Graph Theory and Its Derivations in Robotics Engineering

Jose Hosea 13516027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516027@std.stei.itb.ac.id

Abstract — Robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. Robotics deals with the design, construction, operation, and use of robots, as well as computer systems for their control, sensory feedback, and information processing.

One of the branch is discrete mathematics, which include logics, mathematics induction, Boolean algebra, recursion, number theory, combinatorics, graph theory, tree, and algorithm complexity.

While there are many disciplinary branch of engineering and science involved in robotics engineering, this paper will only be focusing on how learning discrete mathematics, mainly graph theory and its derivations, is as important as other disciplinary branch for robotics engineers.

***Keywords* — Robotics Engineering, Graph Theory, Tree, Applications.**

I. Introduction

The concept of creating machines that can operate autonomously dates back to classical times, but research into the functionality and potential uses of robots did not grow substantially until the 20th century. The concept itself has become a mainstream objective for scientist around the world for ages. The concept would later become the basic principles of robotics.

The word itself was derived from the word robot, *robota* in Slavic, which means forced labour. And so it became clear, that the purpose of creating these 'machines' were to automate works so they become easier.

These days, robots are being used as substitutes that can help or replicate humans' work. They are used to perform jobs more cheaply, more accurately and more reliably, than humans.

In exploring more ways of creating robots, engineers need to understand some basic concepts of science, i.e. branch of computer science, that could become useful when designing or manufacturing robots. They are also important for robotics engineers who seek to optimize their work, or expand their creations' field of work.

This paper will introduce the definition of robot, what does a robotics engineer do, and lastly how graph theory and its derivations can be applied in this branch of engineering.

II. Theories

2.1 Robotics Engineering

2.1.1 Robots

A **robot** is a machine—especially one programmable by a computer— capable of carrying out a complex series of actions automatically. Robots can be guided by an external control device or the control may be embedded within. Robots may be constructed to take on human form but most robots are machines designed to perform a task with no regard to how they look.

Robots have replaced humans in performing repetitive and dangerous tasks which humans prefer not to do, or are unable to do because of size limitations, or which take place in extreme environments such as outer space or the bottom of the sea. There are concerns about the increasing use of robots and their role in society. Robots are blamed for rising technological unemployment as they replace workers in increasing numbers of functions. The use of robots in military combat raises ethical concerns. The possibilities of robot autonomy and potential repercussions have been addressed in fiction and may be a realistic concern in the future.

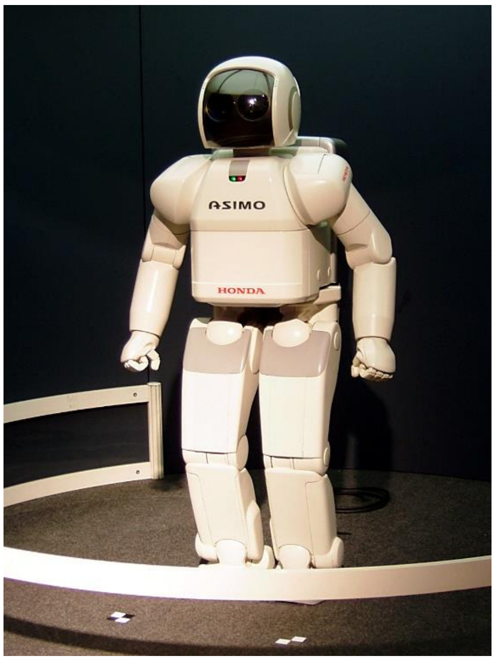


Figure 2.1. A Fully Functional Robot : HONDA ASIMO

There are a lot different types of robots that can be made, but the basic similarities of their construction is the most important part in creating a robot, with the rest of the process after it are left to engineers' creativities and goals.

They are used in many different environments and for many different uses, although being very diverse in application and form they all share three basic similarities when it comes to their construction:

1. Robots all have some kind of **mechanical construction**, a frame, form or shape designed to achieve a particular task. For example, a robot designed to travel across heavy dirt or mud, might use caterpillar tracks. The mechanical aspect is mostly the creator's solution to completing the assigned task and dealing with the physics of the environment around it. Form follows function.

