CSL860: Routing in the presence of faults Amitabha Bagchi
IIT Delhi Scribe: Md Tanveer Alam, Harsh Sanghvi

Lecture 5: Scheduling in Networks and Embeddings

23rd and 24th September, 2008

5.1 Introduction

The two main aspects of any network are routing and scheduling. Routing is concerned with deciding the path between any two nodes of the network. Scheduling deals with the policies which decide how to send packets across links.

Uptil now, we had been studying the routing aspect of the network. We had a brief look at the graph properties like diameter and expansion. Then we looked at the effect that faults have on the connectivity of the network. We studied about two kinds of faults namely adversarial faults and random faults.

Now, we turn our attention to the other very important aspect of the network i.e. scheduling. Scheduling policy is a very important factor in determining the efficiency and robustness of the network. As before, we would start with a very simple model. Later, we would study how the behaviour of network changes on introduction of faults in the network. The material in this lecture is taken from Leighton, Maggs and Rao's 1994 paper [1].

5.2 Some results from probability theory

1. **Chernoff bounds**: Let $X_1, X_2......, X_n$ be independent binary random variables (it can only aquire values 0 and 1), and $X = \sum_{i=1}^{n} X_i$ then $\forall \delta > 0$ and $\mu \geq E(X)$

   $$P(X \geq (1+\delta)\mu) \leq e^{-\min(\delta,\delta^2)\frac{\mu}{3}}$$

   For $0 \leq \delta \leq 1$
   $$P(X \leq (1-\delta)\mu) \geq e^{-\delta^2 \frac{\mu}{2}}$$

2. **Lovasz local lemma**: Let $A_1, A_2......, A_n$ be a set of '*bad*' events. By '*bad*' events, we mean that we are interested in the event in which none of A_i where $1 \leq i \leq n$, happens. Suppose that every event A_i is

independent of all but at most d events and $Pr(A_i) \leq p \; \forall \; 1 \leq i \leq \; n$. If $ep(d+1) < 1$, then

$$Pr(\bigcap_{i=1}^{n} \bar{A}_i) > 0$$

5.3 Scheduling and its parameters

The model: Consider the following setting. We have a network $G(V, E)$ and a collection of paths $R = \{(u, v, p) : u, v \in V, \; p \, is \, a \, path \, in \, G \, between \, u \, and \, v\}$. As before, we make the following assumptions.

- At any given time only one packet travels through any link.
- All packet movements are synchronous w.r.t. a global clock.
- Each node is allowed to buffer as many packets as it needs to.

The parameters: To enable us to quantize the scheduling problem, we define the following parameters of scheduling.

- **Congestion(C):** Congestion is defined as the maximum over all the edges of the number of paths using an edge.
- **Dilation(D):** Dilation is the maximum length of paths in R.

To illustrate, the ring network model discussed earlier (refer Figure 1 from Lecture 1) with m initial packets and n nodes will have congestion $C \leq m$ because in the worst case all packets can pass through a single edge. Also, dilation $D \leq \frac{n}{2}$ which is the maximum length of all paths in the ring model.

Now, given a scheduling problem with parameters C and D, let us discuss it's approximate upper bound and lower bound. By definition of congestion, a packet may have to wait for C steps and then move D steps, so the lower bound can be taken as $\Omega(C + D)$. There may be conditions when it may take lesser steps to route and hence the lower bound is not strict.

Also for the upper bound, consider the worst case when every edge has congestion C and each path has D edges. Hence, it would require at most $O(CD)$ steps to sink all the packets. ■

Theorem 5.1 *Given any scheduling problem with parameters C, D we can route all packets in $O((C + D)2^{\log^* \max(C,D)})$ steps.*

