The main goal of this project is to determine the most likely scenario, every 15 seconds, about the occupancy of different rooms in an office. In order to solve the given problem, the chosen Probabilistic Graphical Model was a mixture of modified Markov Chains (MC) and Hidden Markov Models (HMM). We used MC to model rooms without sensors inside and HMM in other cases. The modification is that there can be multiple previous timestep nodes feeding into the current timestep. In the model, a state variable represents the probability that there are 1 or more people in the room, with no distinction made for one vs multiple people. This is one of the main simplifications in the model. In addition, the following assumptions were made:

- The current occupancy of a room is influenced by the occupancy of the adjacent rooms in the previous timestep, as people can only walk so fast through the office.
- Since we only consider whether a room has people or not, the sensor data was simplified as it follows:
  - Doors sensor emission tables were trained considering activation just when at least two people walk through the door.
  - The robot information is used in terms of binary state (there are people in a room or there are no people).
  - According to the testing data, the robots are 100% accurate. Therefore, when a robot gives information about a room, that was considered basically 100% true and the probability distribution of that room is changed radically following the robot information, ignoring all other factors.
- The initial probabilities were set as 0.05 probability of people being in them (this was set by trial and error). The only exception is the outside, which starts with probability 0.99 for occupancy since that is where people start from.
- If electricity is cheaper at a given time-step, then the model should require a lower probability to turn lights on compared to if electricity is more expensive.

To construct the graph used to calculate transition probabilities between rooms from the previous time-step to the current time, we first started with the full office adjacency graph (modelling how all rooms connect to each other), and then iteratively removed connections from rooms that had the most connections, carefully testing which rooms removal would either increase the model accuracy (by reducing overfitting and noise) or have the least negative impact. This process was repeated until rooms had at most 3 connections. This was done since our algorithm efficiency is exponential in the number of connections rooms have, so reducing the maximum number of connections massively improved performance while actually improving its accuracy, likely due to (as stated above), reducing overfitting and noise from so many factors.

In relation to the probability tables, the initial probability states were set as described above. The transition probability and the emission probability tables were constructed based on the empirical data. We used the structure of the function 'estProbs' (from the tutorials) with a modification for the case of transition probabilities, given that it needed to use the 'parents' one time step back in time.

The algorithm used was a mix between mini-forward (for rooms modelled as MC) and forward (for rooms modelled as HMM). It was implemented using a modification of 'mini-forward' and 'forward' functions from the tutorials (the modification allows us to use a state variable which has more than one parent). We made the following approximation to calculate the probability distribution of a particular room in a particular time: The main concept is that the model starts with prior probabilities (initial probability tables), then it obtains a posterior. Then, the posterior is going to be the prior and so on. Then, when the posterior is calculated it is done in the following way, take the case of room 2 (r2) which has neighbours r1 and r4. Then to get the probability distribution for room r2 at time t,

$$P(r_{2_t}) = P(r_{2_t}|r_{4_{t-1}}, r_{1_{t-1}}, r_{2_{t-1}}) \cdot P(r_{4_{t-1}}) \cdot P(r_{1_{t-1}}) \cdot P(r_{2_{t-1}}),$$

where every $P()$ is a probability distribution table.

We made the following independence approximation, $P(r_{4_{t-1}}, r_{1_{t-1}}, r_{2_{t-1}}) = P(r_{4_{t-1}}) \cdot P(r_{1_{t-1}}) \cdot P(r_{2_{t-1}})$. This can be justified by considering that in the previous time-step, the probability distributions for every room was constructed based on the interactions between one another, so the dependency between parent rooms in the past is already accounted for (the only interaction that it is not considered is the last 15 seconds). This is a pretty good approximation which allows us to maintain tiny probability tables and reduce the computation time. The alternative would be to calculate the entire Markov chain for all previous time steps every time get_action is called, which is simply not feasible, especially given that our approximation loses very little, if any, information.

After applying the mini-forward algorithm it is returned a normalised distribution. Nevertheless, sometimes we are dealing with multiplication of tiny probabilities. Therefore, in order to avoid the underflow issue, normalisation is applied in every time step so we can be sure that every probability distribution is going to remain normalised through time and there is not going to be underflow.

The following diagram (figure 1) shows the interactions for room 2 at time t. Every room was modelled in the same way given its adjacent rooms (remember that for some cases there were simplifications and a room might not interact with every adjacent room to improve performance). If you look at the room r_2 at time t in this diagram (middle green circle), you see it is influenced by the probability that r_4, r_1 and itself had people in them 15 seconds ago. Alternatively, consider the room r_1 at tine t, which is influenced by room r_2 and itself 15 seconds ago, but also, like a HMM, has a sensor attached with an emission probability that gives us a more accurate probability value for this node. All these newly calculated probabilities will then be used to calculate room probabilities at $t + 1$, and so on.

