Journal of Engineering and Applied Sciences 14 (6): 1976-1984, 2019
ISSN: 1816-949X
© Medwell Journals, 2019

Optimizing Heuristic Graph Formation with Application in Kinematic Synthesis of a Robot Arm with Revolute Joints

[2]Doaa Mahmood Badr Ali, [1, 3]Hazim Nasir Ghafil and [3]Karoly Jarmai
[1]Department of Mechanics, Faculty of Engineering, University of Kufa, Najaf, Iraq
[2]Department of Computer, Faculty of Computer Science and Math, University of Kufa, Najaf, Iraq
[3]University of Miskolc, Miskolc, H-3515 Miskolc, Egyetemvaros, Hungary

Abstract: Automated kinematic synthesis is still a raw material and researchers have to research more in this critical stage of the design process. Graph theory intensively has used in this field of science but in this study, it has been used in a very different way to present a new, simple, abstract and automated method to construct optimum kinematic synthesis for a robot. The new presented method is intended for robot manipulator with revolute joints and it is based on the heuristic graph formation principle. In this research, ant colony optimization algorithm and graph theory toolbox have introduced as tools to improve the heuristic path or graph before mapping to the equivalent kinematic synthesis of the robot. Two practical examples were presented to prove the efficiency of the new method.

Key words: Optimization, graph theory, ant colony system, kinematic synthesis, robotics, graph theory toolbox

INTRODUCTION

Kinematic synthesis for robot arm is a crucial step in robot arm design it precedes all the next kinematic and kinetic analysis. Kinematic synthesis is divided into sublevels type synthesis which is choosing the optimum topology and dimensional synthesis which is estimating the optimum dimensions of a specific type. The two levels have been chosen depending on the designer ambiguity and experience. Also, many researchers try to make the kinematic synthesis as an automated process. There were many heuristic methods and numeric optimization framework and (Liu and McPhee, 2007) is well reviewed these works and has introduced an evolutionary method using the Genetic algorithm and the incidence matrix from graph theory to find the optimum type and dimensional synthesis. Oliva and Goodman (2010) is an extension to Liu research and has used convertible agents and an evolutionary algorithm. Also, Pucheta and Cardona (2013) is well reviewed prior works and introduced a method based on graph theory to enumerate the topological alternatives and has used precision position method to dimension the feasible alternative. In this study, a simple automated method was presented to find the optimum topology as well as the optimum dimensions of a robot arm for a prescribed task. The new method is a graph theory based and works under some limitations will be described later, it says that the planned graph from the base point to the task point can be mapped (under some limitations) to an equivalent kinematic synthesis of a robot with a configuration as the same as the generated graph. The procedure is described in details and two practical examples are presented and the resulted synthesis are simulated by 3D package software for the prescribed task.

MATERIALS AND METHODS

Graph theory: The first form of this theory was established by Leonhard Euler (Thiele, 2005) while trying to solve the seven bridges of Konigsberg problem in Fig. 1, this problem had asked for a path that can be cross over all the seven bridges such that each one should be crossed once. Graph theory (West, 2001; Bollobas, 2013) will be explained briefly in the following section.

Definitions: The proposed solution in this research is based on the graph theory principles, so, we have to explain some terminologies in this field of science briefly that are useful in this text.

Graph: A collection of sets of Vertices V and Edges E is called graph and can be described by the triple (V, E, g). V is always nonempty set while E could be empty set. g is mapping called incidence mapping and for each edge, $e \in E$ there is subset $g(e) = \{u, v\}$ where u and v are vertices. Figure 2 explains an example of a graph with four vertices v_1-v_4 as well as four edges e_1-e_4.

