QUANTUM NETWORKING PROBLEM STATEMENT

ABHAY SHANKAR K: CS21BTECH11001

1. Barebones

We are given a static network in the form of a graph $\mathcal{G} = (V, \mathcal{E})$, and a demand vector $\mathcal{D}$. The demand vector is a mapping from the set of all pairs of nodes in the graph to a non-negative integer.

Each edge is a 3-tuple (c, τ, Γ) with

- c being the cost of the link
- τ being the cooldown time
- Γ being the capacity of the link - the number of bell pairs one can send across.

Each element of the demand vector is a 3-tuple (u, v, d), with

- u: source node
- v: destination node
- d: the number of bell pairs to be sent

We seek to find a routing scheme that minimizes the cost of the network, while satisfying the demand vector. We will interpret this as assigning each edge to a given demand for a time slot, i.e., finding a function $f : T \times \mathcal{E} \to \mathcal{P}(\mathcal{D})$ that minimises the cost

$$\mathcal{C} = \sum_{t \in T} \left(\sum_{e \in E'(t)} c(e) \right)$$

where $E'(t) = \{e \in \mathcal{E} \mid f(t, e) \neq \{\}\}$ is the set of all edges that are active at time t, and $T \subset \mathbb{N}$ is the set of all time slots during which the network is active.

Alternatively, we could also choose to optimise any of the following quantities, or establish relationships between their optimal values:

- Minimise the total time taken, i.e. $|T|$
- Maximise the utilisation of the network, i.e. $\dfrac{\sum_{t \in T} |E'(t)|}{|\mathcal{E}|\,|T|}$
- Minimise the maximum utilisation of the network, i.e. $\max\limits_{t \in T} \dfrac{|E'(t)|}{|\mathcal{E}|}$, e.g. congestion control.

Our assumptions are as follows

- Discretized times slots
- The bell-pair measurement never fails.
- The quantum links between two nodes do not fail.
- All measurements are instantaneous, and occur simultaneously relative to a synchronised global clock.

In the following sections, we shall relax these assumptions, progressing towards a more practical model.

2. Dynamic demands

We now consider the case where the demand vector is not static, but changes over time. We will assume that the demand vector is a function

$$\mathcal{D} : T \times V \times V \to \mathbb{N}$$

and that the demand at time t from node u to node v i.e. $\mathcal{D}(t, u, v)$ is known in advance $\forall t \in T, u, v \in V$.

The cost function remains the same, but the allocation function (our objective) needs to be recomputed at each time instance or each time the demand vector changes.

Clearly, such re-evaluation is not scalable, so we have two choices

- Experiment with clever ways of reusing the previous allocation function to compute the new one.
- Abandon the link allocation model and use standard routing algorithms. We chose not to do this for the simpler case because the link allocation function is a more general model, and we can always use it to implement a routing algorithm.

2.1. **Oracular dynamic demands.** As an intermediate between truly dynamic demand vectors and a static demand, we can consider a demand matrix $\mathcal{D}$ where the i'th row represents the demand vector at time i. We can then use the same model as before - no changes are necessary. This means that the demand is dynamic, but the algorithm knows the demand in advance.

3. Fallible links

We now consider the case where the links between nodes can fail. We assume

- Links are independent, i.e. the failure of one link does not affect the failure of another.
- The failure of a link is independent of the demand on that link.

If the network is represented using an adjacency matrix, we can encode the probabilistic network evolution with another matrix, and multiply the two matrices to get the new adjacency matrix.

Generally, we can leverage the existant routing algorithms for classical networks