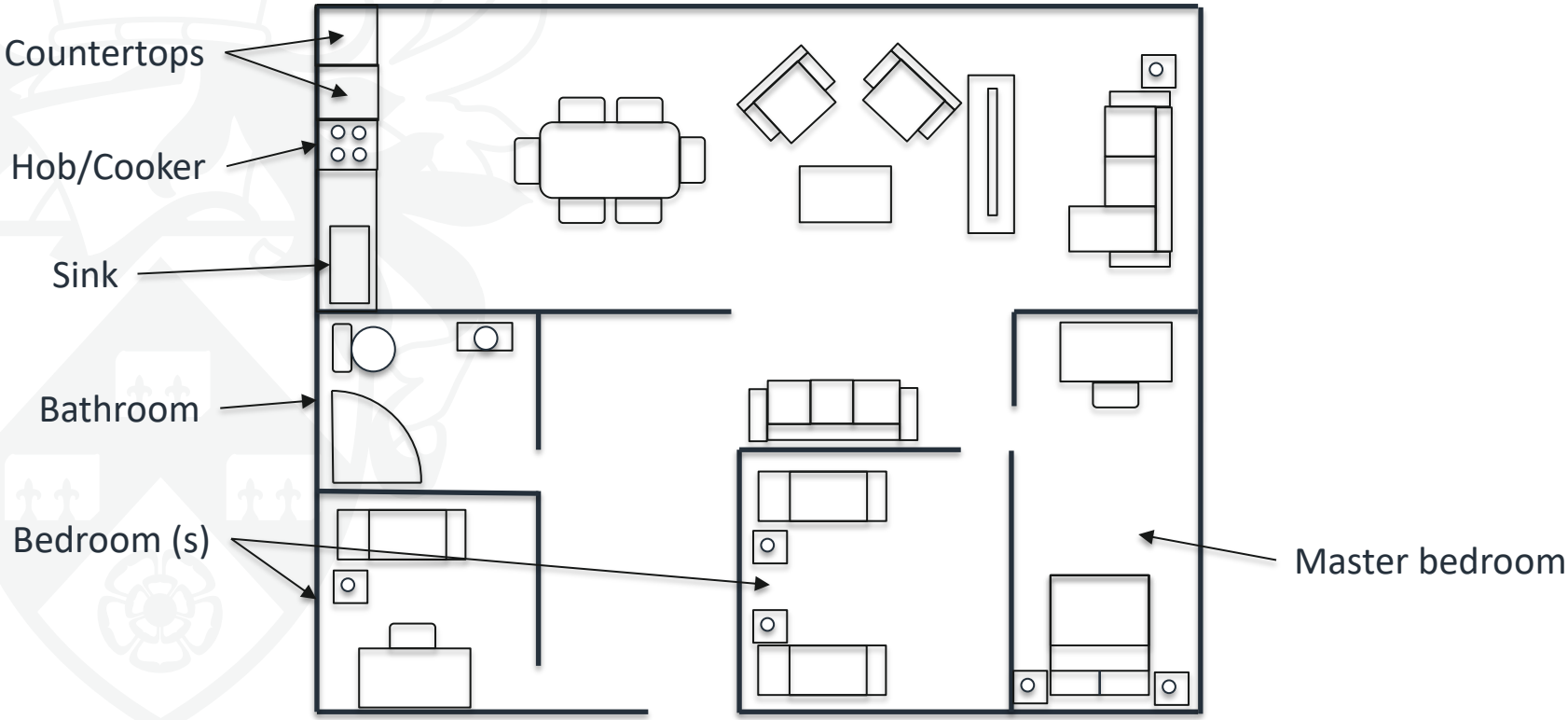
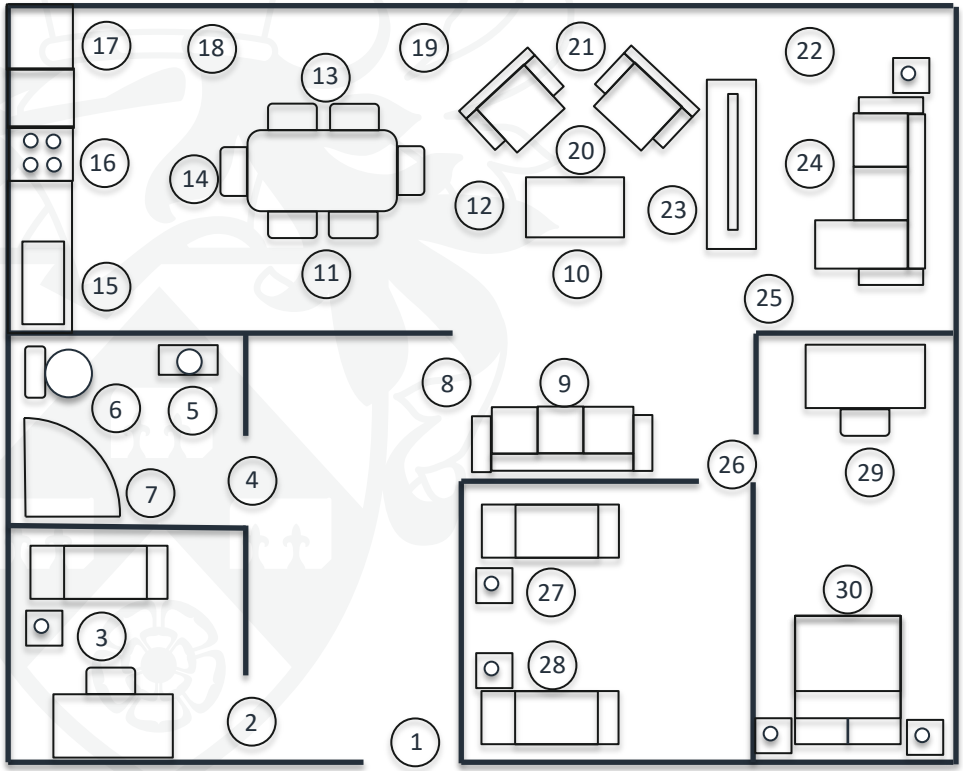
University of York Update

December 2021

Environment Modelling



Environment Modelling



Create a **graph** using nodes for key locations within the environment, with edges connecting neighbouring nodes