2. Robots have **electrical components** which power and control the machinery. For example, the robot with caterpillar tracks would need some kind of power to move the tracker treads. That power comes in the form of electricity, which will have to travel through a wire and originate from a battery, a basic electrical circuit. Even petrol powered machines that get their power mainly from petrol still require an electric current to start the combustion process which is why most petrol powered machines like cars, have batteries. The electrical aspect of robots is used for movement (through motors), sensing (where electrical signals are used to measure things like heat, sound, position, and energy status) and operation (robots need some level of electrical energy supplied to their motors and sensors in order to activate and perform basic operations)

3. All robots contain some level of **computer programming** code. A program is how a robot decides when or how to do something. In the caterpillar track example, a robot that needs to move across a muddy road may have the correct mechanical construction and receive the correct amount of power from its battery, but would not go anywhere without a program telling it to move. Programs are the core essence of a robot, it could have excellent mechanical and electrical construction, but if its program is poorly constructed its performance will be very poor (or it may not perform at all). There are three different types of robotic programs: remote control, artificial intelligence and hybrid. A robot with remote control programing has a preexisting set of commands that it will only perform if and when it receives a signal from a control source, typically a human being with a remote control. It is perhaps more appropriate to view devices controlled primarily by human commands as falling in the discipline of automation rather than robotics. Robots that use artificial intelligence interact with their environment on their own without a control source, and can determine reactions to objects and problems they encounter using their preexisting programming. Hybrid is a form of programming that incorporates both AI and RC functions.

2.1.2 Robotics Engineer

A robotics engineer is a behind-the-scenes designer, who is responsible for creating robots and robotic systems that are able to perform duties that humans are either unable or prefer not to complete. Through their creations, a robotics engineer helps to make jobs safer, easier, and more efficient, particularly in the manufacturing industry.

A robotics engineer will spend the majority of their time designing the plans needed to build robots. They also design the processes necessary for the robot to run correctly. Some of the engineers are also responsible for designing the machines that actually assemble the robots. Once the design phase has been completed, only then do they move toward assembling the unit.

This type of engineer is responsible for creating several different types of robots that are used to complete a variety of different tasks. Prior to a robot being constructed, the engineer will have first researched and determined exactly what the robot will be used for, and the manner in which it will accomplish its goal. And it is likely that the building process will take a great deal of time. Robots are highly technical and difficult to create, and the task can be very tricky. For this reason it's not uncommon for a robotics engineer to only work on a handful of projects throughout his or her entire career. Typically, the engineer's duties include:

- Building, configuring, and testing robots.
- Designing software systems to control their robotic systems, such as those robots used for manufacturing.
- Designing automated robotic systems that are used to increase the production and precision levels within a specific industry.
- Analyzing and evaluating the prototypes and robotic systems they have created. This is generally a never-ending task, since technology is constantly changing and advancing.
- Reviewing and approving cost estimates and design calculations.
- Serving as technical support for the robotic systems they have created.
- Teaching plans paths to robots.
- Performing research into the design, operation and performance of robotic mechanism components or systems.

2.2 Graph Theory

2.2.1 Definitions

Conceptually, a *graph* is formed by *vertices* and *edges* connecting the vertices.

Formally, a graph denoted as G = (V, E) is a pair of sets where V is the *set of vertices* and E is the *set of edges*, formed by pairs of vertices. E is a *multiset*, in other words, its elements can occur more than once so that every element has a *multiplicity*. Often, we label the vertices with letters (for example: a, b, c or v1, v2, v3 or numbers.

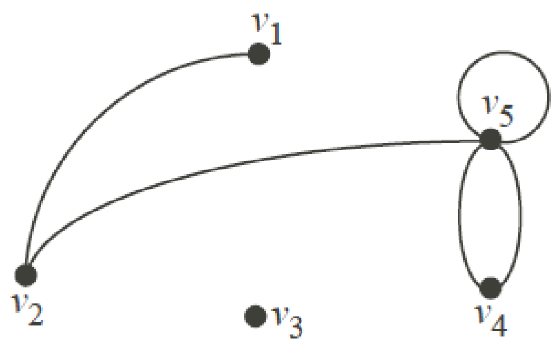


Figure 2.2. A Graph

Frequently used terminologies to define the characteristics of a graph:

1. The two vertices u and v are *end vertices* of the edge (u, v).
2. Edges that have the same end vertices are *parallel*.
3. An edge of the form (v, v) is a *loop*.
4. A graph is *simple* if it has no parallel edges or loops.
5. Variations of *simple graph* :
   - A *complete graph* is a *simple graph* in which each of its vertex has an edge to every other vertex, often denoted as K_n .
   - A *circle graph* is *a simple graph* in which each of its vertex has the *degree* of two.
   - A *regular graph* is a *simple graph* in which each of its vertex has the same *degree* with every other vertex.
   - A *bipartite graph* or *bigraph* is a graph in which its set of vertices decomposed into two disjoint sets such that no two vertices within the same set are *adjacent*.
6. A graph with no edges (i.e. E is empty) is *empty*.
7. A graph with no vertices (i.e. V and E are empty) is a *null graph*.
8. A graph with only one vertex is *trivial*.
9. Edges are *adjacent* if they share a common end vertex.
10. Two vertices u and v are *adjacent* if they are connected by an edge, in other words, (u, v) is an edge.
11. The *degree* of the vertex v, written as d(v), is the number of edges with v as an end vertex.
12. For *directed graphs*, *degree* of the vertex is divided into:
    - *Indegree* is the number of head ends adjacent to a vertex.
    - *Outdegree* is the number of tail ends adjacent to a vertex.
13. By convention, we count a *loop* twice and *parallel edges* contribute separately.
14. A *pendant vertex* is a vertex whose degree is 1.
15. An edge that has a pendant vertex as an end vertex is a *pendant edge*.
16. An *isolated vertex* is a vertex whose degree is 0.
17. A *path* is the distance or edges passed between start vertex to end vertex.
18. A *cycle or circuit* is a *path* in which the start and the end vertex are the same.
19. A graph is *connected* if and only if every pairs of vertices have at least a *path* to each other, otherwise it is called a *disconnected graph*.
20. For directed graphs, a graph is *strongly connected* if and only if every pairs of vertices have at least a *path* to each other, otherwise it is called a *weakly connected graph*.
21. A *subgraph* is a graph in which its vertices and edges are subsets of another graph.
22. A *spanning subgraph* is a subgraph which contains all of the vertices of another graph.
23. A *cut-set or bridges* is subset of edges of a connected graph that if were deleted from the graph would make the graph *disconnected*. So, cut-set always resulting a pair of *connected subgraph*.
24. A *weighted graph* is a graph in which its edges have values.

Accordingly, graphs can be divided into categories based on its number of vertices, and its edges' orientation

a) Number of vertices
   1. *Limited Graph* is a graph with finite number of vertices.
   2. *Unlimited Graph* is a graph with infinite number of vertices.

b) Edges' orientation
   1. *Undirected Graph* is a graph in which edges have no orientation.
   2. *Directed Graph or Digraph* is a graph in which edges have orientations.
   3. *Mixed Graph* is a graph in which some edges may be directed and some may be undirected.

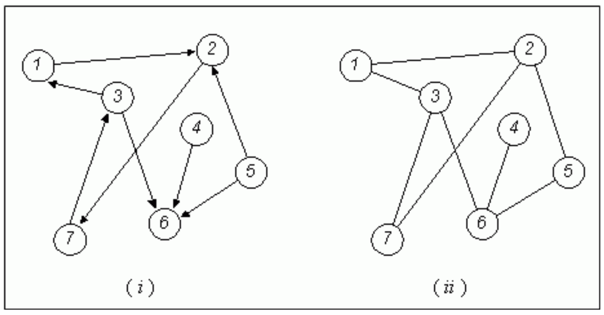


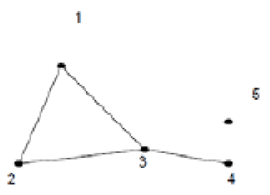
Figure 2.3. Directed Graph(i) and Undirected Graph(ii)

2.2.2 Graph Representations

There are ways to represent graphs in computer memory, such as:

1. *Adjacency Matrix*

   This representation contains the adjacency between pairs of edges in a square matrix with the size of number of the graph's vertices. M_{ij} contains 1 if the edges are adjacent, 0 if the edges are not.



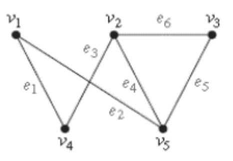
$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 2.4. Adjacency Matrix

For graphs that contain parallel edges, the matrix no longer contains 0 or 1, containing counts of edges associated with vertices *i* and *j* .