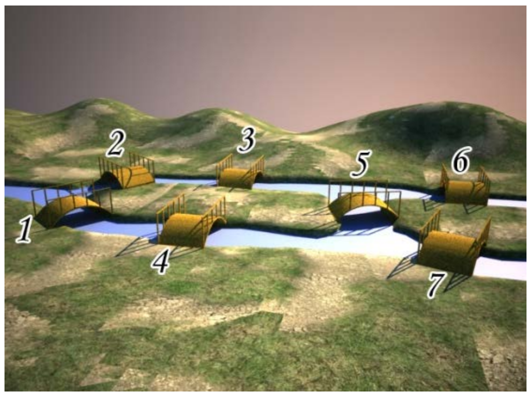


Fig. 1: Description of the seven bridges of Konigsberg

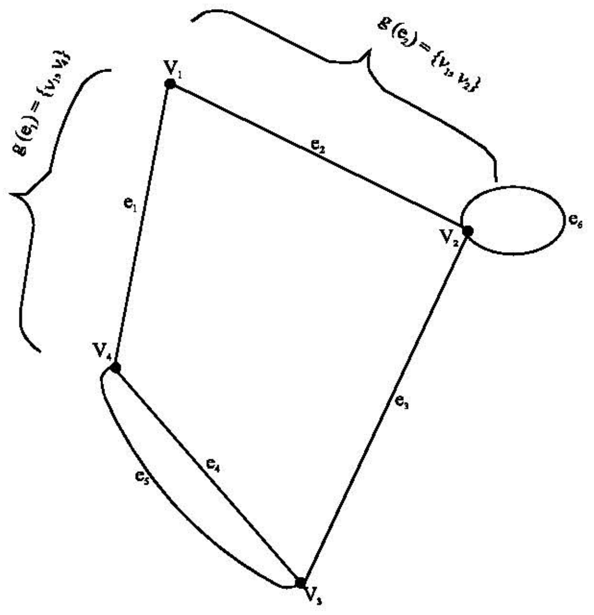


Fig. 2: Graph representation

Incidence: Consider Fig. 2 every edge is said to be incident with the vertices that it connects, so, edge e_2 is incident with vertices v_1 and v_2. Also, the vertex is said to be incident vertex if it belongs to one or more edges, so, v_1 is incident with edges e_1 and e_2.

Adjacent vertices: Vertices are said to be adjacent if the same edge connects them, for example, v_1 is adjacent to v_2 and v_4.

Adjacent edge: The edge is said to be adjacent to another if they have a common vertex, for example, e_3 and e_4 are adjacent edges.

Self-loop: An edge has the same end vertices, for example, e_6 in Fig. 2.

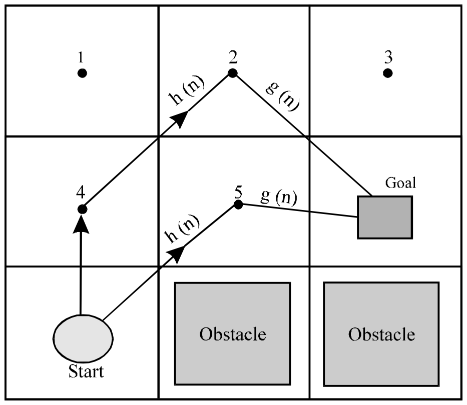


Fig. 3: Grid used by A*algorithm to reach the goal

Parallel edge: An edge shares its end vertices with another edge, e_5 in Fig. 2.

Graph formation: A graph is a set of points in the space it describes many different physical quantities versus others like pressure and temperature, strain and stress, etc., in this research we will take consider the graph which describes the path of a moving particle in a constrained space. One of the methods of generating such a graph is heuristic methods like A-star algorithm (Hart *et al*., 1968) (Ghafil, 2013).

A*search: It deals with the searching area as a grid collection and generates a path between two given points. Figure 3 illustrates how the algorithm chooses next grid point in the way to the goal. The algorithm starts from (start) point and selects the minimum cost grid point according to the equation:

$$f(n) = h(n)+g(n) \tag{1}$$

Where:

f(n)	=	The cost of the
(n) point, h(n)	=	The distance the successor and the current points
g(n)	=	The distance between the successor and goal points

Problem identification: Consider the problem of graph formation which is illustrated in Fig. 4. The search space is limited from 0-30 in both x and y-axis and A*should plan a graph between the start and task points avoiding the circular obstacles. It is obvious that the generated graph is zigzagged and containing many unnecessary segments. Thus, the next research is to find a solution to this problem and improve the quality of the heuristic path.

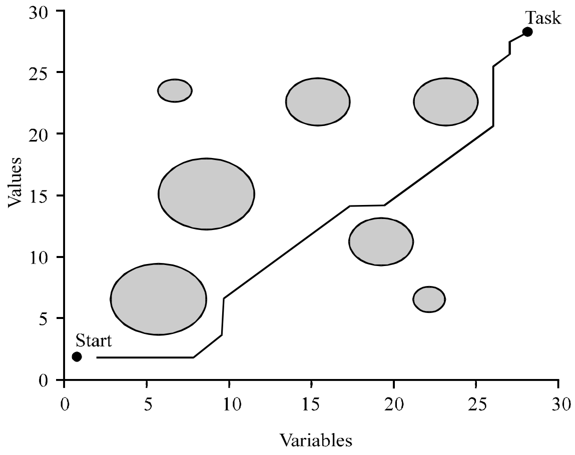


Fig. 4: Graph formation in constrained Cartesian space

Problem discretization: The proposed tools to improve the heuristic graph is based on the discretization of the continuous points on the graph into a combinatorial problem which is a graph theory problem. The procedure can be explained as follows.

Consider the unacceptable graph in Fig. 4, the graph consists of a series of points in the Cartesian coordinates and each point has a slop which is defined by:

$$m = \frac{y_{i+1}-y_i}{x_{i+1}-x_i} \tag{2}$$

I = 1, ..., n where n is the number of points consisting the graph and m is the slope of the graph at point i. For each point on the graph, calculate the slop using (Eq. 2) where i indicates the index of the current point and i+1 indicates the index of the next point on the graph.

The Heuristic graph is segmented straight lines, thus, whenever there is a change in the slop value on the graph, 9set a node or vertex as illustrated in Fig. 5. The vertices are numbered for illustration purpose and no matter if they are not numbered.

