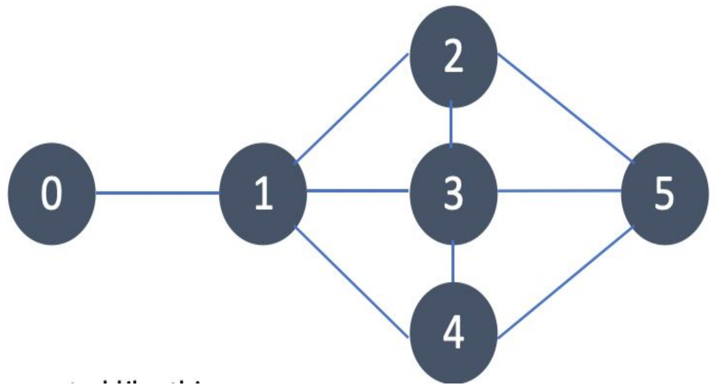
Snowplow Simulation

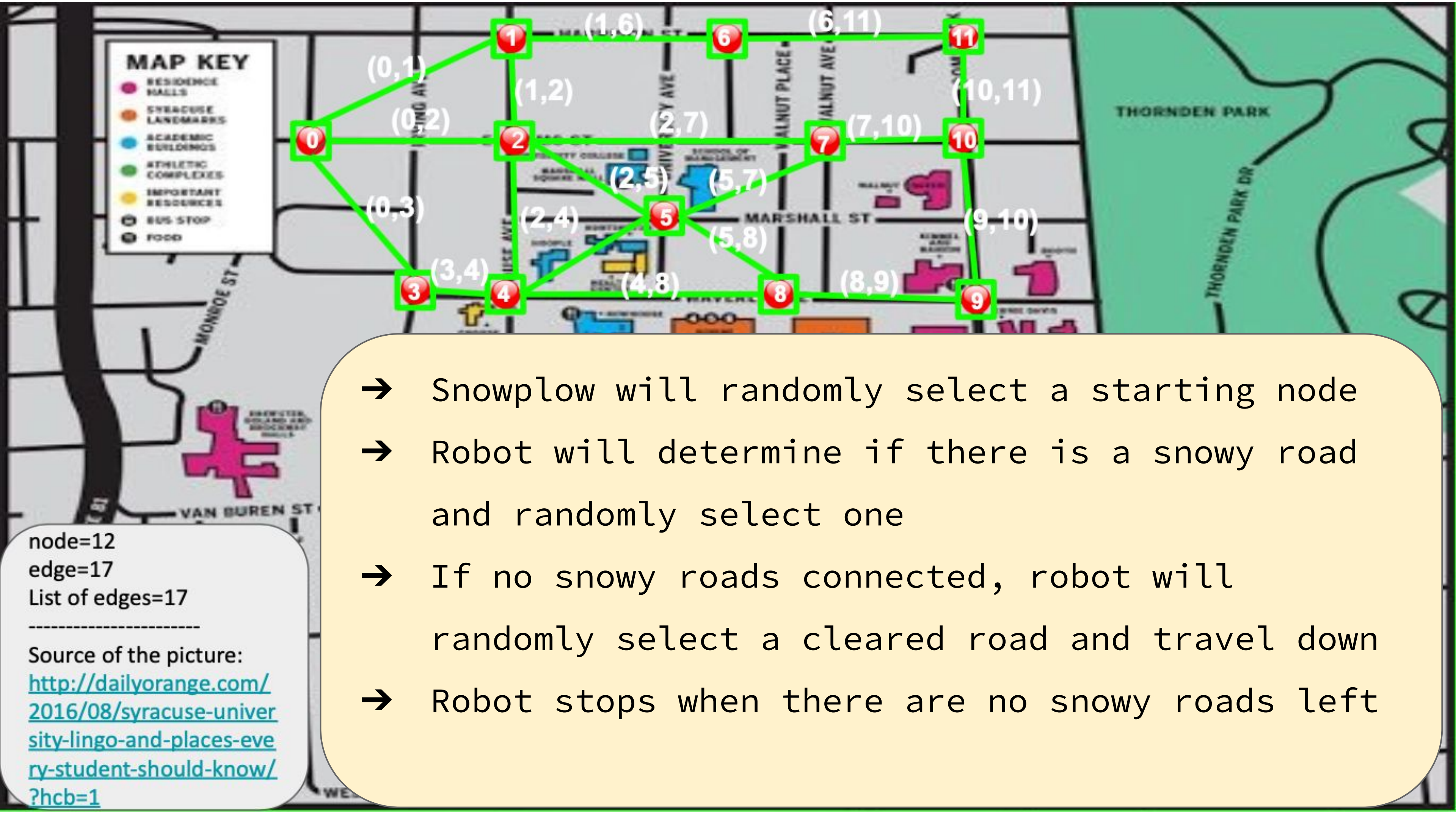
By: Thomas, Jiayao, Yitao, Kezia

Overview



Sample Network

- ❖ Our plan of attack:
  - ➢ Create simulation program
  - ➢ Run 10,000 trials
  - ➢ Create histogram of average time and number of backtracks



- ➔ Snowplow will randomly select a starting node
