REGARDING OPTIMAL RESOURCE ALLOCATION IN QUANTUM NETWORKS

ABHAY SHANKAR K, RISHIT D, SAKSHI TAKALE

Abstract. We wish to present a well-informed mathematical approach to the economics of quantum networks in the future. To do so, we discuss theoretically optimal link-bandwidth allocation and assign costs proportionate to the 'importance' of a link or its ends, and also present ways to assess said importance.

1. Recap

Find a function $f : T \times \mathcal{E} \to \mathcal{P}(\mathcal{D} \times \mathbb{N})$ 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
- $T \subset \mathbb{N}$ is the set of all time slots during which the network operates.
- The output of the allocation function is a set of (α, n) pairs, which specify the number of qubits (n) of which demand (α) the edge should transmit. This corresponds to $(\alpha_1, \gamma_1)..((\alpha_k, \gamma_k))$.
  - Here $\forall k\ \gamma_k < \alpha_k$`.demand`, i.e. the edge cannot transmit more qubits than the demand,
  - $\Sigma_k \gamma_k \leq \Gamma$ for a given edge,
  - Flow conservation: $\forall t \in T\ \forall u \in V\ \forall g \in \mathcal{E}_u\ \Sigma_g s(g) f(t, g) = 0$ where $\mathcal{E}_u$ is the set of edges incident on u and $s(g)$ is 1 for outgoing edges and -1 for incoming edges.
  - α_k is a valid demand.

2. Algorithm

I have no clue. Big graphy stuff. Probably finding the minimum demand, find the maxflow path between the nodes, assign the demand, and ad infinitum. Also probably have to do some BFS timeslot checks to make sure the link is not full. Maybe do buffering if link at low capacity.

2.1. TL;DR.

- Since we are aware a priori of all the demands on the network, we can use a link-state algorithm similar to OSPF, recomputing the shortest paths in each timeslot - while this remains impractical, it is merely a theoretical benchmark.
- Due to congestion concerns, simply using the cost of the link is suboptimal. We instead determine a pair of heuristics in the next section, which allows us to select a path based on both local link attributes and global path attributes. In this section, we assume such heuristics, viz. $\boldsymbol{\eta}$ for links and $\mathcal{H}$ for paths, exist.

2.2. Steps.

(1) Initialise the allocation matrix representing the output of the allocation function to $\mathbf{0}$. We grow this matrix with each timeslot.

(2) For each demand, $\alpha = (u, v, d)$, we find the set $\mathcal{P}$ of all shortest paths from u to v using $\boldsymbol{\eta}$. We may achieve this with a modification of Dijkstra's SSSP algorithm (shown below).

(3) We select a single path for each demand using $\mathcal{H}$, and mark the appropriate cells in the allocation matrix.

Dijkstra, except store all shortest paths. Maybe not *just* the shortest paths, give a little leeway. $\boldsymbol{\eta}$ within 0.05 maybe? (Let range of $\boldsymbol{\eta}$ be $[0, 1]$).

3. Heuristics

Too many ideas. We won't know which subset of this is practical until we math it.

- Total link contributions from links in the path (capacity, cooldown).
- Buffer size at nodes involved - that way greater chance of not being dropped.
- Bottleneck bandwidth of the path.
- Node costs!
- Betweenness-centrality, except for maxflow paths? Do we need a new link attribute for this? Maybe something in the topology that creates a certain distribution (think pmf) of this shiny new attribute among all the edges?

  Edit: Nah. Just BWC fine.
- GHZ extra cost?

  Edit: Sakshi problem.
- Link-layer QoS guarantees? - i.e. number of times a link tries to reconstruct a qubit if measurement keeps failing before giving up. We could incorporate a QoS guarantee into the link-layer protocol.

  Edit: Also Sakshi problem.

Complexity in generating $\mathcal{H}$:

We may have buffering, where a qubit does not move in a given timeslot. Thus, $\mathcal{H}$ needs to modify each path to account for buffering delays where consecutive duplicate nodes are inserted to indicate the same. This cannot be done elsewhere, as we require knowledge of the global qubit state to perform decisions rearding buffering.

THIS IS BEGINNING TO FEEL LIKE AN RL PROBLEM. WHY. I CANNOT.

Edit: Maybe do one RL, compare to various homegrown heuristics, and pick a good one. Or maybe just RL, eh.

4. Sample

Rishit Problem.

So we put up a few different networks of a few different topologies (not too many) and an arbitrary demand vector, then compute the cost of each one.