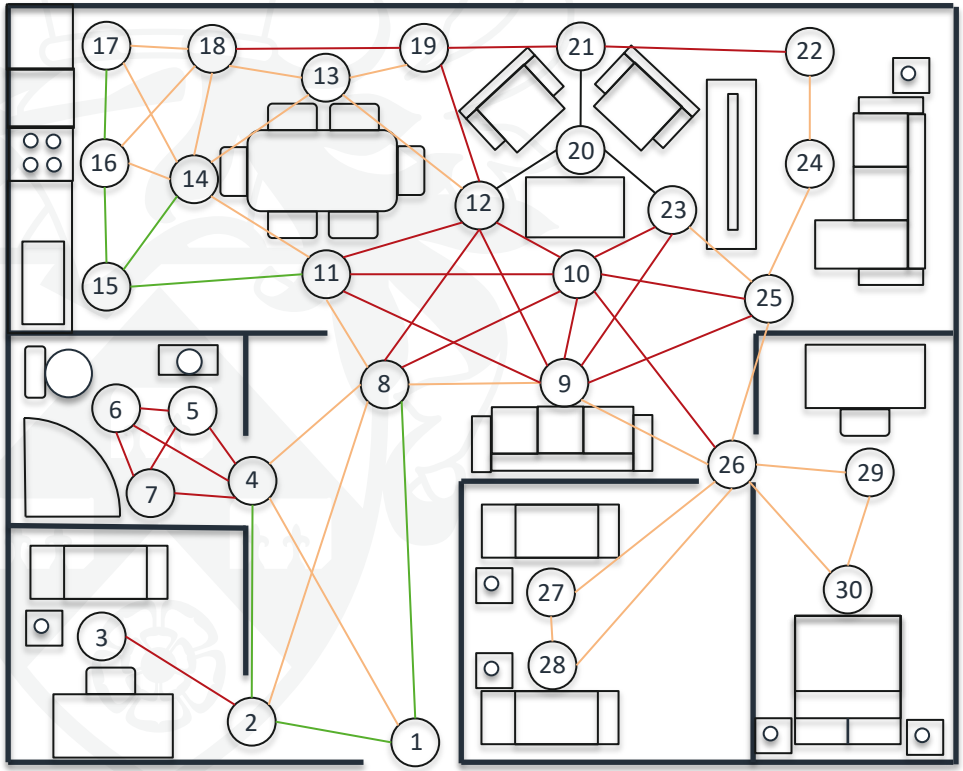
Resolution of the graph can be increased or decreased

Edges connecting the nodes are assigned a linear distance and risk factor

The risk factor corresponds to the probability of successfully moving along the edge

$$P_{success} + P_{return} + P_{fail} = 1$$

Environment Modelling



Each edge is weighted:

- Green **low** risk
- Orange **medium** risk
- Red **high** risk
- Back **very high** risk

Risk categories are assigned based on clutter and also whether they intersect one or more edge

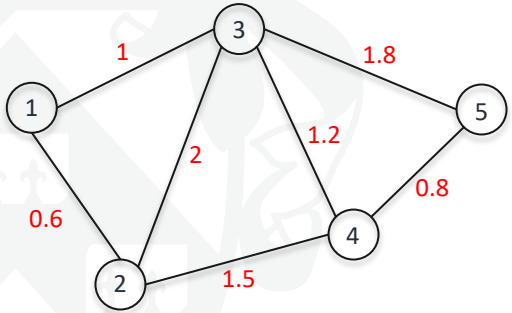
Each risk category has **three sub-categories** (low, medium, high)

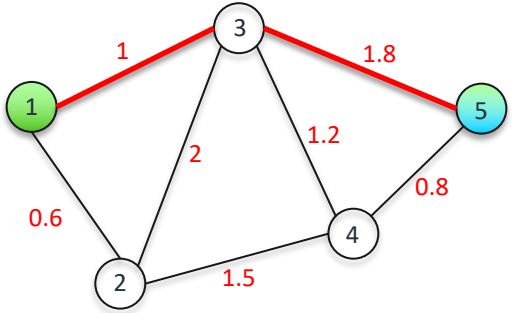
- ”low-low”
- ”medium-low”
- “high-medium”, etc

Path Finding

Dijkstra's Algorithm

- Shortest distance path in a graph



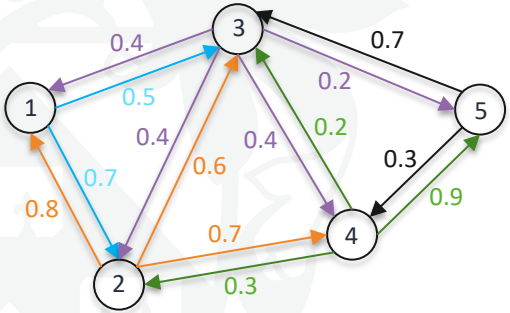


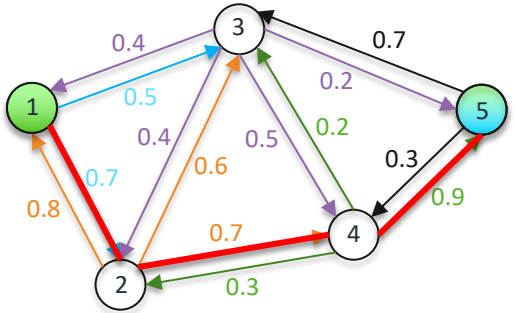
Shortest Distance Path:

Path Finding

Dijkstra's Algorithm

- Adjust the parameters of the algorithm to **maximise** a parameter, such as probability
- Rather than a distance value, each edge has a probability of success (P_s) and failure ($1 - P_s$)