Add segments or call it an edge among all the vertices, it is very important that the connecting edge does not collide with any obstacle in the Cartesian space. The resulted object which is shown in Fig. 6 is a simple directed graph there is no parallel edges and no self-loops.

In this way, we had discretized the problem and mapped it into a combinatorial problem which is defined in graph theory literature as finding shortest path on a graph where starting point for our example is vertex (1) and ending point is vertex (11) Fig. 6. Many algorithms were dealt with finding the shortest path on a graph like Dijkstra algorithm (Dijkstra, 1959),

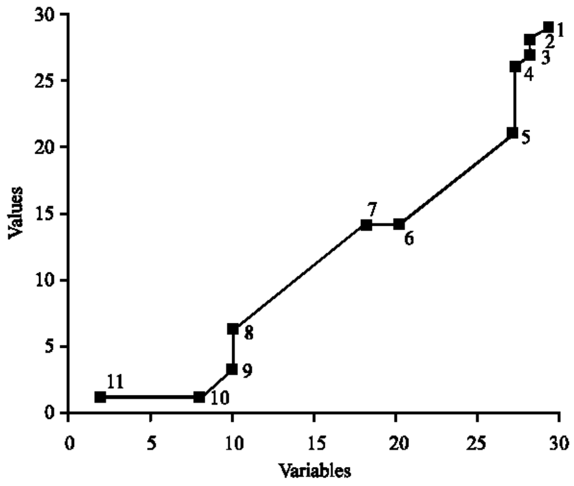


Fig. 5: Vertices on the graph represent the points of changing the slop

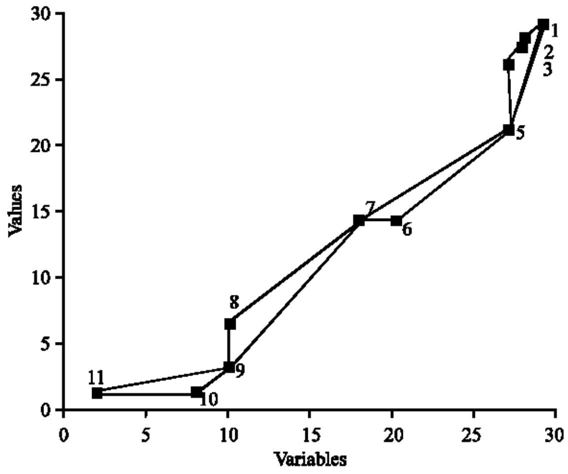


Fig. 6: The resultant simple graph after adding possible edges between vertices

Bellman-Ford's algorithm (Cherkassky *et al*., 1996), Floyd-Warshall algorithm (Swathika and Hemamalini, 2017), Johnson's algorithm (Pettie, 2002). In this research, the proposed optimization tools to find the shortest path on the graph are graph theory toolbox in MATLAB and ant colony optimization algorithm.

RESULTS AND DISCUSSION

Ant algorithm: Ant algorithms are designed for combinatorial problems; It is perfect for finding the shortest path on the graph this is ant algorithms are one of the solutions to improve the heuristic path problem shown in Fig. 6. Ant algorithm can be explained as follows: consider Fig. 7, assume a barrier put at the path

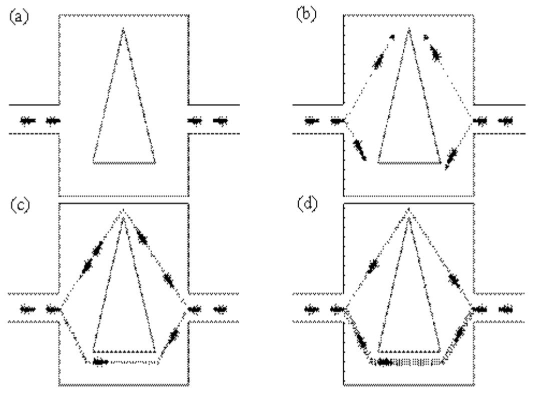


Fig. 7: How ants choose their path: a) Ants meet barrier on their way; b) Randomly some of the ants go to the left or right the barrier; Ants deposit pheromone while walking; c) Because of the evaporation, the long path will have less pheromone than short path and d) By time pheromone concentration which is represented by the dashed line will be high on the short path and this will be a temptation for ants to follow it

barrier they have to decide which path they have to take. Initially there is no information about the new path, so, some of the ants go to the left and some go to the right of the barrier and while they are moving, they deposit a pheromone. Ants who take the long path should spend more time than ants who take the short path and the evaporation rate of the pheromone is a function of time.

Consequently, the concentration of the pheromone on the short path will be greater than what is on the long path. This is a gradual process happening over time and for each time cycle, ants follow, probabilistically, the path of more pheromone. By the time, ants abandon the long path due to its poorness with pheromone.