- ➔ Robot will determine if there is a snowy road and randomly select one
- ➔ If no snowy roads connected, robot will randomly select a cleared road and travel down
- ➔ Robot stops when there are no snowy roads left

node=12
edge=17
List of edges=17

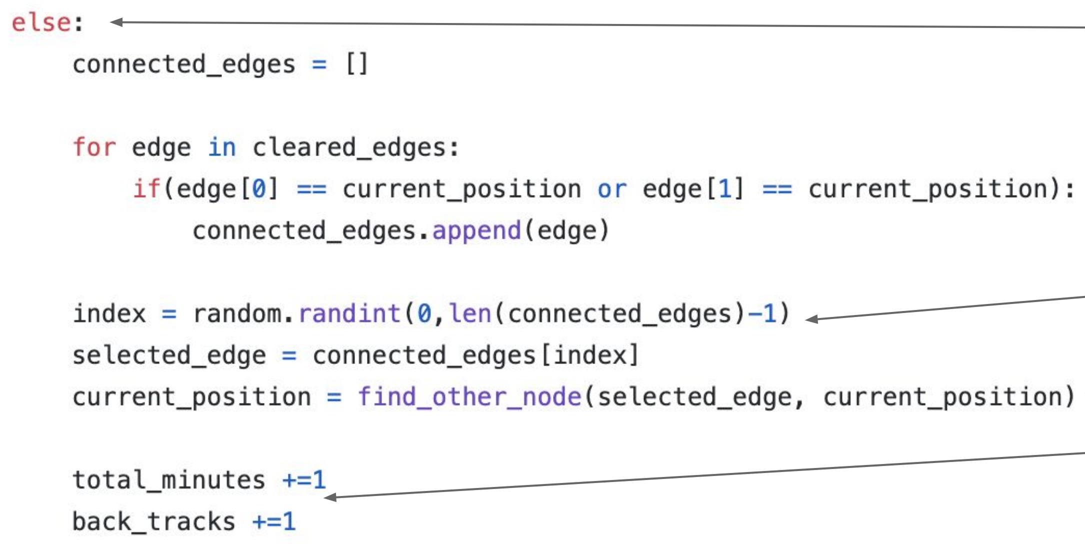
Source of the picture:
http://dailyorange.com/2016/08/syracuse-university-lingo-and-places-every-student-should-know/?hcb=1

