**CAB203 Project Report**

**William Ee, N10494448**

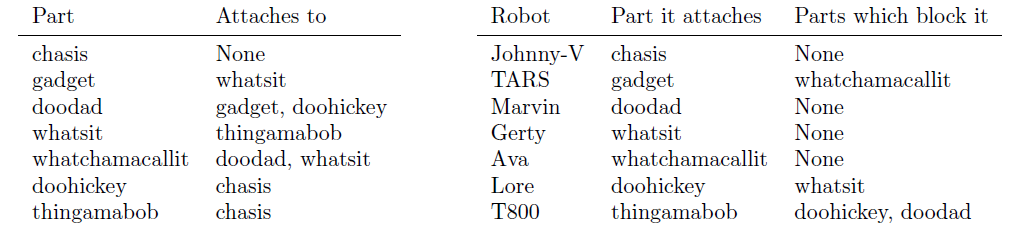
**June 4, 2021**

**Introduction**

A company is building a factory for fabricating widgets which are built using many different parts. The company plans to achieve this through the use of an assembly line with robots on the side to attach parts. This is done sequentially such that each robot is adds a unique part onto a partially completed widget and places it back on the conveyor belt where the next robot will retrieve the widget and add its part onto the widget and continues until the widget has all parts attached on it. This results in an assembly line which starts off with a widget without any parts and ends with a fully completed widget with all parts.

However, the company’s engineers are struggling with figuring out how to position the robots to achieve a working assembly line. This is because certain parts of the widget are required to attach to other parts, thus requiring other parts to be already placed. Additionally, certain parts also block certain robots from placing their part, thus, that robot would need to place their part before the blocking part. A description of the parts and robots has been provided by the engineer as seen in figure 1.

Given the list of parts and robots, the engineer has requested for an arrangement of the robots on the assembly line such that the widgets are properly assembled with all parts attached or determining that it is not possible to do.



**Figure 1 List of Parts and Robots**

**Instance Model**

An instance model would be a set containing the robots, which part it attaches, which respective part/parts its part attaches to and the part/parts which blocks the robot. The instance model essentially contains information in the summarised list of parts and robots below in figure 2.

So let be the set describing the robots and parts where is robot, is the part it attaches, is the set of part/parts attaches to, and is the set of part/parts which block. This can be modelled as:

{(Johnny-V, chasis, {}, {}), (TARS, gadget, {whatsit}, {whatchamacallit}),…}

The instance is hence modelled by () and given a specific instance in figure 2 the instance is thus:

({(Johnny-V, chasis, {}, {}), (TARS, gadget, {whatsit}, {whatchamacallit}),…})

|  |  |  |  |
| --- | --- | --- | --- |
| **Robot ()** | **Part ()** | **Attaches to ()** | **Blocked by ()** |
| Johnny-V | chasis | None | None |
| TARS | gadget | whatsit | whatchamacallit |
| Marvin | doodad | gadget, doohickey | None |
| Gerty | whatsit | thingamabob | None |
| Ava | whatchamacallit | doodad, whatsit | None |
| Lore | doohickey | chasis | whatsit |
| T800 | thingamabob | chasis | doohickey, doodad |

**Figure 2 Summarised List of Robots and Parts**

**Solution Model**

In order to produce widgets properly using an assembly line, we need to find a sequential arrangement for the robots in the assembly line to allow them to attach their respective parts to necessary parts without being blocked. Hence, the solution model will be an ordered arrangement of robots , where comes first in the assembly line and places its parts and passes it to which attaches its part and so on until finally reaching robot which places its part, thus completing the widget.

**Problem Model**

Let the instance be given. The assembly line of robots can be modelled as a directed graph

Given the instance model, the set of vertices are the robots in the assembly line in That is:

For edges , include an edge connecting robot in to robot in , if ’s part attaches to ’s part . Additionally, include an edge if is blocked by ’s part. Essentially, for every edge robot must come before in the assembly line. is given by:

Note that as this is a directed graph, symmetric cases are not included. Additionally, as all robots must take a partially completed widget and pass it on to another robot until it reaches the end of the assembly, this graph is connected, that is all robots should have at least one edge which connects it to or from another robot.

Deriving a set of robots and edges which describe which robots must come before others , we need to find an ordered arrangement of robots as described in the solution model. Hence the solution is a topological ordering of the instance graph . Note if the instance graph is cyclic and not acyclic, then no topological ordering can be found.

**Solution Method**

After calculating an instance graph from instance we find a topological ordering which is a total ordering on the vertices of a directed acyclic graph . To find a topological ordering we can use Khan’s Algorithm which is a recursive formulation.

To implement Khan’s algorithm, we create a list let us say , which holds an order of robots whereby robots in come before in the assembly line and there is no specific order for robots in a particular . The set of robots keeps track of the robots that have not been added to yet, so whenever a robot is added to , it is removed from .

So firstly, we start this recursive definition by establishing an empty list of vertices and a full set of robots . That is:

Then we define a recursive case for a particular call of the definition , assuming has been defined. We do this by finding all vertices that have no edges coming in, so in this scenario we are finding the robots that have no requirement for any other robot to come before it. So, for the robots in the set of robots not yet added into the list , we add it into in the list if for all robots in the same set of robots , there are no edges . This is defined by:

Next, within the same recursive case , for the remaining robots yet to be added for recursive case, we take the set of robots that have yet to be added in the previous recursive case and remove the robots that were just added to the list . Thus, updating the set of robots that have not been added to the list, defined by:

Then continue defining recursive cases until all robots have been added to the list , such that . At this point, the list will contain a topological ordering of robots for the assembly line, such that for the list of robots whenever , .

We may also break out of the recursive cases if during the recursive cases other than the base case, no robot could be found without an edge going towards it. That is: and . This indicates a cyclic graph, thus breaking the asymmetry required for a topological ordering. In this case we determine that it is not possible to arrange the robots.

In summary, the solution can be found from an instance as follows:

1. Calculate an instance graph from instance .
2. Calculate and recursively using Khan’s Algorithm thus finding a topological ordering given by the list