Highest Probability Path: 1 → 2 → 4 → 5

$0.7 \times 0.7 \times 0.9 \approx 44\%$

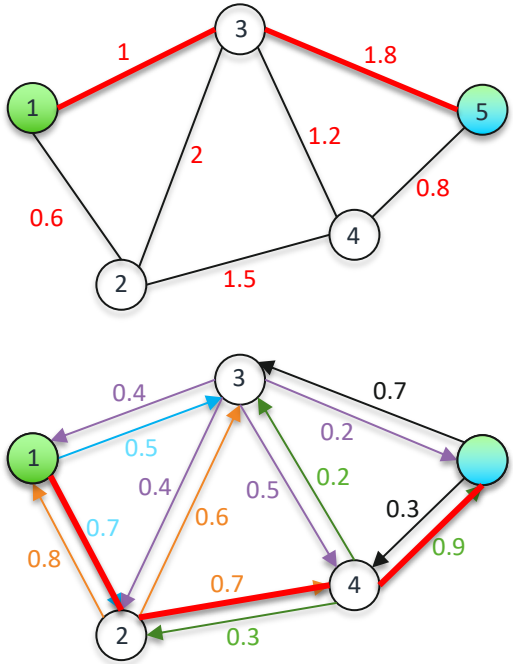
Shortest Distance Path: 1 → 3 → 5

$0.5 \times 0.2 = 10\%$

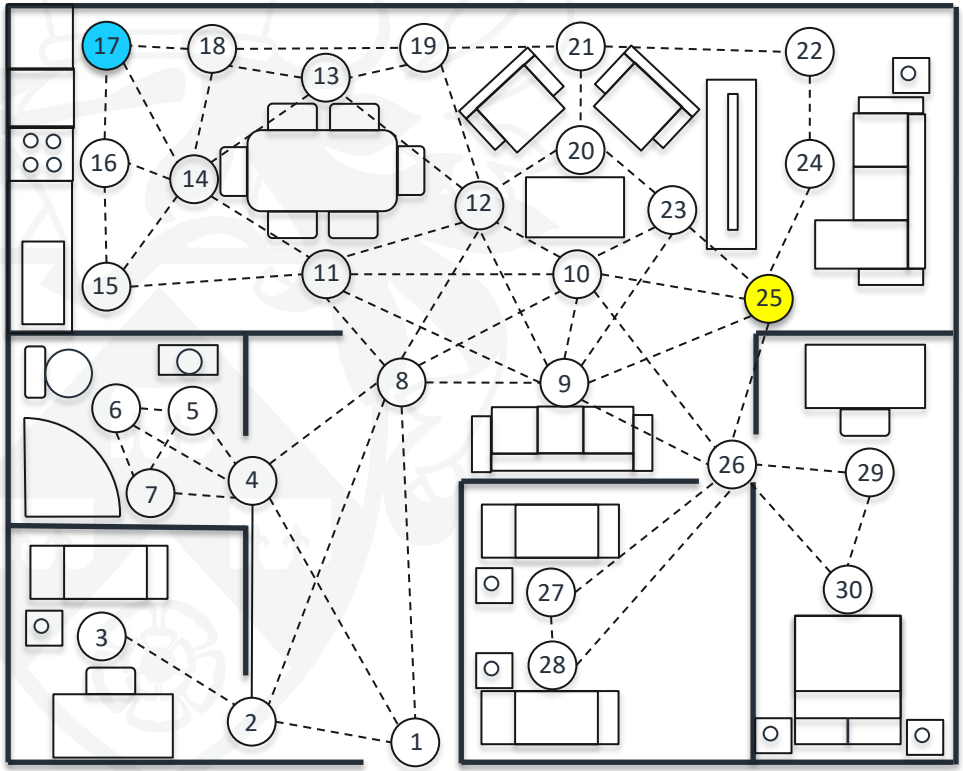
Path Finding

Dijkstra's Algorithm

- Enables the **shortest distance** path and the **highest probability** path to be determined for any graph-based environment
- In our environments, PRISM is then used to **validate** each path by creating a model at runtime
  - PRISM ensures that the output from Dijkstra is correct by **systematically solving** the path against a PCTL relationship…
  - **Rules** can be applied on whether a path is deemed acceptable by the planner, i.e., $P_{max} \geq 0.80$



Path Finding Example (Agent)



Start Node

Final Node

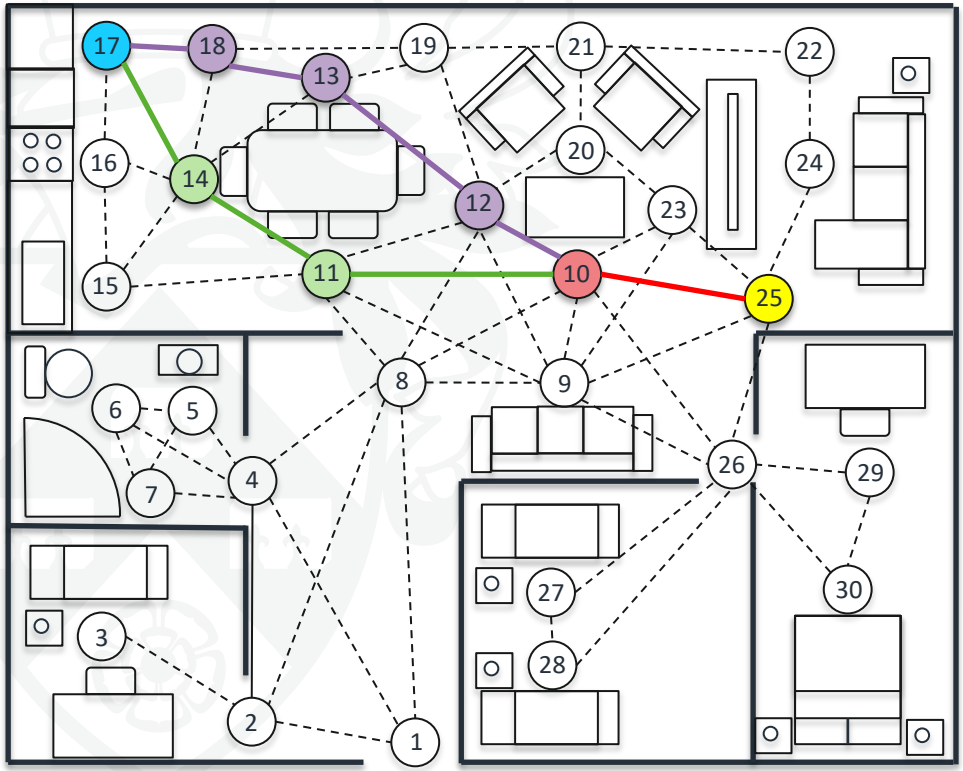
If we take the previously defined environment, where the robot:

- Starts at Node 25
- Ends at Node 17

Dijkstra's creates **two solutions:**

- Path of least distance
- Path of maximum probability of success

Path Finding Example (Agent)

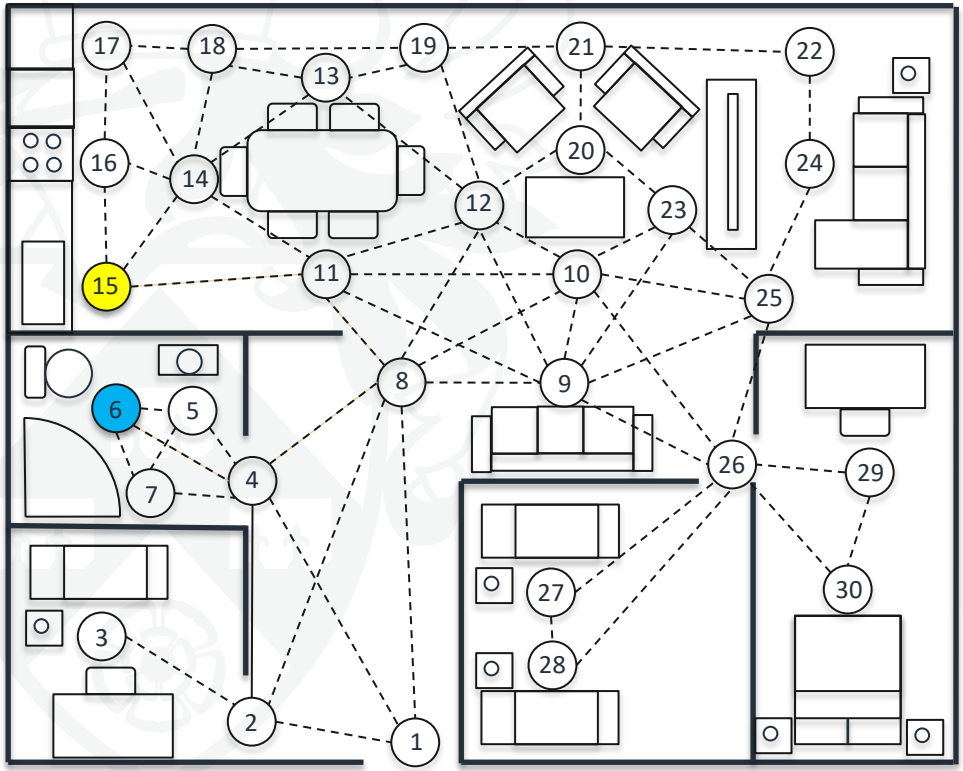


- Start Node
- Final Node
- Highest Probability Path (Robot)
- Minimum Distance Path (Robot)
- Mutual Path (Robot)

