The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

Generalize A* Algorithms to Solve Travel Salesman Problem

Maha Sabah Saeed[1], Shilan Abdullah Hassan[2], Manal Ali Atiyah[3]

[1,2]Network Department, Computer Science Institute, Sulaimani Polytechnic University, Sulaimani, Iraq
[3]Database Department, Computer Science Institute, Sulaimani Polytechnic University, Sulaimani, Iraq

Email: maha.saeed@spu.edu.iq[1], shilan.abdullah@spu.edu.iq[2], manal.Ali@spu.edu.iq[3]

Abstract:

Generalize A* is a search algorithm that has widely been used in the pathfinding research group. Its efficiency, simplicity, and modularity are often highlighted as its strengths compared to other algorithms. Dijkstra algorithm is another type of A* but without using of heuristic function. These algorithms were used previously to find the shortest path between two states (start and goal).

Travelling Salesman Problem (TSP) is one of the most common combinatorial optimization problems. It comes in the categorization of NP-Hard problems. The solutions of TSP are not possible using traditional algorithms. It is having many application branches like mathematics, computer science, and engineering. TSP is designed to find the shortest path by visiting all instances of the problem. This study makes an adaptation to the A* algorithm to work on TSP with the heuristic function that tries to look for a better path which gives priority to nodes that are supposed to be better than others and Dijkstra's algorithm just explores all possible ways and compare between the two algorithms for the same problem. The study result will help for future studies.

Keywords: A* algorithm, Dijkstra's algorithm, travelling salesman problem (TSP), heuristic function (HF).

الملخص:

تصبح خوارزميات البحث من طرق تحسين الملائمة لحل العديد من المشكلات المعقدة. A * هي خوارزمية بحث يتم استخدامها على نطاق واسع في مجموعة البحث عن المسار. غالبًا ما يتم تمييز كفاءتها وبساطتها ونمطيتها كنقاط قوتها مقارنة بالخوارزميات الأخرى. خوارزمية Dijkstra هي نوع آخر من A * ولكن بدون استخدام وظيفة الكشف عن مجريات الأمور. استخدمت هذه الخوارزميات سابقًا للعثور على أقصر مسار بين نقطتين (البداية والهدف).

تعد مشكلة البائع المتجول (TSP) واحدة من أكثر مشاكل التحسين الاندماجي شيوعًا. يأتي في تصنيف مشاكل NP-Hard. حل مشكلة TSP غير ممكنة باستخدام الخوارزميات التقليدية. كما ان لديها العديد من الفروع التطبيقية مثل الرياضيات وعلوم الكمبيوتر والهندسة. تم تصميم TSP للعثور على أقصر طريق من خلال زيارة جميع العقد. تعمل هذه الدراسة على إجراء تكيّف على خوارزمية A * للعمل علىTSP مع وظيفة الكشف عن مجريات الأمور التي يحاول البحث عن مسار أفضل ان يعطي الأولوية للعقد التي من المفترض أن تكون أفضل من غيرها للوصول الى الهدف، بينما تستكشف خوارزمية Dijkstra جميع الطرق الممكنة للوصول الى الهدف والمقارنة بين خوارزمياتان لنفس المشكلة. نتائج الدراسة لها تأثير على الدراسات المستقبلية.

الكلمات المفتاحية: خوارزمية A * ، خوارزمية Dijkstra ، مشكلة البائع المتجول (TSP) ، الوظيفة الإرشادية (HF).

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

پوختە:

ئەلگۆریتمەکانی گەڕان دەبنە شێوازی گونجاوی باشکردن بۆ چارەسەرکردنی زۆرێک لە کێشە ئاڵۆزەکان. *A ئەلگۆریتمێکی گەڕانە کە بە شێوەیەکی بەرفراوان لە گروپی تۆڕینەوەی دۆزینەوەی ڕێگادا بەکارهاتووە. زۆرجار کارایی و سادەیی و مۆدیۆلارییەکەی وەک خاڵە بەهێزەکانی بە بەراورد بە ئەلگۆریتمەکانی تر ئاماژەی پێدەکرێت. ئەلگۆریتمەکەی Dijkstra جۆرێکی تری لە *A یە بەڵام بەبێ بەکارهێنانی فەنکشنی هێوریستی. ئەم ئەلگۆریتمانەی پێشتر بەکاریان دەهێنا بۆ دۆزینەوەی کورتترین ڕێگا لە نێوان دوو خاڵ (دەستپێکردن و ئامانج).