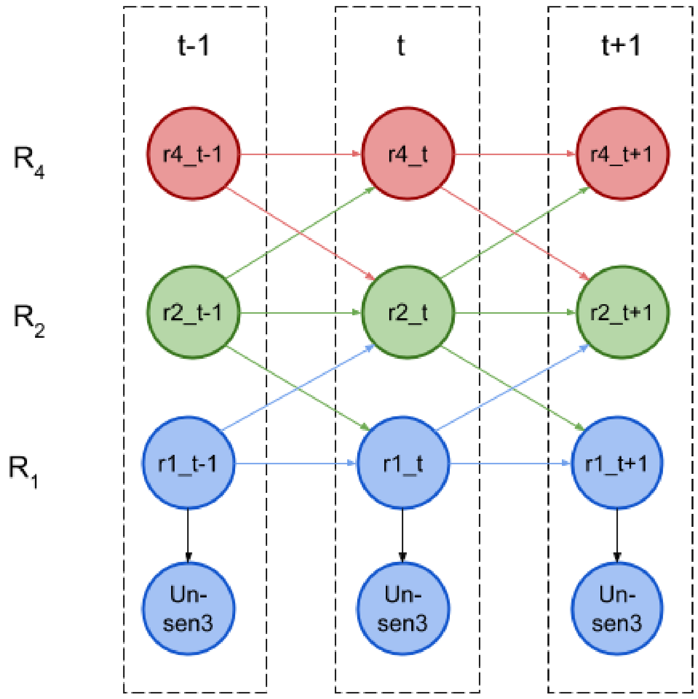
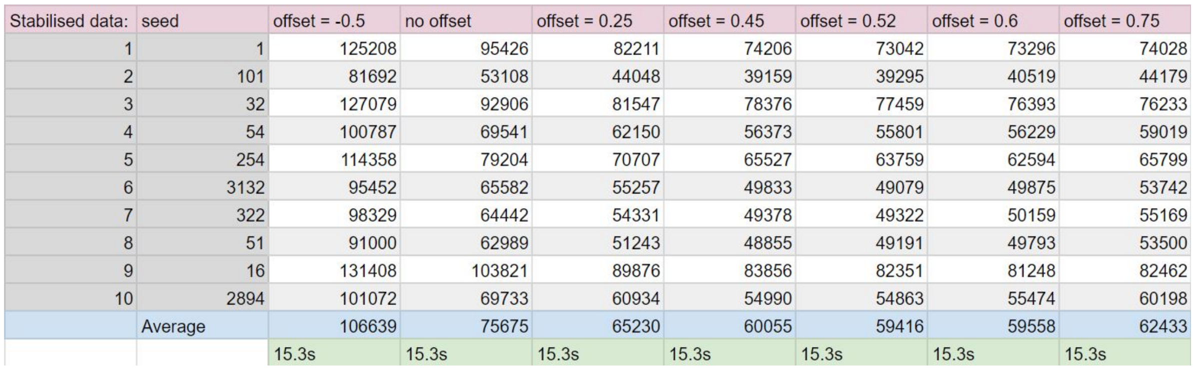


Figure 1: Partial diagram of the network

In terms of complexity, the lecture mini-forward and forward algorithms are, $O(n \cdot |X|^2)$, where n is the number of time steps which are going to be considered and $|X|$ is the number of possible values of X. In the case of the modified mini-forward and forward, there are multiplications (joins) between the room state in t and all connected room states at time $t - 1$, including itself. Therefore, the complexity in every time step for one state room X is $O(|X|^{p+1})$, where p is the number of adjacent rooms connected to X in our graph (which doesn't include all real world room adjacencies). Then, considering that every state variables is binary (people in the room/people not in the room and sensor active/sensor off), the total complexity will be $O(n \cdot 2^{p+1})$, where n is the number of time steps that will be considered (in one day there are 2400) and p the largest number of adjacency rooms of any room (it was not always considered all the adjacency rooms in order to simplify the model and reduce the time complexity).

Due to how we created our model, dealing with broken sensors is easy, we simply switch to treating the room with the broken sensor as a MC like all other rooms that don't have a sensor until the sensor reactivates.

During development, we tried a number of optimisations to improve performance, some of which worked, while others didn't. By far the most impactful one was introducing an offset value when determining whether to turn lights on or off. So rather than simply turning lights on if $P(\text{People in room}) \geq P(\text{No people in room})$, we introduced an offset, so lights would go on if $P(\text{People in room}) + \text{offset} \geq P(\text{No people in room})$, skewing in favour of having lights on rather than off. The optimal value for this offset was found through careful testing to be .56, which is significant, but improved our average score (at least in our testing) by over 10k cents. Attempting to build on this success further, we tried introducing an individual offset for every room, but after optimising for the electricity price in data.csv, we found that it had overfitted and performed worse on average for random electricity prices. We did however find two rooms that benefitted from having zero offset, which improved the average score. Another improvement we made was including electricity price in the decision to turn lights on or off. We added another offset, similar to the one discussed above, but dependent on electricity price, which high prices making it slightly less likely that the light will be turned on, and low prices making it slightly more likely which improved the score of our model further. Here (table 1) is some of the testing data used to derive the optimal value of offset:



Stabilised data:	seed	offset = -0.5	no offset	offset = 0.25	offset = 0.45	offset = 0.52	offset = 0.6	offset = 0.75
1	1	125208	95426	82211	74206	73042	73296	74028
2	101	81692	53108	44048	39159	39295	40519	44179
3	32	127079	92906	81547	78376	77459	76393	76233
4	54	100787	69541	62150	56373	55801	56229	59019
5	254	114358	79204	70707	65527	63759	62594	65799
6	3132	95452	65582	55257	49833	49079	49875	53742
7	322	98329	64442	54331	49378	49322	50159	55169
8	51	91000	62989	51243	48855	49191	49793	53500
9	16	131408	103821	89876	83856	82351	81248	82462
10	2894	101072	69733	60934	54990	54863	55474	60198
	Average	106639	75675	65230	60055	59416	59558	62433
		15.3s	15.3s	15.3s	15.3s	15.3s	15.3s	15.3s

Table 1: Sensitivity Analysis