For weighted graphs, cells of the matrix represent the value of the edges. If the vertices *i* and *j* are not connected then M_{ij} can be set to infinite.

2. *Incidence Matrix*
   This representation contains the connection between edges and vertices in a matrix with the size of number of graph's vertices times the number of its edges. M_{ij} contains 1 if the vertex is an end vertex to the edge, 0 if not.



$$\begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline v_1 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 1 & 1 & 0 & 1 \\ v_3 & 0 & 0 & 0 & 0 & 1 & 1 \\ v_4 & 1 & 0 & 1 & 0 & 0 & 0 \\ v_5 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Figure 2.5. Incidence Matrix

3. *Adjacency List*
   This representation contains the adjacency between pairs of edges in a list.

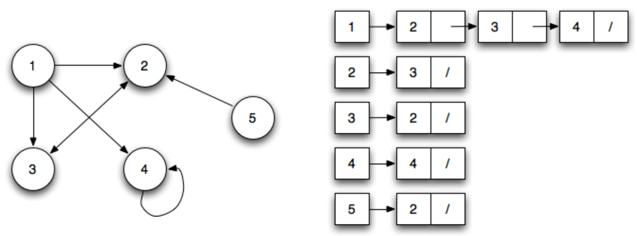


Figure 2.6. Adjacency List

2.2.3 Tree

A *tree* is an undirected graph in which any two vertices are connected by exactly one path. It is a unique derivation of graph.

A graph must have some characteristics to be called a tree:

a) Every pair of its edges is connected with exactly one path.
b) It has *n-1* edges, *n* is number of vertices, and it is a connected graph.
c) It does not have a circuit, and adding an edge to it will only make one circuit.
d) All of its edges are bridges.

A group of disjoint tree is called a *forest*.

There are variations of tree, such as:

a) *Rooted Tree* is a tree in which one vertex has been designated as the *root*. A tree with no designated *root* is called a *free tree*.
   There are some derivations of *rooted tree* like *ordered tree* and *n-ary tree*.
b) *Spanning Tree* is subgraph and a tree which were made from an undirected connected graph, which include all of the graph's vertices and with minimum possible number of edges. A graph can have more than *one spanning tree*. A derivation of the *spanning tree* called *minimum spanning tree* is a *spanning tree* made from a weighted graph, which contains the minimum weight of all variants of the graph's *spanning trees.*

III. THE APPLICATION OF GRAPH THEORY

3.1 Robotics Construct

In creating a robot, an engineer must have a goal as to answer the question of 'why create it', and by doing so, the step of designing the robot will become focused mainly to the goal.

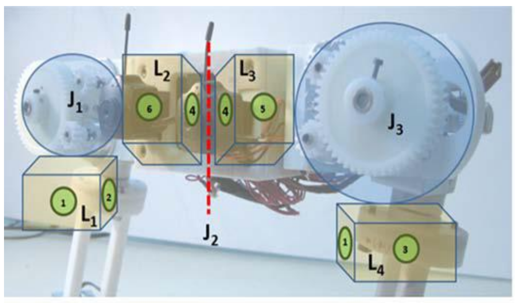
By defining a goal, one can determine which feature the robot must have, therefore one can hold what science technique need to be applied in the process of creating the robot.

These are some examples of how graph theory and its derivations can be applied in the making of a robot.

3.1.1 Modular Robot

A modular system can be represented by its corresponding graph, showing the relationship existing among different modules. The application of graph theory to modular robots requires the use of specific graph elements. This degree of specialization is achieved by labeling graph vertices and edges in such a way that there are as many labels as different types of robot links and joints.

Figure 3.1. Modular Robot



All this information is set forth in the Assembly Incidence Matrix, AIM, so that it can be easily included in the robot's algorithms. The AIM is a *N+1* times *M+1* matrix with *N* vertices and *M* edges. The extra column *M+1* indicates the link type, while the extra row *N+1* shows the joint type. See the example below.