Ant system: Ant system (Dorigo, 1992) or just as can be illustrated as follows: artificial ants make a tour over points in a graph and this tour is just segmented lines connecting the points. Each segment has two quantities the length of the segment (cost) and the pheromone, the later updated continuously during runtime. Ants choose the next point on the graph according to the probabilistic rule called state transition rule:

$$p_k = \begin{cases} \dfrac{[\tau(i,j)][\eta(i,j)]^{\beta}}{\sum_{u\in J_m(i)} [\tau(i,u)]\big[[\eta(i,u)]\big]} & \text{if } (i)\in J_m(i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

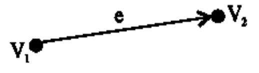


Fig. 8: Edge representation

Where:

τ = The pheromone on the segment which is connecting point i and j

η = The inverse of the distance between point i and j

$$\eta = \frac{1}{\delta(i,j)} \tag{4}$$

J_m (I) set of points that have to be visited by the ant m which is positioned at point i and u is the set of all points of the tour. β is weighting parameter control the importance of the cost with respect to the pheromone. When all ants generate their tour among points, the global update rule is implemented to update the pheromone values using the equations:

$$\tau(i,j) = (1-\alpha) * \tau(i,j) + \sum_{m=1}^{N} \Delta\tau_m(i,j) \tag{5}$$

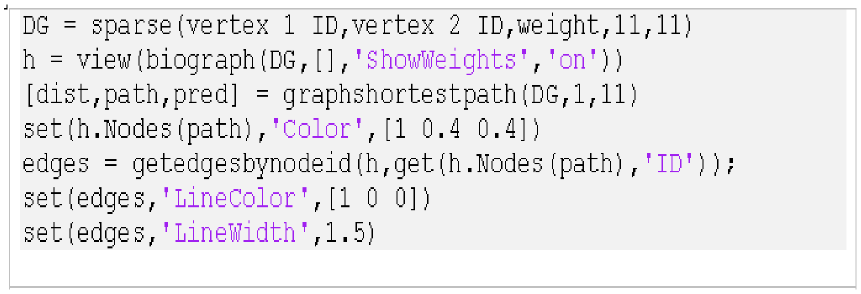
$$\Delta\tau_m(i,j) = \begin{cases} \dfrac{1}{L_m} \text{If } (i,j)\in u \\ 0 \qquad \text{otherwise} \end{cases} \tag{6}$$

where, $0<\alpha<1$is the pheromone decay parameter and L_m is the length of the path generated by the ant m. N is the total number of artificial ants. Glabowski *et al*. (2012) has solved the problem of the shortest path on a graph using ant colony optimization in details.

Graph theory toolbox: Graph object was introduced in MATLAB, since, R2015b and it contains many useful functions in graph theory calculations. Consider solving the problem in Fig. 6 where we have to find the shortest path from the start point of the graph (vertex 1) to the endpoint of the graph (vertex 11). The procedure can be described as follows.

Arrange the data of the directed graph in the form: It is important to order the edges correctly starting from the first edge and proceeded in the direction of the final edge on the graph also for a specific edge; The first vertex should be directed to the second vertex, for example, the edge in Fig. 8 should be represented as $e = (v_1, v_2)$:

- Make the sparse matrix using sparse ( ) function using the following syntax



```
DG = sparse(vertex 1 ID,vertex 2 ID,weight,11,11)
h = view(biograph(DG,[],'ShowWeights','on'))
[dist,path,pred] = graphshortestpath(DG,1,11)
set(h.Nodes(path),'Color',[1 0.4 0.4])
edges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
set(edges,'LineColor',[1 0 0])
set(edges,'LineWidth',1.5)
```

Fig. 9: Code snippet for the shortest path

Table 1: Directions of edge

Edges	Vertex 1	Vertex 2	The weight of edge (length)
Start edge	-	-	-
Intermediate edges	-	-	-
Final edge	-	-	-

Table 2: Coordinates of the ordered vertices

Vertex ID	x-axis	y-axis
1	29	29
2	28	28
3	28	27
4	27	26
5	27	21
6	20	14
7	18	14
8	10	6
9	10	3
10	8	1
11	2	1

- Sparse (vertex 1 column, vertex 2 column, weight column, n, n) where n is the maximum number in the column vertex 2
- Use graph shortest path ( ) to find the shortest path on the graph by the following syntax

[dist, path] = graph shortest path (sparse matrix, starting vertex, goal vertex) where dist is the total length of the improved graph and the path is a set of ordered points to represent the improved graph. It is worth to mention that graph shortest path is based on Dijkstra algorithm and achieves the best walk from vertex to vertex over the given segments, so, it is impossible to jump from the start point to the endpoints because in this case there is no walk.

Example: Consider improving the graph in Fig. 4. From step 1-3 in 5, we can determine the nodes (vertices) where the graph changes its slop. Figure 5 illustrates the results where there are 11 vertices and they should be arranged with their coordinates in a matrix in correct order from 1-11 as shown in Table 1 and 2.