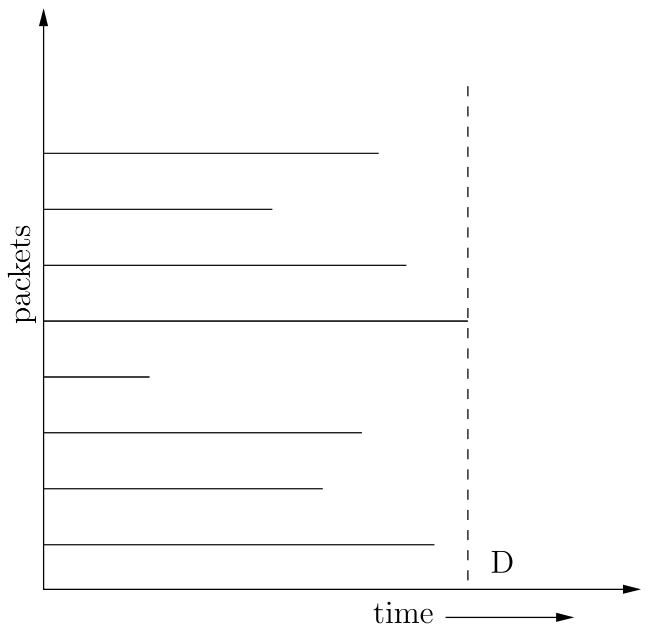

Figure 1: Packets with independent movements

Proof: We prove the above theorem by construction of a randomized scheduling algorithm. Consider the situation when all packets move independent of congestion as shown in Figure 1. The figure shows all the packets moving unobstructed in time along the X-axis. We will describe a recursive algorithm to introduce random delays in between these packet paths. Our objective will be to reduce congestion from C to a constant.

Suppose in the situation shown in Figure 1, we give a random initial delay to all the packets chosen independently from $[2C]$ where $[k] = \{0, 1, 2, \ldots, k\}$. After this initial delay, we would make every packet move independently as if there were no delay. The new situation is shown in Figure 2. Notice that while in Figure 1, the maximum time taken by any path was D, now maximum time is $D + 2C$. In general, congestion would come down, as packet's time of travel has been spread randomly.

We shall use the following notation:

- **t-interval**: an interval of t consecutive time steps.
- **t-frame**: a t-interval beginning at a multiple of t.

- I_j: $I_j = \lceil 18(\log I_j + 1) \rceil$ for all $j > 1$ where $I_0 = \max(C, D)$
- $A_e(I_j)$: the event that more than I_j packets use an edge e in an I_j-interval. Alternatively, $C \geq\ I_1$ for an I_1-frame.

Now we describe a recursive algorithm where at each level of recursion j we divide the timeline into I_j-intervals and convert each of these I_j-intervals into sub-problems of the original scheduling problem independent of other intervals at the same level.

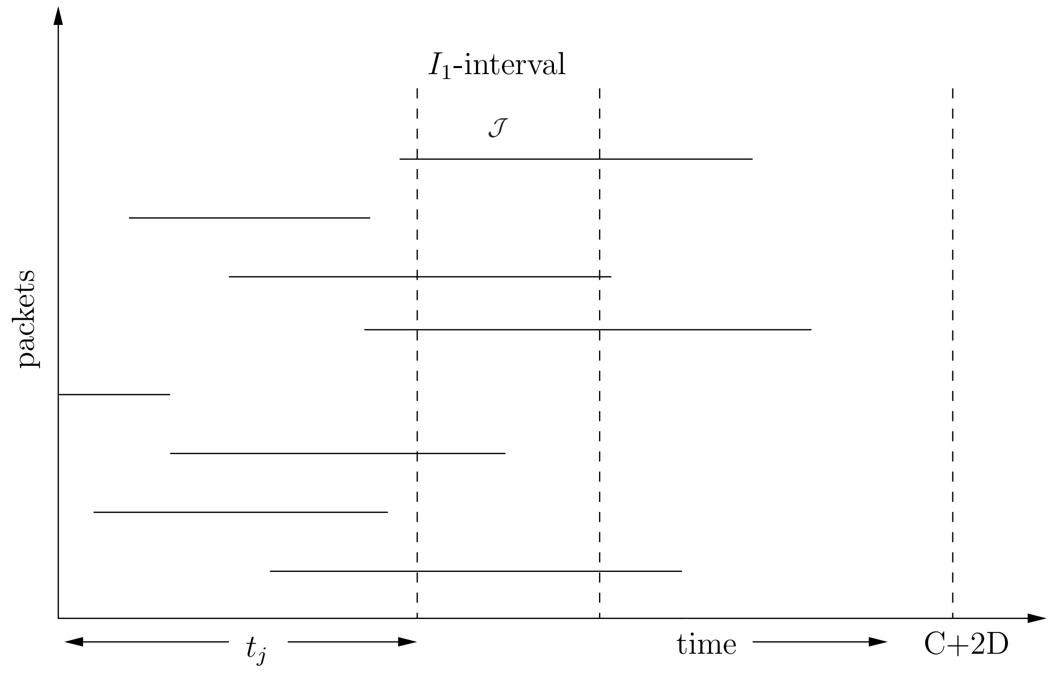

Figure 2: Packets with initial random delays applied

We consider any I_1-interval in this new situation and call this particular I_1-interval as $\mathcal{J}$. Now in this interval $\mathcal{J}$, for any edge e, let packets numbered $1, 2, \ldots, l$ packets pass through edge e. Clearly, $l \leq C$. We define the random variables $X_1, X_2, \ldots, X_l$ such that:

$$X_j = \begin{cases} 1 & \text{if the packet } j \text{ uses } e \text{ in interval } \mathcal{J} \\ 0 & \text{otherwise} \end{cases}$$