Minimum distance path is 2.9 m with 58% chance of success

Maximum probability path is 4.0 m with 92% chance of success

Path Finding Example (Human)



Start Node

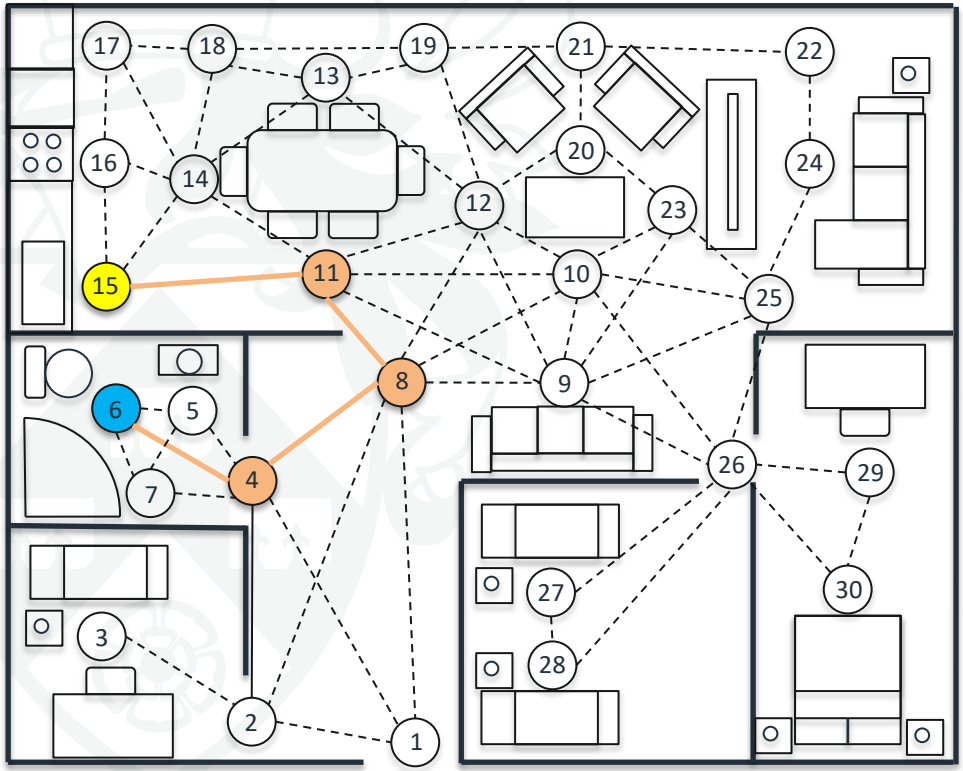
Final Node

Introduce a *human* into the environment at Node 15

The human intends to navigate towards Node 6

Unlike the robot, it is **assumed** the human moves along the path of **least distance.**

Path Finding Example (Human)



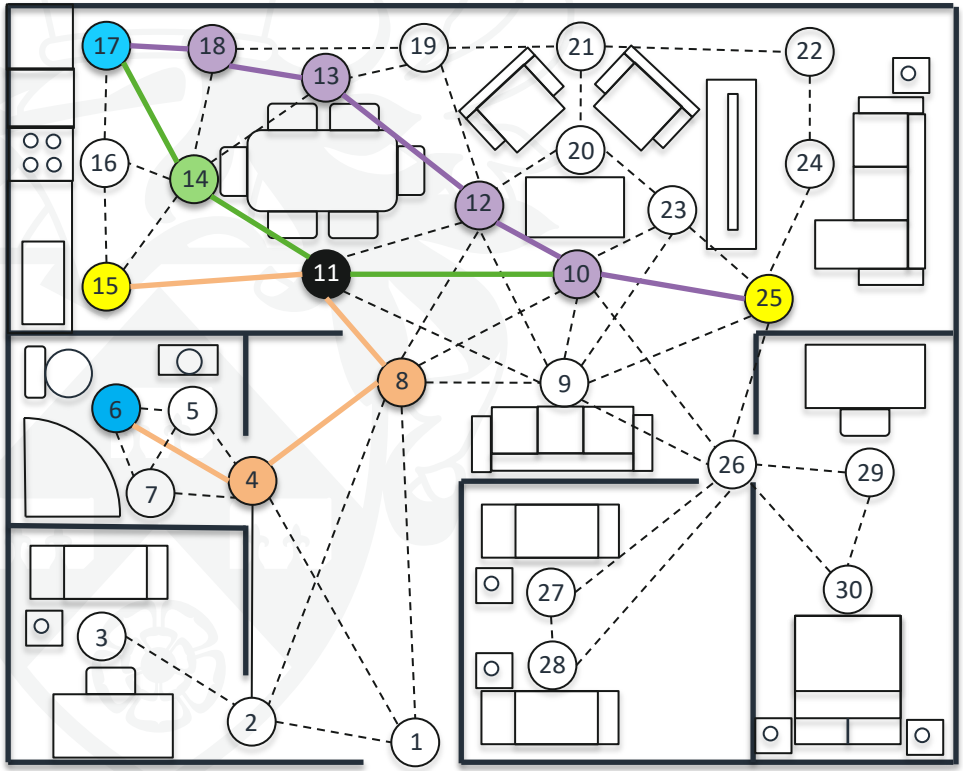
Start Node

Final Node

Minimum Distance Path (Human)

This path causes a **spatial conflict** with the path of the *agent*

Path Finding Example (Co-Op)