From step (4) in 5, estimate all the possible edges that connect vertices in Table 1 without colliding with obstacles in the search space. There should be 19 ordered edges and directed from vertex 1 (start point) in the direction of the vertex 11 (goal point) and data should be held in the matrix as shown in Table 2. The length law can calculate the weight of the edge which is the length between its vertices and the results should be as in Fig. 6:

$$\text{Weight} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \quad (7)$$

Table 3: Weights of the edges in the directed graph

Edge ID	Vertex 1 ID	Vertex 2 ID	Weight
1	1	2	1.414
2	1	3	2.236
3	1	4	3.605
4	1	5	8.246
5	2	3	1.000
6	2	4	2.236
7	2	5	7.071
8	3	4	1.414
9	3	5	6.082
10	4	5	5.000
11	5	6	9.899
12	5	7	11.401
13	6	7	2.000
14	7	8	11.313
15	7	9	13.601
16	8	9	3.000
17	9	10	2.828

Calculate the sparse matrix using sparse ( ) by assigning second, third and fourth columns of the matrix which is shown in Table 2, n is the maximum number in the third column (vertex 2ID) and has a value of 11. The shortest path can be estimated by assigning the sparse matrix, start vertex and goal vertex to graph shortest path function, the following code snippet in Fig. 9 can be used to for the above mentioned procedure: the result should be as in Fig. 10. Figure 11 illustrates the original graph versus the improved one where the improved graph has a total length of 41.4956 while the total length of the original graph is 43.8701.

Practical application; Kinematic synthesis (Liu and McPhee, 2007): Selecting the best mechanism among enormous proposed choices is still an open field and all the achieved researches are limited and no one has introduced a complete formulation or systematic solution. The following procedure should be followed o conduct a kinematic mechanism synthesis (Table 3):

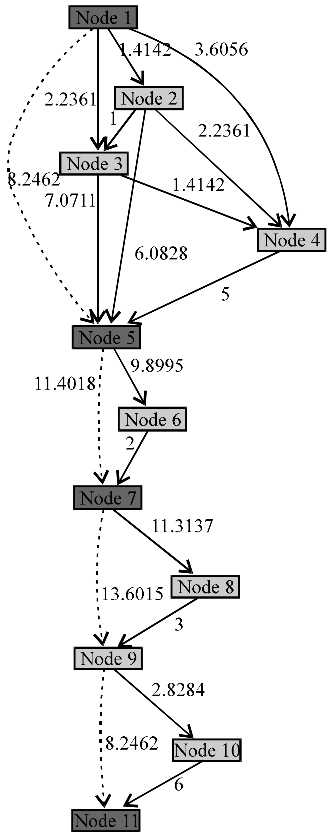


Fig. 10: Shortest path on the directed graph

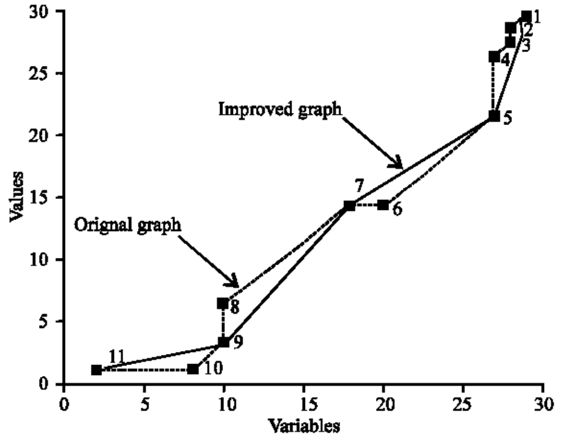


Fig. 11: Improved graph versus original graph

- Define the task of the mechanism
- Construct the type synthesis
- Evaluate the dimensional synthesis

For a specific task, the selection of the optimum topology for a mechanical structure is called type synthesis while estimating the optimum dimensions and

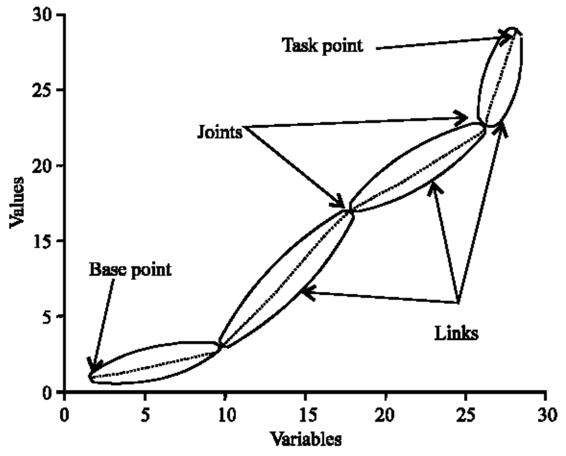


Fig. 12: The corresponding robot configuration to the improved heuristic graph

inertial properties for a specific topology is called dimensional synthesis. Selection the optimum topology is done according to the following requirements:

- The nature of the moving mechanism, for example, the rotated linkages are different from translated linkages
- The degree of freedom of the mechanism
- Number of the discrete outputs where each output has a single or multitask purposes
- The nature of the required task, a single task or multitask or generally how many tasks are required from the mechanism.
- The complexity of a specific task

Designers usually give equal attention to the constraint as well as the above mentioned requirements.