Hence, the random variable $X = \sum_{i=1}^{l} X_i$ would denote the number of packets passing through an edge e. Consider any edge e on the timeline showing the flow of any packet. Since, the line is being shifted with an initial delay, the edge on the timeline of the packet can either be inside $\mathcal{J}$ or outside. Now, for an edge e to remain inside $\mathcal{J}$, we can give at most I_1 initial delays to the packet. In all the other cases the edge e would get outside the interval $\mathcal{J}$.

$$P(X_i = 1) \leq \frac{I_1}{2C}$$

(Since from $2C$ possible initial delays only I_1 values lead to this event of $X_i = 1$).

Now, as $l \leq C$ we get,

$$E(X) = E(\sum_{i=1}^{l} X_i) \leq \frac{I_1}{2}$$

Using Chernoff's bounds

$$\begin{aligned} P(X \geq I_1) &= P(X \geq (1+\epsilon)\frac{I_1}{2}) \\ &\leq e^{\frac{-I_1}{6}} \\ &\leq e^{-3(\log\ \max(C,D)+1)} \\ &= (e.\max(C,D))^{-3} \end{aligned}$$

But, since we chose $\mathcal{J}$ to be any arbitrary I_1-interval, we can have $O(2C+D)$ such intervals (i.e. for different values of t_j, see Figure 2) all of which are equi-probable. Hence, we conclude

$$P(A_e(I_1)) \leq \frac{2C+D}{(e.max(C,D))^3}$$

We have so far bounded p of the Lovasz Lemma. Next, to apply Lovasz Lemma, we need to know the bound on the value of d. We notice that the only way when the congestion of two edges is dependent is when there are one or more packets common to them. For any edge e the maximum number of paths including e is C (by definition of congestion C). Also, the maximum number of edges for each such path is D (by definition of dilation D). Hence, maximum number of edges dependent on edge $e \leq CD$. Thus, $d \leq CD$.

Now to test whether $ep(d+1) < 1$ we plug in the values of $p \leq \frac{2C+D}{(e.max(C,D))^3}$ and $d \leq CD$. It turns out that $ep(d+1) < 1$ holds for all values of C and D.

Thus applying Lovasz Lemma, we can conclude that there exists some setting of starting delays for which none of the $A_e(I_1)$ events occur for all edges e. Let us take one such setting. We have clearly upper bounded the congestion at each edge e to I_1 by definition of event $A_e(I_1)$. Now divide the time space into I_1-frames. In each I_1-frame, congestion $\leq I_1$ and dilation $\leq I_1$. Now, each I_1-frame is a subproblem of the original problem.

