 4 of the same resource cards to the bank and get in exchange one card of your choice. To make maritime trading a little less costly, there are trading ports on the board on the outer hexes, which you can use by placing your settlements on those vertices. There are 18 vertices that are connected to a port. There are 2 ports each for all 5 types of commodities and there are extra 8 ports for all types of resource trading. When you get to a port you can do maritime trading by just giving 3 cards of the commodity mentioned on the port. You can also upgrade your settlements to cities, which are worth two points. Each player has 5 settlements and 4 cities. There are also 2 special cards worth 2 points each for building the longest road and largest army for playing the most knight cards.

Problem definition: I have been playing Catan with my family for over two years now and it is our favorite thing to do when we all are together. The most tedious part of the game is placing the first two settlements, and who gets to place them first. The figuring out part is difficult as you need to consider which commodities are important for your strategy and which hexes have the highest probabilities. So, to make that part easy I decided to develop an algorithm which will generate a random board with hexes placed and tokens assigned to each one of them. The algorithm will also rank all the 54 vertices in descending order of preference. People have done a lot of research on the game and found out scores for the resources and tokens. I will use those scores to determine the rank of the vertices. I will use the A* shortest path search algorithm as a basis to come up with a solution for the problem. The factors to consider will be that there needs to be at least a gap of two edges between two settlements and as the players start to place settlements those vertices become unavailable for other players. It is also not allowed to place settlements at the ports in the beginning. The board can be considered as a map and each vertex will be a state. The A* algorithm is used to find the shortest path between an initial and final state. Its space complexity is $O(b^d)$.

Objectives:

1. To develop an algorithm to place the first two settlements for each player on the board taking into consideration all the limiting factors mentioned in the problem definition.
2. To understand the working of the A* search algorithm and to use it as a basis for coming up with the solution.
3. To assign random positions to all the hexes and give them tokens.
4. To analyze the time complexity and space complexity of the algorithm developed.
5. To implement the algorithm using the PyCharm IDE.

Literature review table:

Reference	Objectives	Problem Statement	Methodology	Algorithm	Advantage	Disadvantage	Performance measure value
1	To track indoor position using the A* algorithm.	Passive Infrared (PIR)-Based Indoor Position Tracking for Smart Homes Using Accessibility Maps and A-Star Algorithm.(2018)	Building a grid map, designing the algorithm, developing a PIR sensor model.	A* search algorithm.	Provided a solution for home robot localization.	High space complexity.	7
2	To modify the A* algorithm for increasing efficiency.	Modified A-Star Algorithm for Efficient Coverage Path Planning in Tetris Inspired Self-Reconfigurable Robot with Integrated Laser Sensor.(2018)	Propose a zigzag plan for the new algorithm, detecting boundary waypoints, develop the star-based plan.	A* search algorithm.	Increases area coverage performance.	Low accuracy of heuristics cause reduction in speed.	6
3	To use the PRDSA scheme to protect IOT from sink hole attacks.	Design and Analysis of Probing Route to Defense Sink-Hole Attacks for Internet of Things Security.(2020)	To develop an algorithm based on the PRDSA scheme and optimize the parameters.	PRDSA scheme-based algorithm.	Better performance than existing schemes.	High energy consumption.	7
4	To find the shortest non-holonomic path.	Guided Hybrid A-star Path Planning Algorithm for Valet Parking Applications.(2019)	To develop a visibility diagram planner, create a hybrid A* algorithm.	Hybrid A* search algorithm combined with a diagram planner.	40% faster than pre-existing algorithms.	High space complexity.	8
5	To provide an automated efficient lift plan.	An A-Star algorithm for semi-optimization of crane location and configuration in modular construction.(2021)	Use the A* to develop a lift plan on a pre-planned sequence.	A* algorithm.	Reduced costs and enhanced safety.	Slow speed due to high heuristics.	8