کێشەی فرۆشیاری گەشتیاری (TSP) یەکێکە لە باوترین کێشەکانی باشکردنی تێکەڵاو. لە پۆلێنکردنی کێشەکانی NP-Hard دێت. چارەسەرەکانی کێشەی TSP بە بەکارهێنانی ئەلگۆریتمەکانی تەقلیدی ناتوانرێت. خەریکە چەندین لقی ئەپڵیکەیشنی هەیە وەک لە بیرکاری، زانستی کۆمپیوتەر، و ئەندازیاری TSP .بۆ دۆزینەوەی کورتترین ڕێگا لەگەڵ سەردانیکردنی هەموو نموونەکانی کێشەکە دارێژراوە. ئەم تۆڕینەوەیە گونجاندنێک لەسەر ئەلگۆریتمەکەی *A دەکات بۆ کارکردن لەسەر TSP بە فەنکشنی هێوریستی کە هەوڵدەدات بەدوای ڕێڕەوێکی باشتردا بگەڕێت کە ئەوڵەویەت دەدات بەو گرێیانەی کە بڕیارە باشتر بن لەوانی تر و ئەلگۆریتمەکەی Dijkstra تەنها هەموو ڕێگاکانی ئەگەری دەکۆڵێتەوە و بەراورد دەکات لە نێوان ئەو دووانەدا ئەلگۆریتمەکان بۆ هەمان کێشە. ئەنجامی تۆڕینەوەکە کاریگەری لەسەر تۆڕینەوەکانی داهاتوو هەیە.

کلیلە وشە: ئەلگۆریتمەکەی A*، ئەلگۆریتمەکەی Dijkstra، کێشەی فرۆشیاری گەشتیاری (TSP)، ئەرکی ئاماژەدەر (HF).

1- Introduction:

A* algorithm is generally used to find the shortest path between two endpoints. Examples of such problems include telephone traffic routing, network traffic routing, the games industry, robot path planning, etc. As the importance of these fields increases, the A* algorithm has become the most popular algorithm to solve such kinds of problems [1].

Many types of algorithms are used to solve TSP, for example by using genetic algorithms, Greedy Heuristic Algorithms, spider monkey, and many others. More efficient solutions are required to solve the pathfinder problems in a more complicated situation with bounded time and resources [2].

The objective of this research is to adapt the steps of the A* algorithm to work with travelling salesman pathfinding problems. This means not only finding the shortest path but also visiting all nodes for one time that is not from the A* traditional algorithm aspect.

2- Related Works

In past years, researchers have suggested several algorithms to solve TSP and many types of improvement on the A-Star algorithm, the most popular are:

1- Genetic Algorithms (GA) that are used to solve TSP by using many types of (encoding, crossover, and mutation) then find the fitness of the population to obtain the best gene [3, 4].
2- Artificial Bee Colony (ABC) algorithm is a new swarm-based optimization algorithm, inspired by the foraging behavior of honeybees [5].
3- Combinatorial ABC (CABC) is one of the successful discrete versions of (ABC) algorithm that can be used for combinatorial optimization problems that are applied to solve TSP [6].
4- The (CABC) algorithm was developed into the Quick Combinatorial Artificial Bee Colony - qCABC- optimization algorithm for TSP which the onlooker bees' behavior is modeled in a more detailed way [7].

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

5- Solving the Traveling Salesman's Problem Using the African Buffalo Optimization (ABO) is used to build a mathematical model from the behavior of a species of wild cows and uses the model to solve six difficult asymmetric instances from the TSPLIB and 33 benchmarks symmetric Traveling Salesman's Problem [8].