If statement …Clearing connected edges

```python
def snowplow_program(num_nodes,num_edges,snowy_edges):
    current_position = random.randint(0,num_nodes-1)
    total_minutes = 0
    back_tracks = 0
    cleared_edges = []

    while(len(snowy_edges) > 0):
        current_snowy_edges = get_snowy_edges(current_position,snowy_edges)

        num_cse = len(current_snowy_edges)

        if(num_cse > 0):
            index = random.randint(0,num_cse-1)
            selected_edge = current_snowy_edges[index]
            cleared_edges.append(selected_edge)
            current_position = find_other_node(selected_edge, current_position)
            se_idx = snowy_edges.index(selected_edge)
            snowy_edges = snowy_edges[:se_idx] + snowy_edges[se_idx+1:]
            total_minutes += 1
```

- ❏ Robot won’t stop until all snowy edges are cleared
- ❏ From the current node, generates a list of snowy edges
- ❏ If there are any snowy edges the program will:
  - ❏ Randomly select an edge
  - ❏ Add edge to cleared edges
  - ❏ Calculate new node
  - ❏ Remove edge from snowy edges
  - ❏ Increment time by one

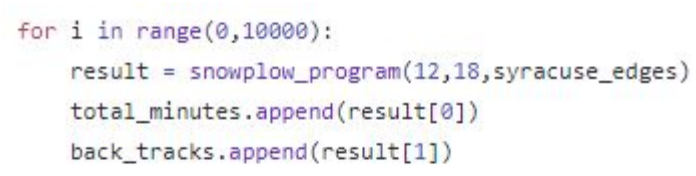
ELSE STATEMENT…WHAT IF ALL THE EDGES ARE CLEARED?



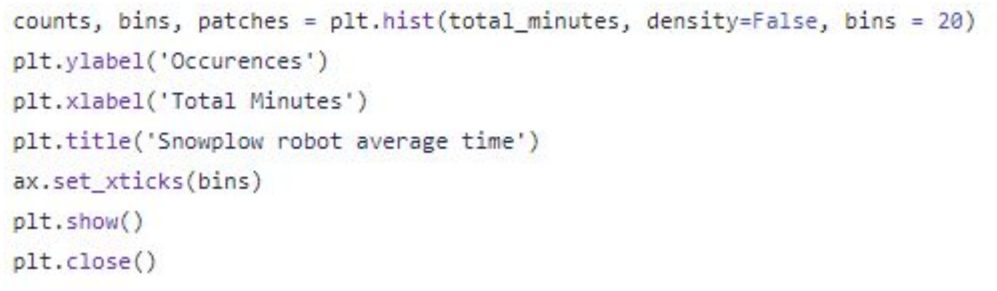
```
else:
    connected_edges = []

    for edge in cleared_edges:
        if(edge[0] == current_position or edge[1] == current_position):
            connected_edges.append(edge)

    index = random.randint(0,len(connected_edges)-1)
    selected_edge = connected_edges[index]
    current_position = find_other_node(selected_edge, current_position)

    total_minutes +=1
    back_tracks +=1
```

- If there is no snowy edges the program will:
  - Generate a list of connected edges
  - Randomly select an edge
  - Calculate new node
  - Increment time and backtracks by one