6	To modify the A* algorithm for optimizing the transportation plan.	Modified A-star algorithm for modular plant land transportation.(2018)	Simulating multi-body dynamics by taking curves as essential parameters.	Modified A* algorithm.	Considers both distance and angle as costs.	Loss of time due to curve of load.	6
7	To find shortest path corresponding to the most efficient SOC equalization.	Active Balancing of Lithium-Ion Batteries Using Graph Theory and A-Star Search Algorithm.(2021)	Propose a battery SOC observer and analyze its stability.	A* search algorithm.	Decreased balance time and energy loss.	No robustness guarantee.	7
8	Improving the A* algorithm for better results.	Path planning for UAV tracking target based on improved A-star algorithm.(2019)	Propose a path planning strategy.	Improved A* algorithm.	Better oscillation and tracking effect.	Time taken to reach desired target is long.	8
9	To analyze exact and over-approximation of DNNs.	Star-Based Reachability Analysis of Deep Neural Networks.(2019)	Use star sets as a symbolic representation of sets of states and apply star-based algorithms to them.	Star-based novel reachability algorithms.	19 times faster speeds than Reluplex, an SMT based approach.	High space complexity.	7
10	To find a solution for data clustering problems.	An Enhanced Version of Black Hole Algorithm via Levy Flight for Optimization and Data Clustering Problems.(2019)	Consider every motion within the search space as a star and apply Levy Flight to the BH algorithm.	Levy Flight black hole algorithm.	Superior performance on a benchmarked dataset.	High natural heuristics.	8
11	Reducing collisions of mobile robots by applying fallback BA algorithm.	A new fallback beetle antenna search algorithm for path planning of mobile robots with collision-free capability.(2019)	Fallback mechanism is applied to the BA algorithm for better results.	Fallback beetle antenna algorithm.	Low time complexity.	Not that effective.	6

12	Using MC algorithm to solve Boltzmann's equation.	Evaluating radiation transport errors in merger simulations using a Monte Carlo algorithm.(2018)	Use a short evolution of the merger remnant to critically assess the errors.	Monte Carlo algorithm.	More accurate than previous approaches.	Cannot model Kilo novae and gamma-ray bursts.	7
13	To improve star identification of trackers.	Development and Ground Evaluation of Fast-Tracking Algorithm for Star Trackers.(2018)	Proposed algorithm is implemented to hardware and ground evaluation is conducted.	Fast tracking algorithms.	Calculates latest altitude 70 times faster.	High space complexity.	7
14	To improve path planning by using two methods together.	Path Planning Using Artificial Potential Field Method And A-star Fusion Algorithm.(2020)	Combine artificial potential field method and A* algorithm	A* fusion algorithm.	Handles complex dynamic obstacles easily.	High heuristics cause speed reduction.	8
15	Applying proposed algorithm to hydrodynamics models and black hole.	How to model supernovae in simulations of star and galaxy formation.(2018)	Proposed algorithm is applied for coupling mechanical feedback and errors are reduced.	FIRE-2 algorithm.	Faster than other sub-grid models.	High space complexity.	7

Implementation and Analysis: I implemented this project using the Python programming language. The algorithm was not very complex as the problem can be easily solved. I created four different python files, main.py, board_generator.py, util.py and constants.py. The constants file contains all the declarations for different variables used in the project. It has all the details regarding the hex scores, connected vertices, port vertices, connected hexes, token values, token scores, colors, and the names for all the hexes. The util file contains functions which help us to find distances between vertices, coordinates of the vertices and hexes for the vertices. The board_generator file is the place where the algorithm gets executed. It contains two classes Boarder and VertRanker. In the Boarder class, the first function assigns the hex numbers and token numbers along with the desert hex. The second function randomizes the hex numbers. Then the `_set_tokens()` function randomly assigns tokens to the hexes for the sixes and eights, then the rest of the tokens and excludes the desert hex number. Then to make the highest probability numbers i.e., six and eight, separate from each other, so the board gets evenly distributed we check the distance between them using a function from the util file. Then the next function is for assigning ranks to the vertices. The ranking function from the other class is used for this. Then the last function is for placing the settlements on the best vertices. First it checks if port vertices and single vertices are allowed, then it removes the desert vertex from the ranking list. Then it shuffles all the players randomly and starts assigning settlements according to their ranks in descending order. Then comes the second class VertRanker. The `_get_hex_rank()` function calculates the hexes' ranks by multiplying the hex scores stored in the constants file with the token assigned to that hex. The `_get_vert_rank()` function calculates the rank of the vertex by adding its hexes' rank with the port distance rank calculated in the following function. The `_get_port_distance_rank()` function calculates the vertexes' by taking the sum of all the port scores divided by the distance between the vertex and all the port vertices. The final rankings are assigned by the `_generate_normalized_ranking()` function taking in account that no two values get normalized to the same value. Then the main file is where we execute the algorithm and get the output. The output is saved in a text file, and the time taken for the execution is also printed using the time library.

For the analysis of the algorithm, the average case time complexity of an A* search algorithm is $O(b^d)$, where b is the branching factor, which is the average number of successors per state and d is the shortest path. In our case, regarding the hexes, all of them are connected to either 3, 4 or 6 of the other hexes. By calculation, the value of b comes out to be 4.4211. This algorithm has good heuristics which helps it to prune away many of the b^d nodes. There are 11 for loops including all the files. 9 out of the 11 loops have logarithmic time as they reduce the size of input in each step, the remaining 2 loops have linear time as they iterate through the complete lists. Therefore, the total time complexity for the algorithm is $O(b^d + \log(n) + n)$.