Proposed method: This research article is presenting a simple automated method to find the best kinematic synthesis at both type and dimensional levels. The abstract idea is that the path planned from a start point to the task point is equivalent to the kinematic synthesis for the robot arm at a specific configuration. Consider Fig. 11 where the hidden line is the improved heuristic graph, the method maps all the nodes on the graph to joints on the robot configuration and maps all segments on the graph to links on the robot. Figure 12 illustrates the corresponding robot type to the improved graph where the robot base frame is corresponding to the start point on the graph and the end-effector coordinate frame is corresponding to the task or goal point on the heuristic path. Also, it is worth to mention that this method works under limitations which are evaluated depending on the designer ambiguity:

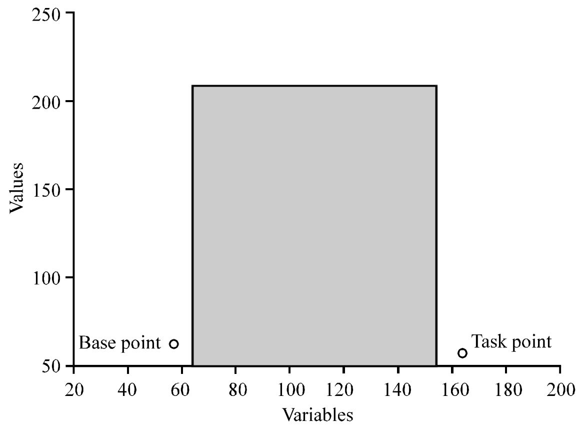


Fig. 13: Case study 1 problem where the rod block is an obstacle

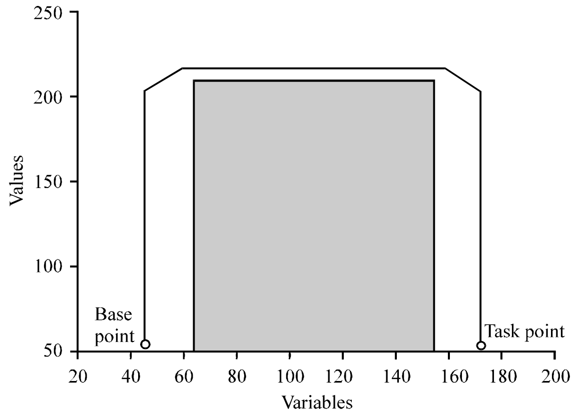


Fig. 14: The planned graph between the base and task points in the C-space

- The workspace should have a relatively low number of constraints
- The coordinates of the base and task points should be selected carefully
- The method provide a solution for a single task kinematic synthesis problem, thus, for multi tasks problem, greedy selection may be used to select the best type that meets all the requirements

Case study 1: Figure 13 explains the case study that will be used to test the proposed approach where according to the requirements of design the base and task points and constrained obstacles are defined.

Firstly the A-star algorithm makes its way starting from the base point to the task point avoiding obstacles in the configuration space denoted by the red block. The blue line in Fig. 14 expresses the path between the two points where the type of the robot should follow. It is obvious from Fig. 14 that this path does not represent optimally planned one.

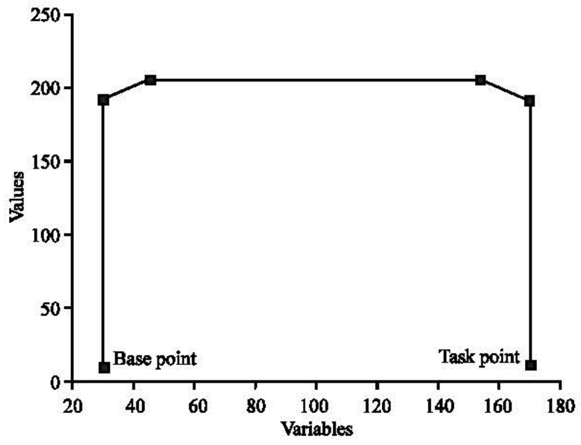


Fig. 15: setting nodes at places of slope directions

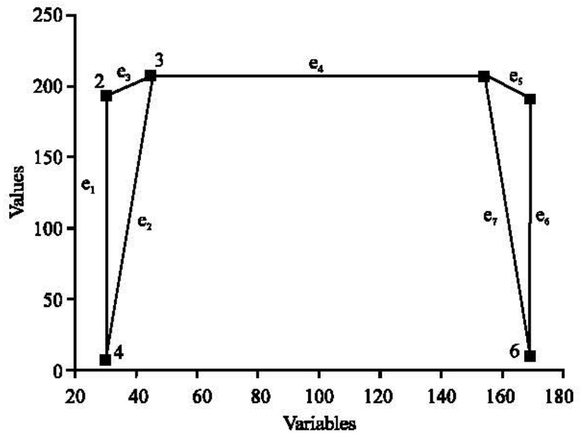


Fig. 16: Resulted directed graph

The next step is tracking the planned path seeking for these points at which the slope get changed and set nodes at these places. For the requirement, start and end points have considered nodes it is like the rotation DOF for the end-effector. Figure 15 shows the noded line where each black squares indicate a node.