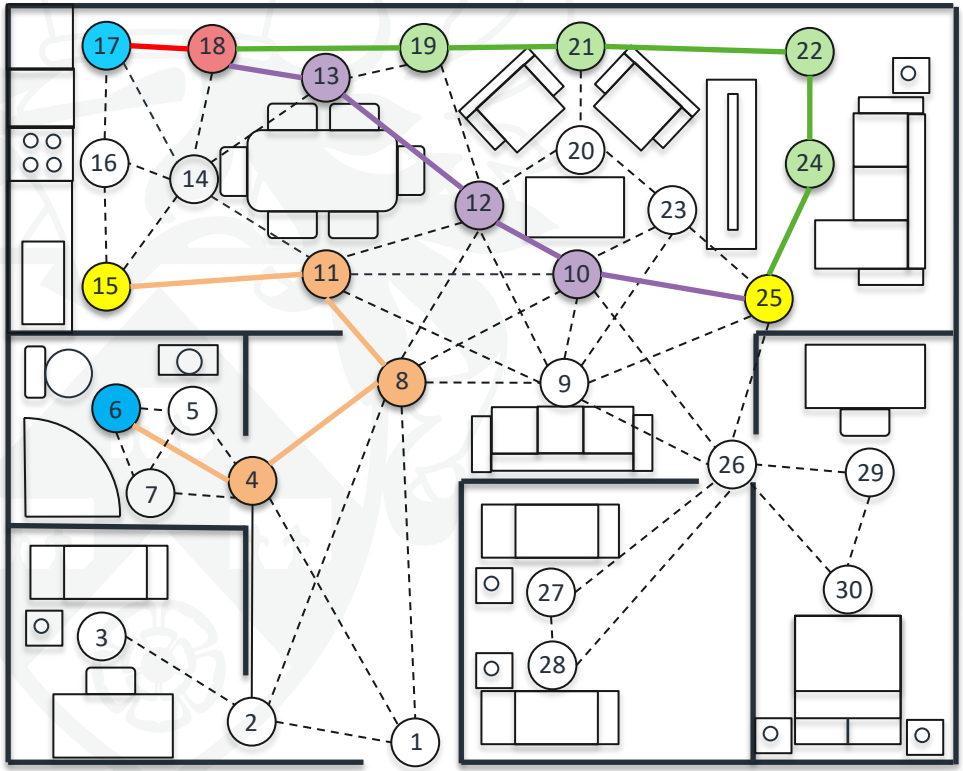


- Start Node
- Final Node
- Highest Probability Path (Robot)
- Minimum Distance Path (Robot)
- Mutual Path (Robot)
- Spatially Conflicted Node

Artificially update the probability *heat map* for the *agent* based on the path of the *human*

- Probability **decreased** from 92% to 23% chance of success

Path Finding Example (Co-Op)



Start Node

Final Node

Highest Probability Path (Robot)

Minimum Distance Path (Robot)

Mutual Path (Robot)

The agent now **re-plans** it's path around the environment, **increasing** the probability of reaching the final node

Maximum probability path is 4.8 m with 81% chance of success

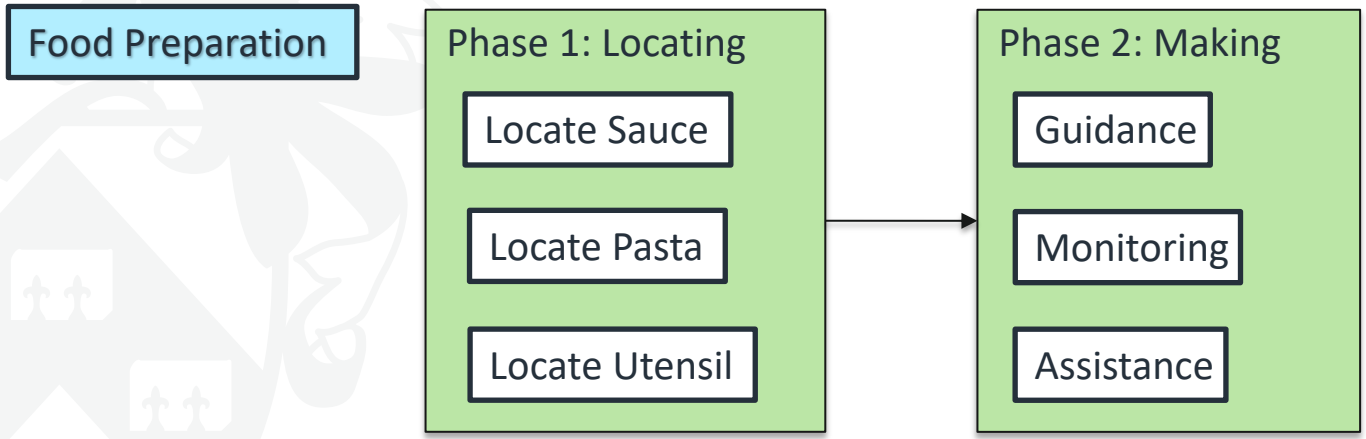
What does this mean?

1. Require knowledge of the **human's position** and **intentions**
   - This could be a fair assumption if we were guiding the human to locations during a cooperative task…
2. Requires **accurate predictions** of the human's behaviour
   - How they will navigate through the environment?
   - Assume the human takes the **least distance** path (Factor some creativity?)
3. Working on **spatial** deconfliction (not temporal & spatial deconfliction)
   - Future work for time-based simulations with kinematic movement analysis?
   - Purely predictive behaviour?
   - Adaptation at runtime?

These are all thing we need to consider in the final solution

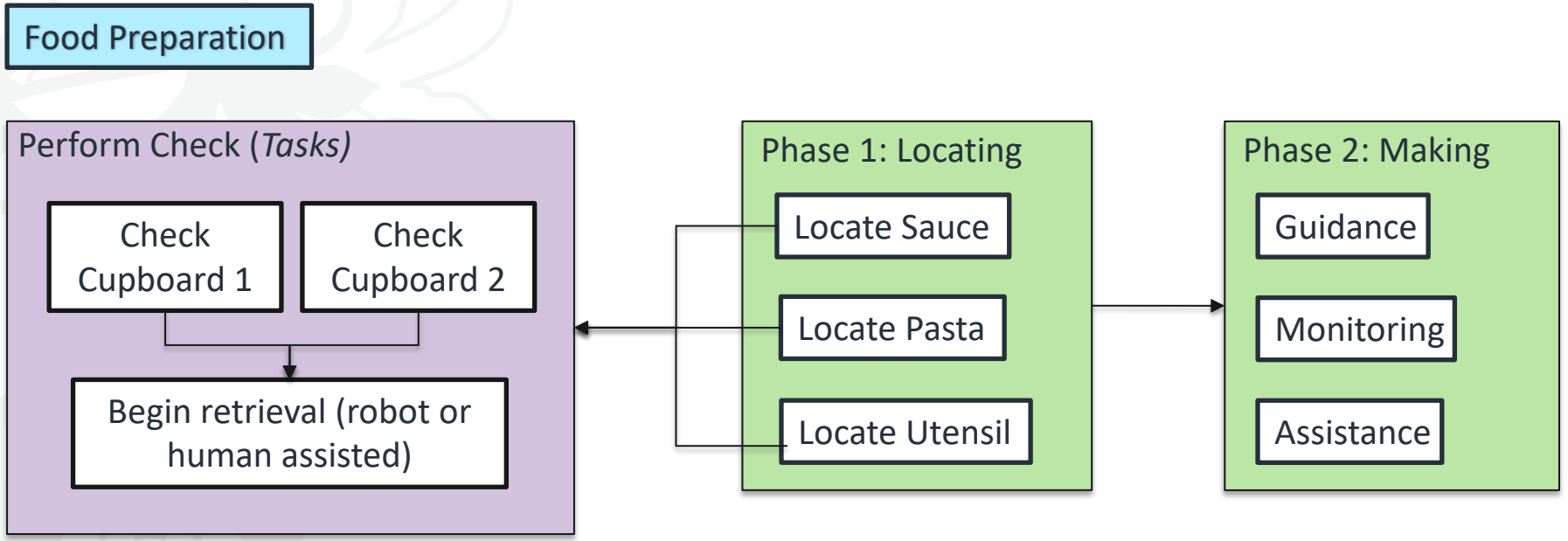
Mission Management

Previously discussed how a *mission* can be comprised of **multiple *tasks*,** each with *sub-tasks.*