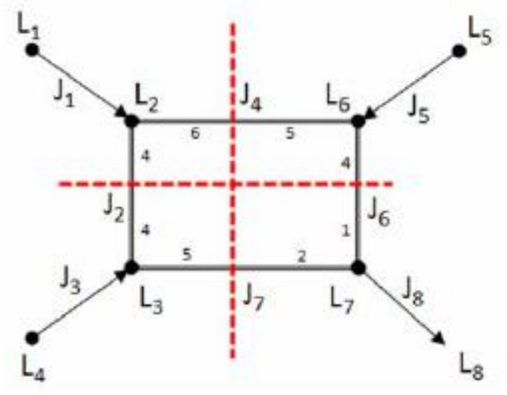


Figure 3.2. Graph of Robot’s Links and Joints

$$
AIM = \begin{array}{ccccccccc|c}
J_1 & J_2 & J_3 & J_4 & J_5 & J_6 & J_7 & J_8 & & \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & L_1 \\
1 & 4 & 0 & 6 & 0 & 0 & 0 & 0 & p & L_2 \\
0 & 4 & 1 & 0 & 0 & 0 & 5 & 0 & p & L_3 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & p & L_4 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & p & L_5 \\
0 & 0 & 0 & 5 & 1 & 4 & 0 & 0 & p & L_6 \\
0 & 0 & 0 & 0 & 0 & 1 & 2 & -1 & p & L_7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & p & L_8 \\
s & c & s & c & s & c & c & s & 0 &
\end{array}
$$

Figure 3.3. Assembly Incidence Matrix

By analyzing the graph (links and joints structure), it can be determined the complexity for the robot to move from one point to another.

3.1.2 Decision Tree

A decision tree is a tree which is made to graph what possible choices can be made in the given circumstances.

While it may sounds too basic and not important, this method helps engineers to analyze problems and ways to resolve them with considerations of robot’s form, position or even its flaws.

For example, soccer robotics engineers would need robots that could perceive its condition like, is it standing? Or is it not? Is it holding the ball? Is it seeing enemy? Or is it an ally? To answer all that questions, one could graph decision trees when designing the mind of high intelligence robot. See the given decision tree.

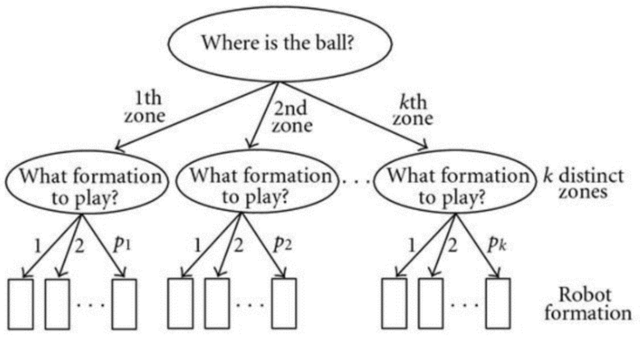


Figure 3.4. Decision Tree for Soccer Robot

By creating the decision tree, one can set robot’s way of solving a given situation. The leaves of the tree are representing its action, responding to its condition or environment and, once more, is up to the engineers who made it.

3.2 Robotic Mapping

Robotic mapping is a useful feature that can be installed into an autonomous robot so it is able to construct (or use) a map or floor plan and to localize itself in it.

The internal representation of the map can be metric or topological:

- The metric framework is the most common for humans and considers a two-dimensional space in which it places the objects. The objects are placed with precise coordinates. This representation is very useful, but is sensitive to noise and it is difficult to calculate the distances precisely.
- The topological framework only considers places and relations between them. Often, the distances between places are stored. The map is then a graph, in which the vertices correspond to places and edges correspond to the paths.

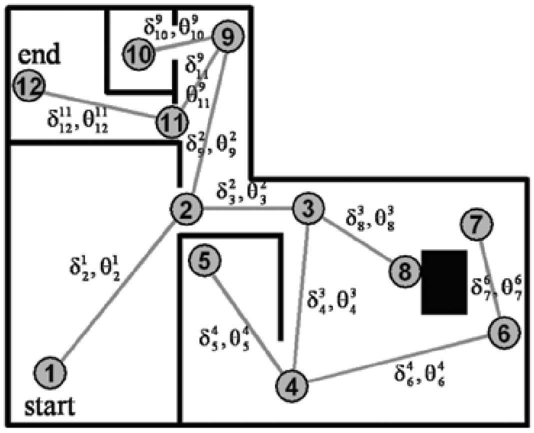


Figure 3.5. Topological Representation of a Map using *weighted graph*

3.3 Multi-Robot System

A multi-robot (MR) system consists of a group of robots that act autonomously and is designed to solve problems that would be impossible for a single robot. In single robot systems, there is a huge responsibility put onto one robot which, in turn causes a single point of failure.