The refinement step is needed where we have a set of ordered nods describing a directed graph with vertices and edges are illustrated in Fig. 16 which is an ordinary graph theory problem.

By calculating the weights of the edges which they are the distances between nodes and apply Dijkstra's algorithm for the shortest path which has embedded in MATLAB from Version R2015b. Figure 17 explains the shortest path on the graph denoted by the reddish line.

The resulted path is equivalent to the robot type and dimensionality. Figure 18 explains the simulation model of the solved robot using CAD file package 3DS MAX.

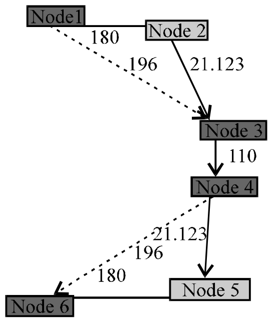


Fig. 17: The shortest path on the graph estimated by Dijkstra

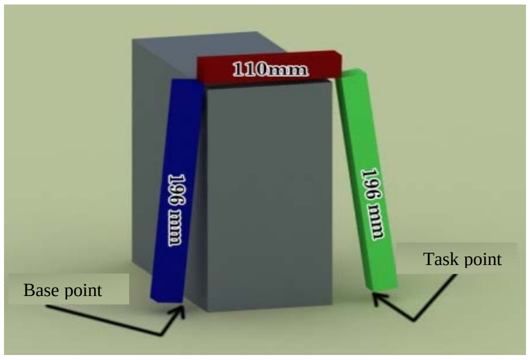


Fig. 18: CAD file for the robot type and dimensionality given by the proposed approach

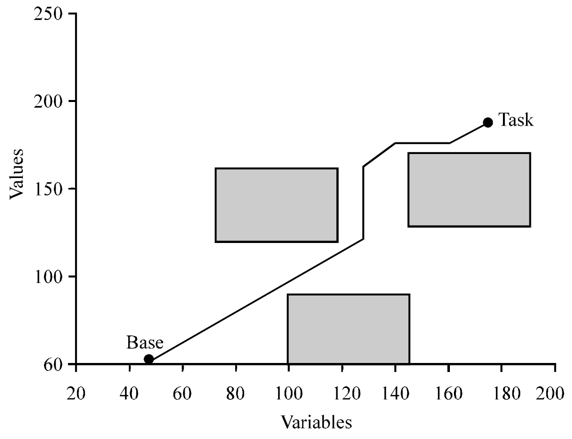


Fig. 19: The planned path for case study 2

Case study 2: A more complicated environment for the robot has suggested in Fig. 19 where the robot has to move among obstacles which they are denoted by red block. After defining the design requirements and condition, the solution can be seen as in Fig. 20-23. In this case study, one of the requirements is maximum link length should not exceed some limits otherwise the