6- An Improved A-Star Algorithm Considering Water Current, Traffic Separation, and Berthing for Vessel Path Planning is designed to obtain the path cost of the vessel in factors of bridge pier, moored or anchored ship, port, shore, path length, current water, obstacle collision risk, traffic separation rule and maneuverability restriction [9].

7- Path Planning of Restaurant Service Robot Based on A-star Algorithms with Updated Weights is created to use an improved A-Star algorithm to discover the best path for a restaurant service robot by using the gridded map of a restaurant to verify the results [10].

8- An Energy Efficient Routing Protocol for Wireless Sensor Networks using A-star Algorithm. In this paper, a new energy-efficient routing protocol (EERP) has been designed for wireless sensor networks using the A-star algorithm. The suggested routing method enhances the network lifetime by sending data packets via the efficient shortest path [11].

In this paper, the adaptive A* star algorithm is used to solve TSP. The rest of the paper is organized as follows Section (3) presents the methodology of the traditional A* algorithm. describes the proposed method, and the simulation results and conclusion are demonstrated in section (4).

3- Methodology

Traditional A* and Dijkstra's algorithm are generic search algorithms that can be used to find solutions for many problems, pathfinding just being one of them [12]. In the standard steps that are used in A*, g(n) represents the exact cost from starting node to any node (n), h(n) represents the estimated cost from the node (n) to the destination node (goal), and f(n)=g(n)+h(n). Fig.1 explains the algorithm step-by-step. Dijkstra's algorithm is a special case of A* algorithms, where h(n)=0 for all nodes [13]. Travelling Salesman Problem is one of the most popular problems that usually new techniques are used to solve it. The description of the problem is: Given a set of cities, the distance between every pair of cities, and the problem is to find the shortest possible path that visits every city exactly one time.

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

Algorithm 1 Traditional A*

```
Input: A graph G(I,J) with source node start and goal node end
Output: Least cost path from start to end
steps:
procedure INITIALIZE
    open − list ← start
    closed − list ← ()
    g(start) ← 0
    h(start) ← heuristic − function(start, end)
    f(start) ← g(start) + h(start)
    while open − list ≠ empty do
        m ← node on top of open-list, with least f
        if m == end then
            return
        remove m form open-list
        add m to closed-list
        for each n ∈ child(m) do
            if n ∈ closed − list then
                continue
                cost ← g(m) + distance(n, m)
            if  n ∈ open − list & cost < g(n) then
                remove n from open-list
            if n ∈ closed − list & cost < g(n)  then
                remove n from closed-list
            if n ∉ open − list & n ∉ closed − list then
                add n to open-list
                g(n) ← cost
                h(n) ← heuristic − function(n, end)
                f(n) ← g(n) + h(n)
```

Fig. 1: Traditional A* algorithm

Table 1: The comparison between using traditional A* and adaptation A*algorithm

Algorithm	**A***	**Adaptation A***
Path	Shortest path (not for all cities)	Shortest path by visiting all cities
Heuristic Function (HF)	Use the estimated distance between any city and the goal	Prepared with HF and without HF (Dijkstra's algorithm)
TSP	Can't use traditional A*	Can use it

3.1 Properties of A* algorithm

A* has several useful properties: First, completeness which means A* is guaranteed to find a path from the start to the goal if there exists a path. The second is admissibility implies to optimal performance if h(n) is an admissible heuristic, which means h(n) is always less than or equal to the actual cheapest path cost from (n) to the goal. The third property of A* is complexity, where it makes the most efficient use of the heuristic. That is, no search methods that use the same heuristic function to find an optimal path examine fewer nodes than A* [14,15].

3.2 Experimental study for generalize A* algorithm

In this paper, some adaptations of the A* algorithm done to work on TSP that mentioned in the following table (1):

According to the Path factor by using the traditional A* algorithm only the shortest path (from start to goal) is required, but in adaptation A*, it must visit all cities only one time. According to the heuristic function (HF) with traditional A* use the estimated distance between the start node and the goal with the distance from node to node, but in this paper, it solved as a traditional and Dijkstra's algorithm (which means without HF) [16].