Simulation Plotting



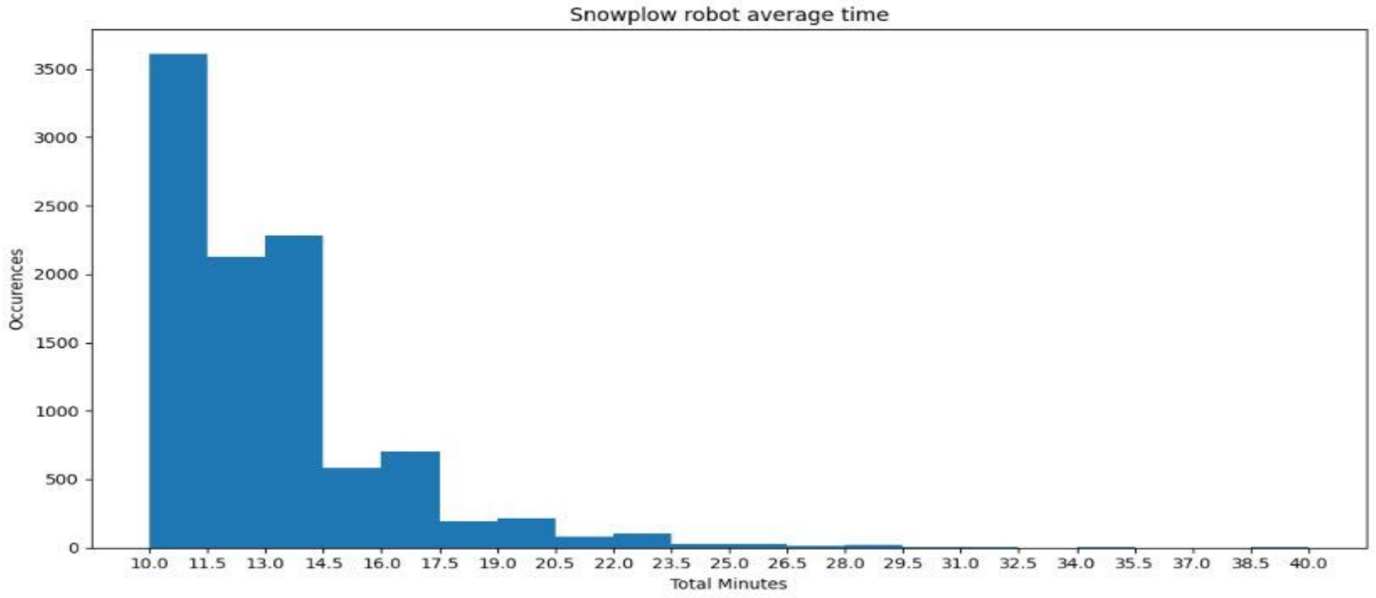
```
for i in range(0,10000):
    result = snowplow_program(12,18,syracuse_edges)
    total_minutes.append(result[0])
    back_tracks.append(result[1])
```



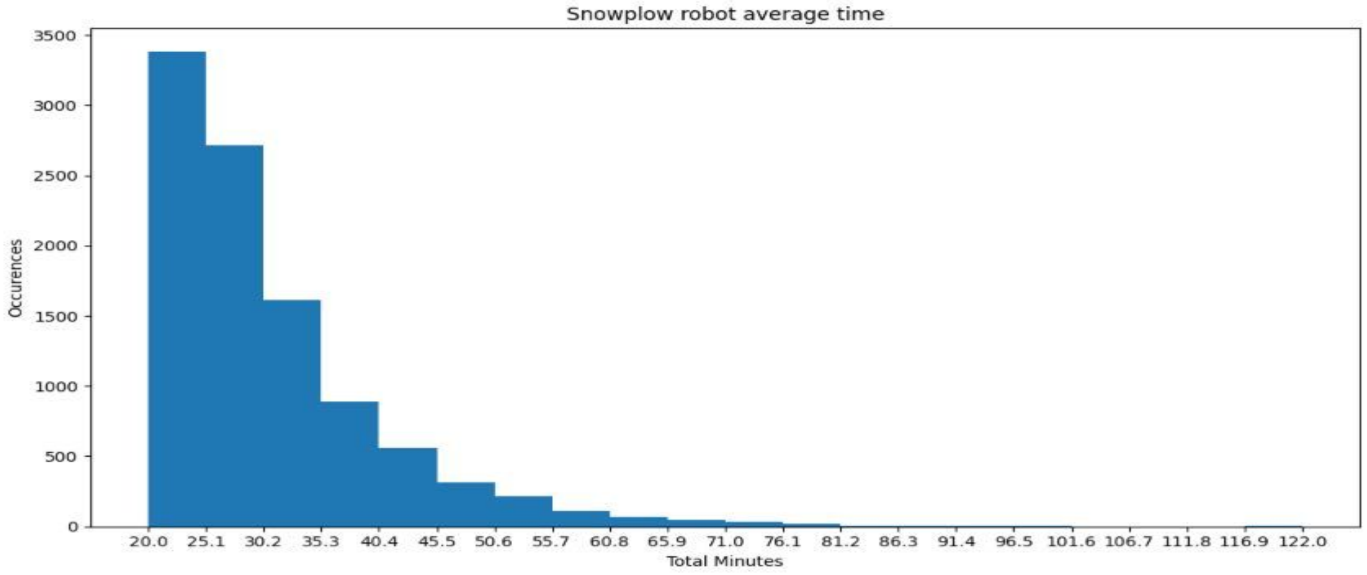
```
counts, bins, patches = plt.hist(total_minutes, density=False, bins = 20)
plt.ylabel('Occurences')
plt.xlabel('Total Minutes')
plt.title('Snowplow robot average time')
ax.set_xticks(bins)
plt.show()
plt.close()
```