Architecture diagram: I have implemented the A* search algorithm for this project. Following is the architecture of this algorithm.

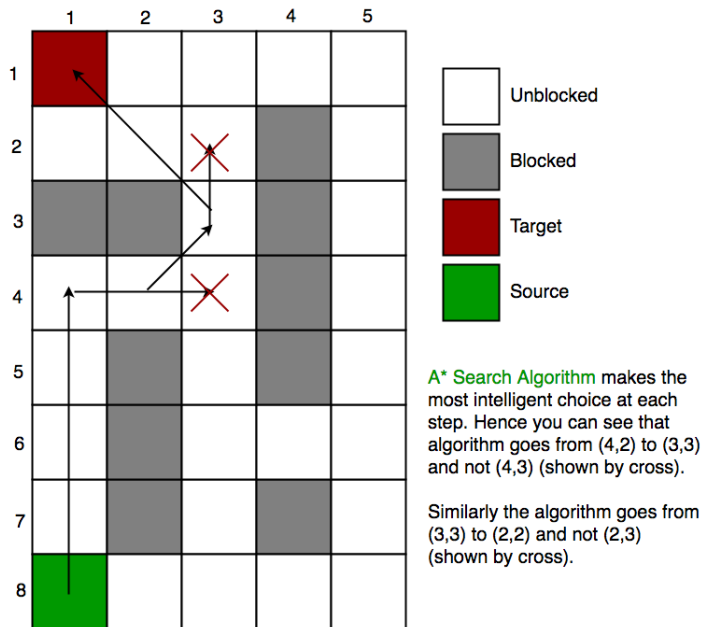


Figure 2: A-star algorithm

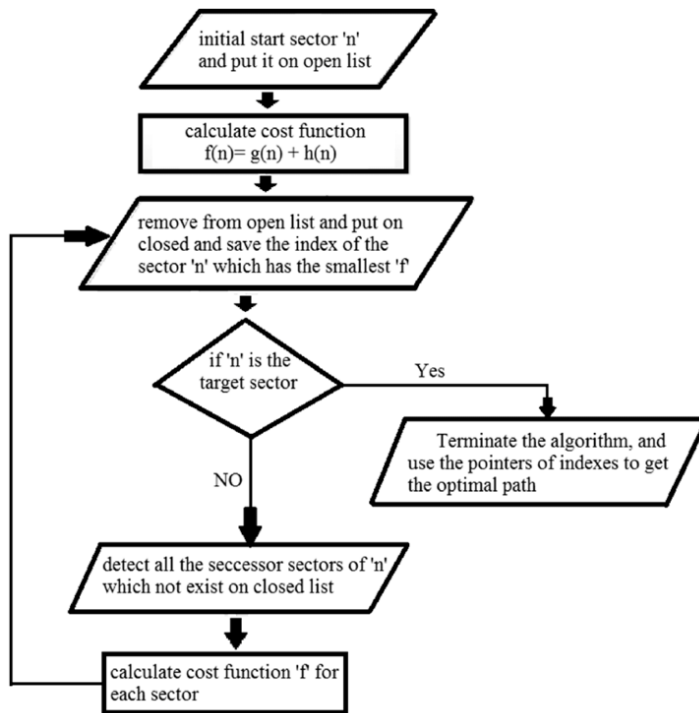


Figure 3: Flow chart for the A-star algorithm

Results and conclusion: In this project we have implemented the A-star algorithm for generating a random board for the game Settlers of Catan. The algorithm was not that complex, and the execution was very fast. The time complexity for the algorithm came out to be $O(b^d + \log(n) + n)$. The space complexity is like other graphing algorithms. It was found over 10 epochs, that the average time for complete execution was 0.1249. The algorithm had good heuristics which helped in pruning some nodes and improving the execution time. We can conclude that the A-star algorithm is very efficient for graphing problems when the heuristics are good. This was a small project having low complexity, so the algorithm was very precise and accurate.