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

By using ten cities (from Iraq Map) as mentioned in table (2), all distances between nodes and the heuristic function that represents the actual distance between each node and the goal were computed by using Google Maps in KM.

Table 2: The distances between ten cities (in KM)

Cities	Sulaymaniyah	Kirkuk	Erbil	Duhok	Baghdad	Anbar	Najaf	Karbala	Basra	Hillah
Sulaymaniyah	0	94	145	264	275	312	413	359	610	363
Kirkuk	94	0	90	200	239	251	389	325	634	337
Erbil	145	90	0	116	319	311	464	399	721	414
Duhok	264	200	116	0	411	380	552	481	833	503
Baghdad	275	239	319	411	0	101	143	87	444	94
Anbar	312	251	311	380	101	0	186	115	535	151
Najaf	413	389	464	552	143	186	0	71	400	54
Karbala	359	325	399	481	87	115	71	0	423	41
Basra	610	634	721	833	444	535	400	423	0	386
Hillah	363	337	414	503	94	151	54	41	386	0

3.2.1 Process of extracting all possible solutions

With generalizing A* algorithms, helped to extract a series of all possible probabilities when entering all nodes at the same time. This has been done using the following code:

Step1: Initialization

```
Row = 0                    #the row being worked on
N = 10                     #number of Nodes
Probability = factorial (N-1) #number of possible solutions
City = 1                   #the start node
City-list={}
```

Step2: Find column solutions (from the first col. to n-2)

For (each Col. till N-2)

While (Row < Probability) and (City<=N)
Probability-Col= factorial (Col)
While (Probability-Col ≠ Zero) and (City ∉ City-list)
add City to the City-list
decrease Probability-Col by one
go to Row+1
go to next City

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

Step3: Find column solutions (of N-2 and N-1)

Row = 0
While (Row <= Probability)
If (City ∉ City-list) and (City<=N)

If (not reached to the last Col.)
add City to City-list in Row
add City to City-list in Row+1 and Col+1

Else
add City to City-list in Row
add City to City-list in Row +1 and col-1

go to next col
go to next city

Else
go to next City
go to Row+2

3.2.2 Adaptive A* algorithm and Dijkstra's algorithm

In Adaptive A* algorithm depends on the heuristic function to find the shortest path node (goal) with the start point as the beginning step, then computed the remaining nodes with this goal to find the path, and finally, calculate the distance between each node with others.

In Dijkstra's algorithm, only needed to find the value of each possible solution found in the previous section and then find the minimum result as the best solution.

Building the Adaptive A* Algorithm program and Dijkstra's Algorithm program are mentioned by the following codes:

- **Adaptive A* program code**

Find the Goal # Choose the goal with the least HF
while (there is a node not visited)
while (current-city <=N) #number of Nodes
if (current-city ==goal)

go to the next city
if ((current-city <=N) and (current-city not in final-list) and (HF of current-city is minimum to other cities))
add the current-city to the final-list
go to the next city
calculate the distance of the final-list

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

- **Dijkstra's Algorithm code**

```
For (each Row till probability)
Path =0
For (each Col till N-1)
path= path + distance [City-list[Row][Col],City-list[Row][Col+1]];
Length-path [Row] = path
Find-minimum (Length-path)
```

4- Outcome Results

The experiments were performed on a laptop Windows 10 with 16 GB of RAM and Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz 2.60 GHz, by Eclipse IDE for Java Developers - 2022-03.

From the tables below table (3 and 4), the following results can be noted after five iterations.