- ❑ Run 10,000 trials and record minutes and backtracks
- ❑ Graph histogram of total minutes

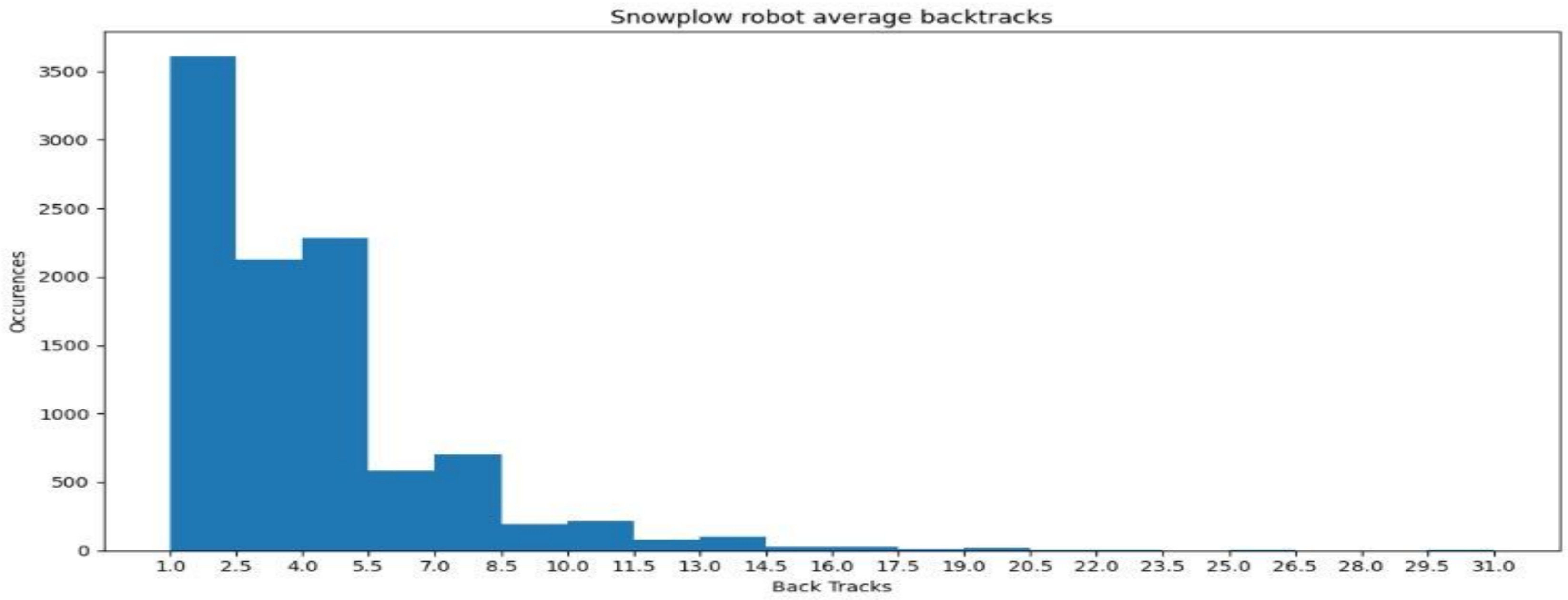
Sample Network Average Time