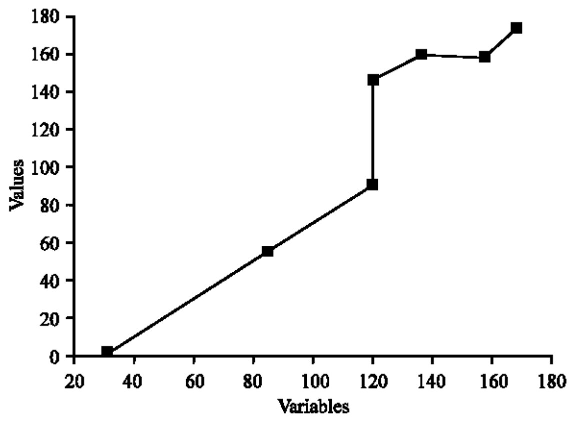


Fig. 20: Noded path

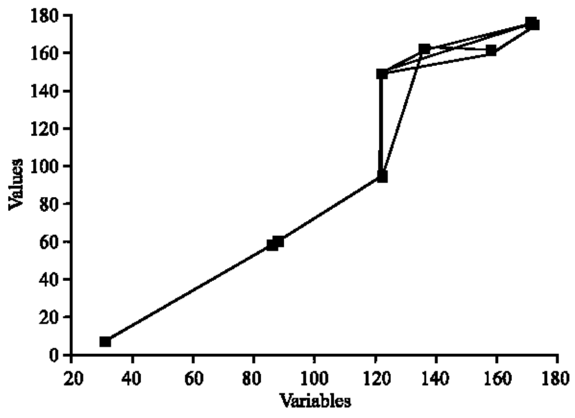


Fig. 21: Possible segments between nodes in the path of case 2

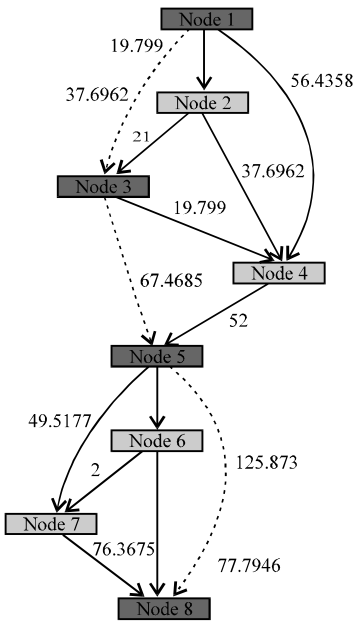


Fig. 22: Minimum segments needed for the task point

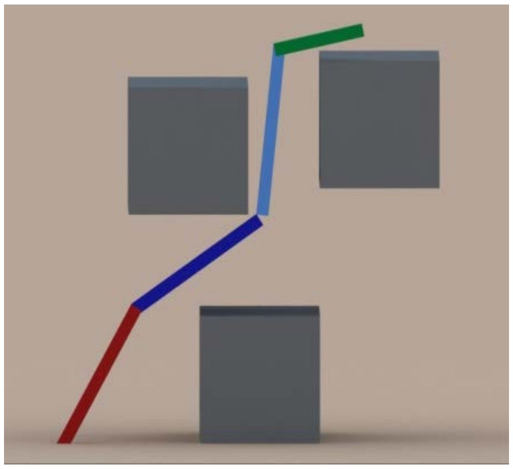


Fig. 23: CAD simulation for the resulted robot of case

automatic approach in this research has to break the link into two equal parts. Figure 22 shows the 3D simulation for the resulted robot for this situation

CONCLUSION

A new automated, graph theory based method for kinematic synthesis was presented in this research. The heuristic graph was generated using A star search algorithm between the start and task points. Two tools to improve the heuristic path was introduced which are ant colony system and graph theory toolbox. The improved graph is mapped to the corresponding kinematic synthesis where graph start point represents the base frame of the robot, the task point on the graph represents the end-effector position, nodes on the heuristic path represents the revolute joints on the robot configuration and line segments on the graph represent the links of the robot. The proposed method provides a systematic automated way to predict the optimum type and dimensional synthesis of a robot but still needs the designer ambiguity to get the feasible solution and the door is open for future research. Two examples were presented to find the optimum planar robots for a single task and the results are perfect.

REFERENCES

Bollobas, B., 2013. Modern Graph Theory. Springer, Berlin, Germany, ISBN:9781461206194, Pages: 394.

Cherkassky, B., V. Andrew V. Goldberg and T. Radzik, 1996. Shortest paths algorithms: Theory and experimental evaluation. Math. Prog., 73: 129-174.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik, 1: 269-271.

Dorigo, M., 1992. Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy.

Ghafil, H.N., 2013. Optimum path planning and performance analysis of a robot manipulator. MSc Thesis, Al-Nahrain University, Bagdad, Iraq.

Glabowski, M., B. Musznicki, P. Nowak and P. Zwierzykowski, 2012. Shortest path problem solving based on ant colony optimization metaheuristic. Image Process. Commun., 17: 7-17.

Hart, P.E., N.J. Nilsson and B. Raphael, 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Cybernet., 4: 100-107.

Liu, Y. and J. McPhee, 2007. Automated kinematic synthesis of planar mechanisms with revolute joints. Mech. Based Des. Struct. Mach., 35: 405-445.

Oliva, J.C. and E.D. Goodman, 2010. Simultaneous type and dimensional synthesis of planar 1DOF mechanisms using evolutionary search and convertible agents (DETC2009-86722). J. Mech. Rob., 2: 1-9.

Pettie, S., 2002. A faster all-pairs shortest path algorithm for real-weighted sparse graphs. Proceedings of the International Colloquium on Automata, Languages and Programming, July 8-13, 2002, Springer, Heidelberg, Germany, ISBN:978-3-540-43864-9, pp: 85-97.

Pucheta, M.A. and A. Cardona, 2013. Topological and dimensional synthesis of planar linkages for multiple kinematic tasks. Multibody Syst. Dyn., 29: 189-211.

Swathika, G.O.V. and S. Hemamalini, 2017. Prims Aided Floyd Warshall Algorithm for Shortest Path Identification in Microgrid. In: Emerging Trends in Electrical, Communications and Information Technologies, Attele, K., A. Kumar, V. Sankar, N. Rao and T. Sarma (Eds.). Springer, Berlin, Germany, pp: 283-291.

Thiele, R., 2005. The Mathematics and Science of Leonhard Euler (1707-1783). In: Mathematics and the Historians Craft, Van Brummelen, G., O.K. May and M. Kinyon (Eds.). Springer, New York, USA., ISBN:9780387252841, pp: 81-140.

West, D.B., 2001. Introduction to Graph Theory. 2nd Edn., Pearson Education, Inc., London.