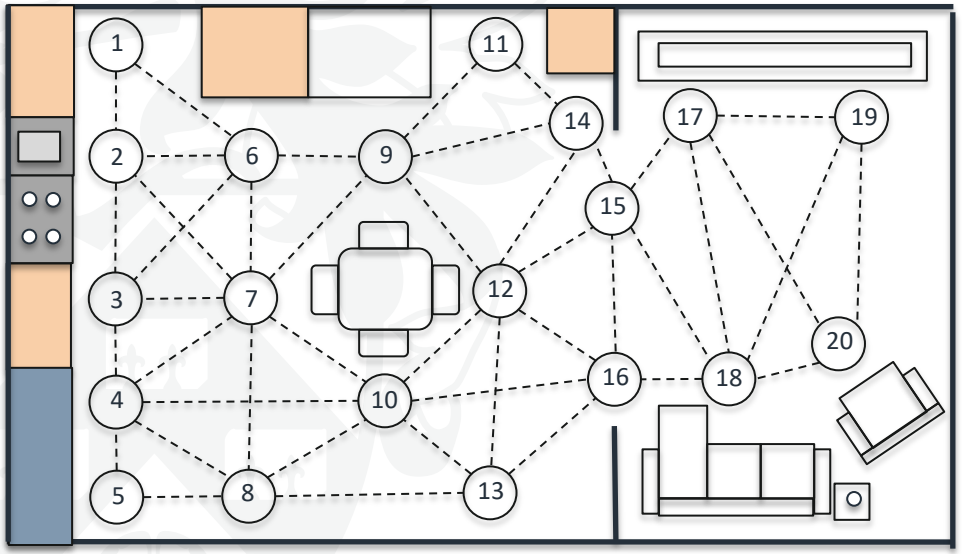
Must perform all tasks in *Phase 1* before *Phase 2* can start

Mission Management



Each *task* in *Phase 1* may be comprised of individual *sub-tasks*… all of which also need to be completed

Mission Management



Two room environment which has a kitchen and sitting area…

Graph consisting of 20 nodes is created, where **edges** represent a linear distance and risk factor (probability values)

Each *task* in a *mission* is performed at a specific node…

For example: **check cupboard** could occur at **Node 1, 6 and 3**

Defining a Mission

Consider the food preparation mission introduced before, with the following abstracted workflow:

Phase 1:
// Check Cupboard 1 → Check Cupboard 2 → Check Fridge → Check Table Clear → Hold/Wait…

All of the tasks in Phase 1 must be completed before Phase 2 can begin.

Phase 2:
// Begin instructions for making → Loop Phase…

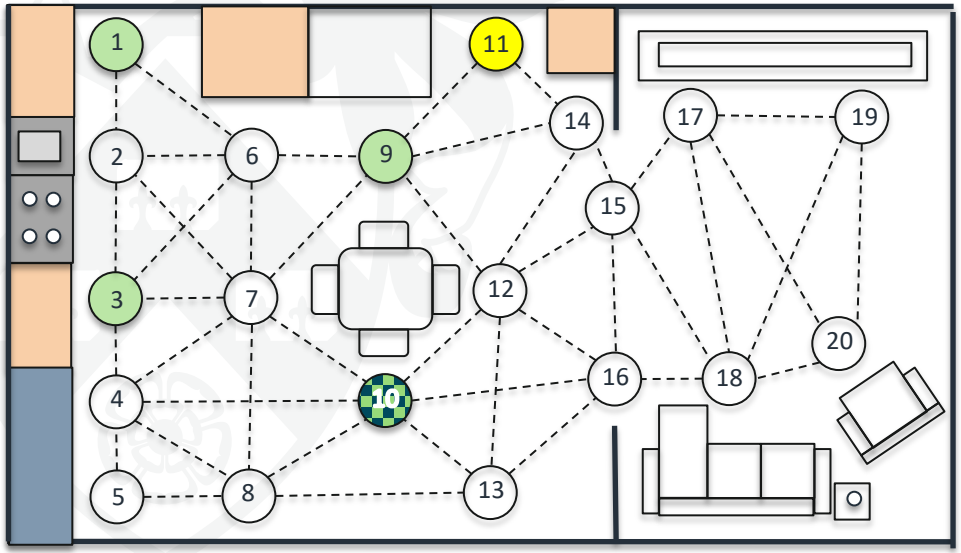
In Phase 1, there are **four** *tasks,* which are **not order dependant**. This means that the tasks can be completed in any order, and only when all have been completed, can progression be made to the **"Hold/Wait"** *task.*

These tasks are denoted as **"unordered tasks"**

Defining a Mission

Based on **Phase 1:**

- Start Node 11
- *unordered tasks at nodes :* [1, 9, 10, 3]
- Final **Hold/Wait** occurring at Node 10.



The mission is therefore defined as:

$$\text{Mission} = [11,1,9,10,3,10]$$

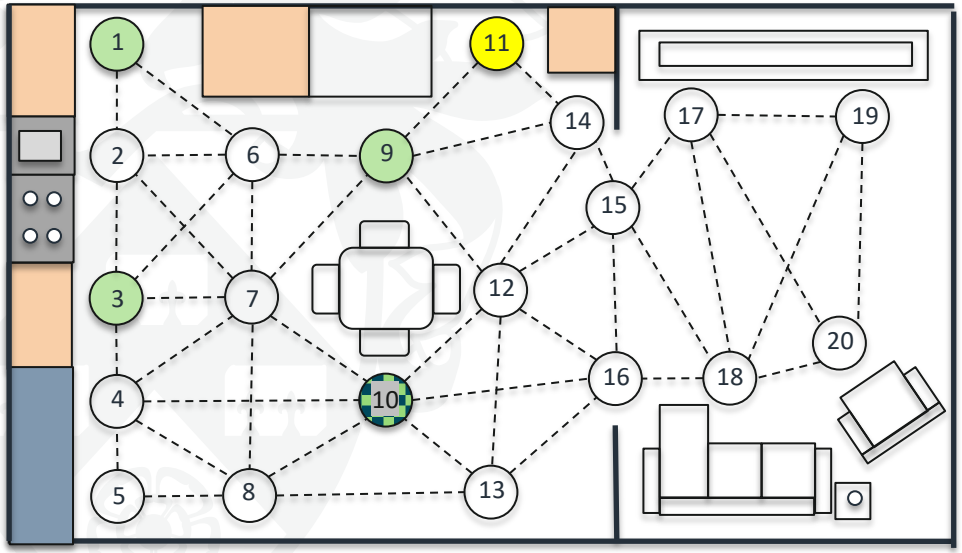
Where each $task \in mission$ has a *header*