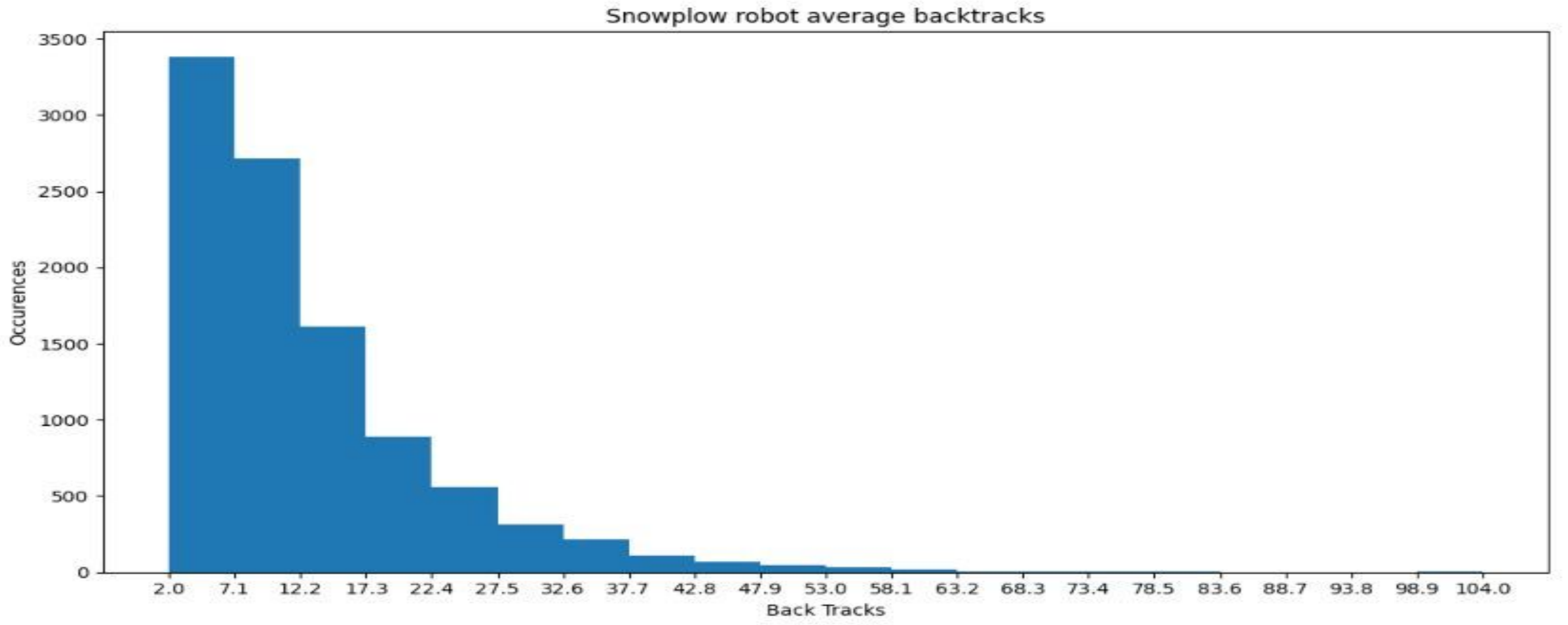
Syracuse network Average Time



Sample Network Average BAcktracks



Syracuse Network Average Backtracks


Snowplow robot average backtracks
Occurences
Back Tracks