MR systems address this issue by creating redundancies and expanding responsibilities, therefore providing a decreased system fail rate. As the field of robotics progresses, there is a need to focus research efforts more towards distributed-type robotic systems.

One of the advantages of having a MR system maintaining formation is that all robots can depart from and arrive at a waypoint at the same time. See the graph below.

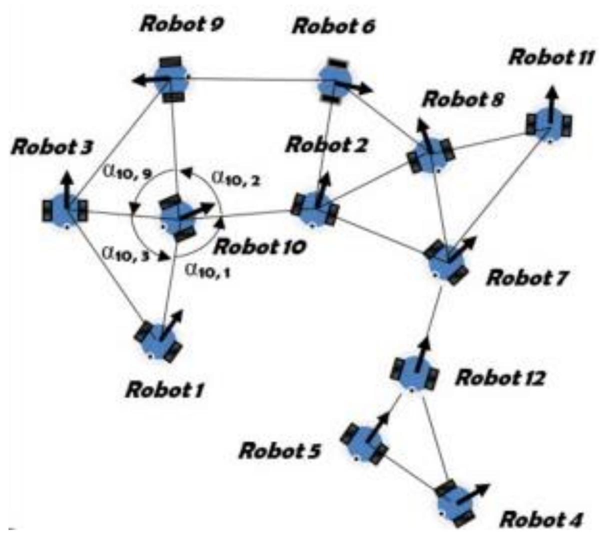


Figure 3.6. Multi-Robot System

Maintaining the formations of robot can be done via wireless communication, by sending each robots position to a controller robot or a module, so it can localize itself and other robots(see *Robot Mapping*) and then it can decide what path or action other robots should take.

IV. Conclusion

By learning discrete mathematics, especially graph theory, robotics engineers can improve their creations. This is proven by how it can be applied in many ways in a robot as well as in a multi-robot system.

It can be used to help the engineers in mapping the links and joints of a robot or other components that frequently used in it, graphs its movement, and control its cable management or even its body placement. It can also be applied in a more advanced problem like robotic mapping. By combining those techniques, one can solve the 'path planning' for the robot.

Accordingly, learning graph derivations, such as tree, can help the engineers to implant their ideas, their solutions to problems into their creations. By implanting decision tree to a robot programming, it can mirror its creator's intentions accurately.

It also can be used in designing a multi-robot system where the problem needs a simultaneous solutions to be applied.

V. Acknowledgment

A special note of thanks to Dr. Judhi Santoso M.Sc. and Dr. Ir. Rinaldi Munir, MT., the most inspiring lecturers at Bandung Institute of Technology for this interesting assignment that has broaden my knowledge on robotics engineering.

I would also like to thank Wikipedia and other sources of my inspirations that helped me get this paper done.

References

[1] J. Baca, A. Yerpes, M. Ferre, J. A. Escalera, R. Aracil, "Modelling of Modular Robot Configurations Using Graph Theory," Universidad Politécnica de Madrid, August 2008.
[2] Anonymous, "The Interplay between Mathematics and Robotics," National Science Foundation Arlington, Virginia, May 2000.
[3] Dr. Rinaldi Munir, "Diktat Kuliah: Matematika Diskrit," Departemen Teknik Informatika, Institut Teknologi Bandung, August 2003.
[4] C. Barca, A. Sekercioglu, A. Ford, "Controlling Formations of Robots with Graph Theory Department of Electrical and Computer Systems Engineering, Monash University, Melbourne, Australia, June 2012.
[5] https://www.sokanu.com/careers/robotics-engineer/
[6] https://www.quora.com/How-important-are-graph-algorithms-in-robotic-AI
[7] https://math.stackexchange.com/questions/1177313/a-discrete-mathematics-formulation-problem-in-a-robotic-system
[8] https://math.stackexchange.com/questions/1177313/a-discrete-mathematics-formulation-problem-in-a-robotic-system

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

[illegible]

Jose Hosea 13516027