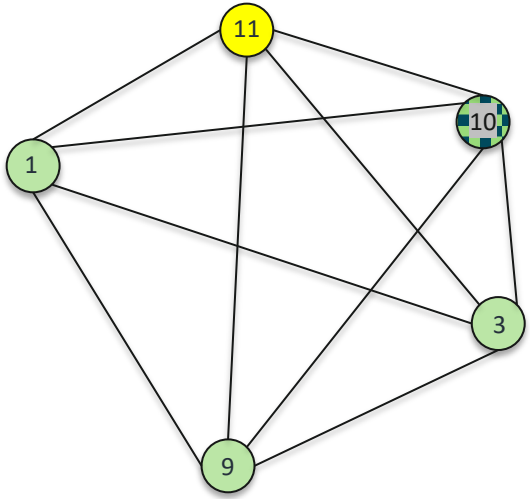
task = [‘S’, ‘C’, ’C’, ‘C’, ‘C’, ‘H’]

These headers are used to **identify** *breaks* and *unordered* tasks to obtain the **most efficient solution**

Defining a Mission

The nodes representing *tasks* in the set *mission* are used to create a fully connected **subset graph**

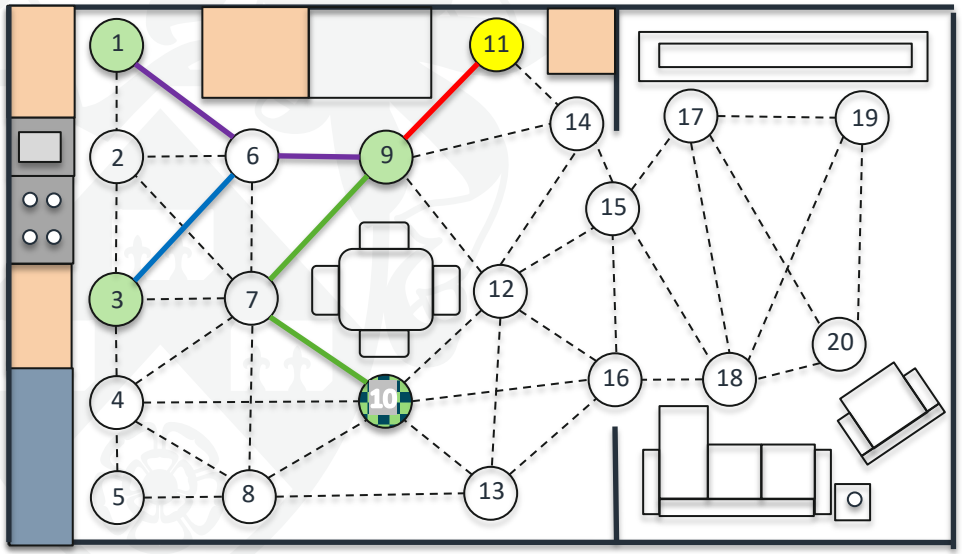


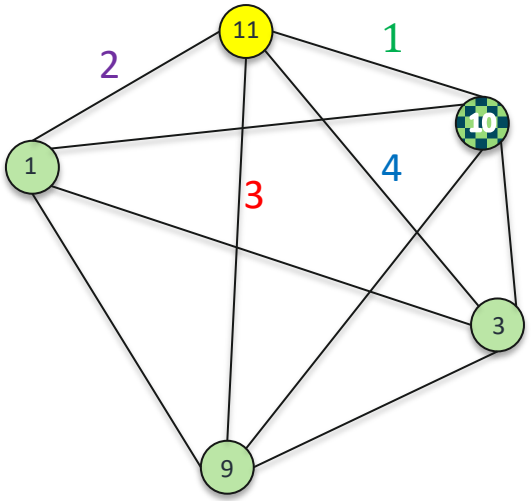


The *edges* of the subset map are unknow and determined by obtaining the solution between each node on the main map using Dijkstra

Subset Graph Solution

The *edges* connecting each node in the subset map are determined by solving Dijkstra's algorithm for **distance** and **probability** on the main map





$1 = \{D_{10\to11}, P_{10\to11}\} = 11 \to 9 \to 7 \to 10$

$2 = \{D_{1\to11}, P_{1\to11}\} = 11 \to 9 \to 6 \to 1$

$3 = \{D_{9\to11}, P_{9\to11}\} = 11 \to 9$

$4 = \{D_{3\to11}, P_{3\to11}\} = 11 \to 9 \to 6 \to 3$

Subset Graph Solution

The order to complete the task is solved as TSP:

- Least distance, and/or
- Maximum probability

For

$$mission = [11, 1, 9, 10, 3, 10]$$

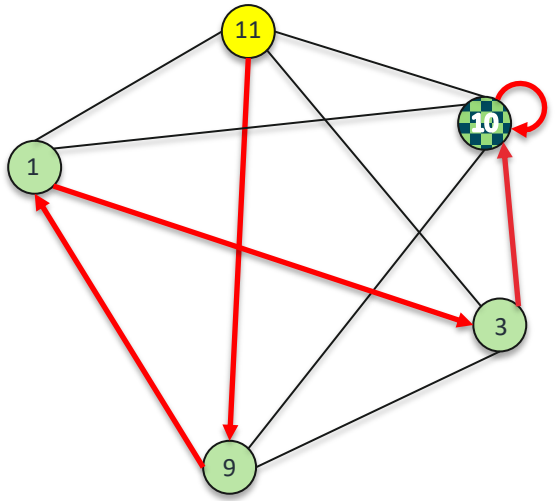
with

$$task = [\text{'S', 'C', 'C', 'C', 'C', 'H'}]$$

Solution

$$Start\ (11) \rightarrow 9 \rightarrow 1 \rightarrow 3 \rightarrow 10 \rightarrow end\ (10)$$

Therefore $mission = [11, 9, 1, 3, 10, 10]$



Multi-Unordered Solutions

This methodology can be **expanded** to **large mission** profiles consisting of **numerous ordered and unordered phases**, such as:

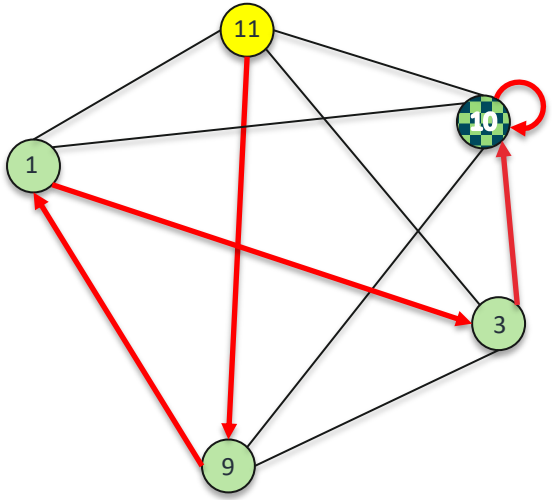
$$mission = [1, 5, 8, 10, 1, 2, 8, 15, 8, 2, 1, 2, 4, 5]$$

where

task =['S', 'C', 'C', 'C', 'H', 'C', 'C', 'H', 'C', 'C', 'C', 'C', 'C', 'H']

This *mission* has **three** task phases:

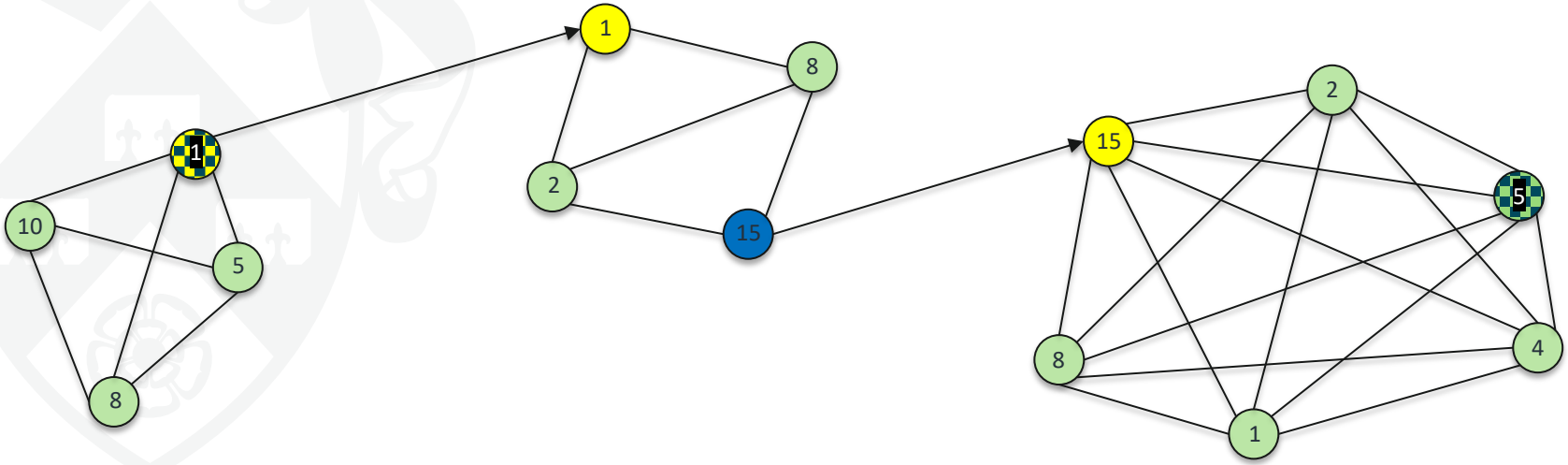
1. [1, 5, 8, 10, 1] → ['S', 'C', 'C', 'C', 'H']
2. [1, 2, 8, 15] → ['H', 'C', 'C', 'H']
3. [15, 8, 2, 1, 2, 4, 5] → ['H', 'C', 'C', 'C', 'C', 'C', 'H']



Multi-Unordered Solutions

Create an **individual sub-graph** for each phase of the mission, where all **unordered tasks** are **fully connected.** Each phase is connected using the **header exit and entry point**.

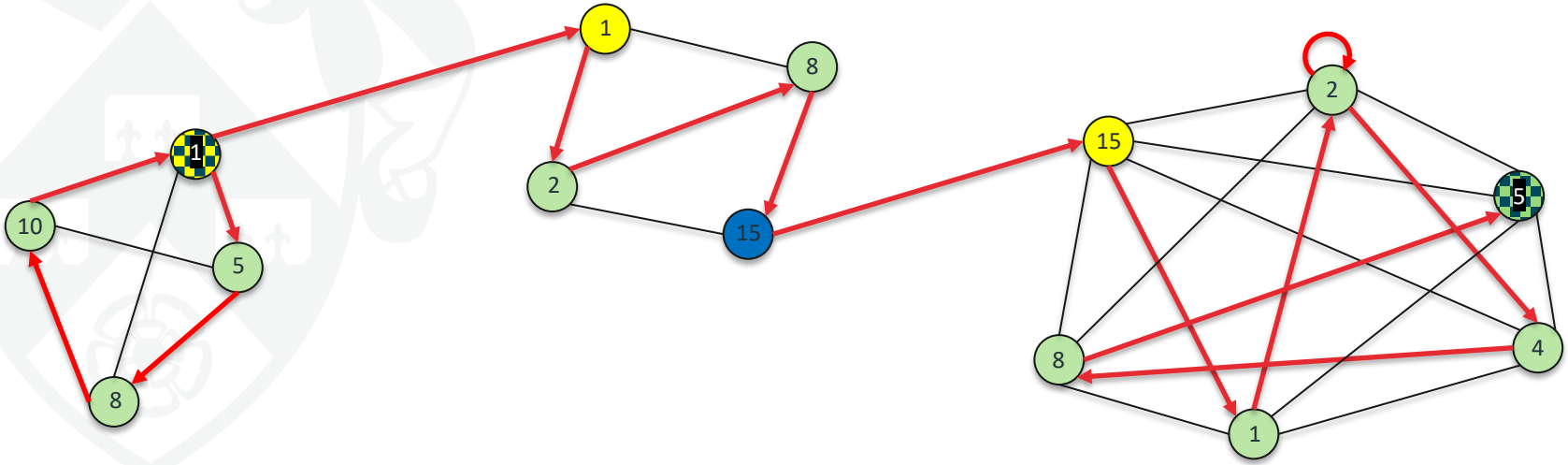
The *edges* are solved in the same manner as the previously demonstrated single unordered solution… by obtaining a local solution using Dijkstra on the main environment map.



Multi-Unordered Solutions

Solving each *phase* as a *least distance* TS problem, the solution is:

1. *Phase 1:* $1 \rightarrow 5 \rightarrow 8 \rightarrow 10 \rightarrow 1$
2. *Phase 2:* $1 \rightarrow 2 \rightarrow 8 \rightarrow 15$
3. *Phase 3:* $15 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 5$



Next Stages

1. Refine the mission management and mission allocation structure
2. Introduce human movement analysis into the mission management analysis
3. Temporal deconfliction analysis human and robot movement during path finding
4. Adding some level of ”creativity” or probabilistic planning for human pathing behaviour
5. Start looking into simulation and interface with ROS