References:

1. Yang, Dan, et al. "Passive infrared (PIR)-based indoor position tracking for smart homes using accessibility maps and a-star algorithm." *Sensors* 18.2 (2018): 332.
2. Le, Anh Vu, et al. "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor." *Sensors* 18.8 (2018): 2585.
3. Liu, Yuxin, et al. "Design and analysis of probing route to defense sink-hole attacks for Internet of Things security." *IEEE Transactions on Network Science and Engineering* 7.1 (2018): 356-372.
4. Sedighi, Saeid, Duong-Van Nguyen, and Klaus-Dieter Kuhnert. "Guided hybrid A-star path planning algorithm for valet parking applications." *2019 5th international conference on control, automation and robotics (ICCAR)*. IEEE, 2019.
5. Bagheri, S. Marzieh, et al. "An A-Star algorithm for semi-optimization of crane location and configuration in modular construction." *Automation in Construction* 121 (2021): 103447.
6. Kang, Nam Kyu, Ho Joon Son, and Soo-Hong Lee. "Modified A-star algorithm for modular plant land transportation." *Journal of Mechanical Science and Technology* 32.12 (2018): 5563-5571.
7. Dong, Guangzhong, et al. "Active Balancing of Lithium-Ion Batteries Using Graph Theory and A-Star Search Algorithm." *IEEE Transactions on Industrial Informatics* 17.4 (2020): 2587-2599.
8. Cai, Yingzhe, et al. "Path planning for UAV tracking target based on improved A-star algorithm." *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*. IEEE, 2019.
9. Tran, Hoang-Dung, et al. "Star-based reachability analysis of deep neural networks." *International Symposium on Formal Methods*. Springer, Cham, 2019.
10. Abdulwahab, Haneen A., et al. "An enhanced version of black hole algorithm via levy flight for optimization and data clustering problems." *IEEE Access* 7 (2019): 142085-142096.
11. Wu, Qing, et al. "A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability." *Soft Computing* 24.3 (2020): 2369-2380.
12. Foucart, Francois, et al. "Evaluating radiation transport errors in merger simulations using a Monte Carlo algorithm." *Physical Review D* 98.6 (2018): 063007.
13. SATO, Yuji, et al. "Development and Ground Evaluation of Fast Tracking Algorithm for Star Trackers." *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan* 16.3 (2018): 202-209.
14. Ju, Chunyu, Qinghua Luo, and Xiaozhen Yan. "Path Planning Using Artificial Potential Field Method And A-star Fusion Algorithm." *2020 Global Reliability and Prognostics and Health Management (PHM-Shanghai)*. IEEE, 2020.
15. Hopkins, Philip F., et al. "How to model supernovae in simulations of star and galaxy formation." *Monthly Notices of the Royal Astronomical Society* 477.2 (2018): 1578-1603.

Turnitin report:

Turnitin

Class Portfolio

My Grades

Discussion

Calendar

NOW VIEWING: HOME > DAA 2SEM 2020-2021

Class Homepage

This is your class homepage. To submit to an assignment click on the "Submit" button to the right of the assignment name. If the Submit button is grayed out, no submissions can be made to the assignment. If resubmissions are allowed the submit button will read "Resubmit" after you make your first submission to the assignment. To view the paper you have submitted, click the "View" button. Once the assignment's post date has passed, you will also be able to view the feedback left on your paper by clicking the "View" button.

Assignment Title	Info	Dates	Similarity	Actions
Assignment1	i	Start 21-Mar-2021 3:37PM Due 26-Mar-2021 11:59PM Post 26-Mar-2021 11:59PM	0% <div></div>	Submit View Download
Assignment2	i	Start 21-Apr-2021 7:40PM Due 30-Apr-2021 11:59PM Post 30-Apr-2021 11:59PM	0% <div></div>	Resubmit View Download

Copyright © 1998 – 2021 Turnitin, LLC. All rights reserved.

[Privacy Policy](#) [Privacy Pledge](#) [Terms of Service](#) [EU Data Protection Compliance](#) [Copyright Protection](#) [Legal FAQs](#) [Helpdesk](#) [Research Resources](#)

Feedback Studio - Google Chrome

ev.turnitin.com/app/carta/en_us/?s=8u=1104437242&o=1573297857&student_user=1&lang=en_us

turnitin

Abhigyan Gandhi | daa assignment

Match Overview

0%

< >

0

1

2

3

4

5

6

7


8

9

10

There are no matching sources for this report.

BITS PILANI DUBAI CAMPUS
ACADEMIC CITY
DUBAI
April 29, 2021



DAA Assignment PHASE-2

Design and Analysis of An Algorithm to Generate A Random Board for Settlers of Catan

Abhigyan Gandhi, 2018A7PS0168U, 20180168@dubai.bits-pilani.ac.in

Submitted to: Ma'am A Razia Sulthana, Instructor-In-Charge,
CS F364 Design and Analysis of Algorithms

Abstract: This document is for the assignment report for the DAA course. I have created the