Table 3. Running Time, Number loops and Total Distance using Dijkstra Algorithm

Test Number	Running Time Dijkstra (Milliseconds)	No. of Loops	Total Distance (in km)
1	1621	362880	1363
2	1599	362880	1363
3	1607	362880	1363
4	1548	362880	1363
5	1545	362880	1363

Table 4. Running Time, Number loops and Total Distance using A* algorithm

Test Number	Running Time A* (Milliseconds)	No. of Loops	Total Distance (in km)
1	1474	72	2017
2	1573	72	2017
3	1462	72	2017
4	1488	72	2017
5	1472	72	2017

DOI: http://dx.doi.org/10.25098/8.1.31

The Scientific Journal of Cihan University – Sulaimaniya **PP: 210-218**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

5- Conclusion

The results from using the Dijkstra Algorithm are more appropriate than the A* algorithm through the vectors of total distance and required time of implementation to solve the problem. The results from using A* algorithm are reliable in the number of loops that needs to solve the problem although they need approximately the same time of execution.

Based on the experiment, it is notable that Dijkstra Algorithm managed to achieve the optimum outcomes on the majority of the problem posed.

References

[1] A. Rafiq, T. Asmawaty Abdul Kadir, and S. Normaziah Ihsan, "Pathfinding Algorithms in Game Development," IOP Conference Series: Materials Science and Engineering, vol. 769, p. 012021, 2020/02/01 2020.
[2] H. A. Abdulkarim and I. F. Alshammari, "Comparison of algorithms for solving traveling salesman problem," International Journal of Engineering and Advanced Technology, vol. 4, pp. 76-79, 2015.
[3] E. Alkafaween and A.B.A. Hassanat, "Improving TSP Solutions Using GA with a New Hybrid Mutation Based on Knowledge and Randomness," Communications - Scientific letters of the University of Zilina, vol. 22, no. 3, pp. 128-39, 2020.
[4] S. Sharma and V. Jain, "A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem," in IOP Conference Series: Materials Science and Engineering, 2021, p. 012194.
[5] H. Jiang, "Artificial bee colony algorithm for traveling salesman problem," in 2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering, 2015, pp. 468-472.
[6] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," 2011 International Symposium on Innovations in Intelligent Systems and Applications, 2011, pp. 50-53, doi: 10.1109/INISTA.2011.5946125.
[7] B. Gorkemli and D. Karaboga, "Quick combinatorial artificial bee colony-qCABC-optimization algorithm for TSP," in 2nd International Symposium on Computing in Informatics and Mathematics, 2013, pp. 97-101.
[8] J. B. Odili and M. N. Mohmad Kahar, "Solving the Traveling Salesman's Problem Using the African Buffalo Optimization," Computational Intelligence and Neuroscience, vol. 2016, pp. 1–12, 2016, doi: 10.1155/2016/1510256.
[9] C. Liu, Q. Mao, X. Chu, and S. Xie, "An Improved A-Star Algorithm Considering Water Current, Traffic Separation and Berthing for Vessel Path Planning," Applied Sciences, vol. 9, p. 1057, 2019.
[10] R. Yang and L. Cheng, "Path Planning of Restaurant Service Robot Based on A-star Algorithms with Updated Weights," 2019 12th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2019, pp. 292-295, doi: 10.1109/ISCID.2019.00074.
[11] A. Ghaffari, "An energy efficient routing protocol for wireless sensor networks using A-star algorithm," Journal of applied research and technology, vol. 12, pp. 815-822, 2014.
[12] M. Nosrati, R. Karimi, and H. A. Hasanvand, "Investigation of the*(star) search algorithms: Characteristics, methods and approaches," World Applied Programming, vol. 2, pp. 251-256, 2012.

DOI: http://dx.doi.org/10.25098/8.1.31

[13] S. Rabin and N. R. Sturtevant, "Pathfinding architecture optimizations," in Game AI Pro 360, ed: CRC Press, 2019, pp. 1-12.
[14] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding," J. Game Dev., vol. 1, pp. 1-30, 2004.
[15] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of A* pathfinding," Procedia Computer Science, vol. 179, pp. 507-514, 2021.
[16] D. Rachmawati and L. Gustin, "Analysis of Dijkstra's algorithm and A* algorithm in shortest path problem," in Journal of Physics: Conference Series, 2020, p. 012061.

DOI: http://dx.doi.org/10.25098/8.1.31