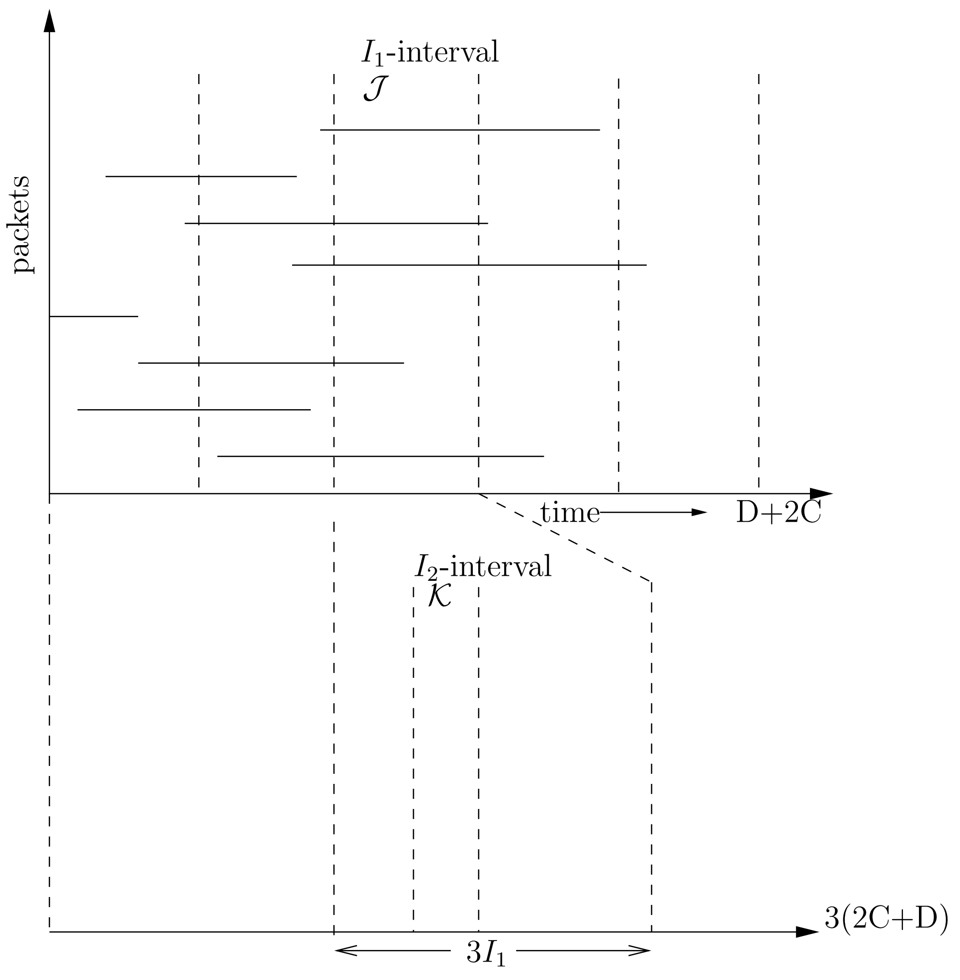

Figure 3: Recursion on I_1

Now, we apply the same algorithm recursively to the I_1-frame. We scale up each I_1-frame to $3I_1$ and give initial delays from the set $[2I_1]$ within each I_1-frame $\mathcal{J}$ (see Figure 3). We can similarly choose any I_2 interval $\mathcal{K}$ and define $A_e^{\mathcal{J}}(I_2)$ as the event when edge e has congestion $\geq I_2$ in any I_2-interval $\mathcal{K}$ within I_1-frame $\mathcal{J}$. As before, we set up random variables $X_1, X_2, \ldots, X_l$ and $X = \sum_{i=1}^{l} X_i$. We can argue similarly that for some $\mathcal{K}$ within $\mathcal{J}$,

$$P(X_i = 1) \leq \frac{I_2}{2I_1}$$

and hence as $l \leq I_1$ at this level of recursion,

$$E(X) \leq \frac{I_2}{2}$$

Therefore by Chernoff bounds,

$$\begin{aligned} P(X \geq I_2) &= P(X \geq (1+\epsilon)\frac{I_2}{2}) \\ &\leq e^{\frac{-I_2}{6}} \quad \leq e^{-3(\log\ I_1+1)} \\ &= \frac{1}{(eI_1)^3} \end{aligned}$$

And thus for any I_2-interval within $\mathcal{J}$,

$$\begin{aligned} P(A_e^{\mathcal{J}}(I_2)) &\leq (\text{no of } I_2\text{-intervals in } \mathcal{J}) \times \frac{1}{(eI_1)^3} \\ &\leq 3I_1 \times \frac{1}{(eI_1)^3} \\ &\leq \frac{3}{e^3 I_1^2} \end{aligned}$$

Also, similar to previous argument, Lovasz Lemma parameter $d \leq CD = I_1^2$. (Since in each I_1-frame congestion $C \leq I_1$ and dilation $D \leq I_1$). It turns out that $ep(d+1) < 1$ which enables us to apply Lovasz Local Lemma and arrive at the conclusion that in each I_1-frame, there exists a setting of inital delay values such that none of the $A_e^{\mathcal{J}}(I_2)$ events happen. We have thus shown the existence of a schedule in each $\mathcal{J}$, such that each I_2-frame within $\mathcal{J}$ has congestion $C \leq I_2$ and dilation $D \leq I_2$.

We can now choose any of the setting in every I_1-interval (which has been shown to exist) and carry out this recursive step successively. We do the recursion for levels $I_0, I_1, I_2 \dots I_k$ (where k is the total number of recursion steps) until the congestion comes to a constant value i.e. 18. At level I_k, since $C \leq I_k$ and $D \leq I_k$ and $I_k = 18$ which is a constant, at this level we can always schedule the packets in at most I_k^2 steps which is a constant (since scheduling problem in worst case is $O(CD)$). Also, at every level of recursion, we expanded the time space 3 times, so the total number of steps in scheduling $= I_0 \times 3^k \times I_k = 18.3^k(2C+D)$, which is $O((C+D)2^{\theta(\log^* max(C,D))})$ steps. And since, $2^{\theta(ln^*(C+D))}$ can be a small quantity as compared to C and D, we can approximate our solution at $O(C+D)$. ■

5.4 Embeddings

An embedding is a mapping from a logical network (guest G) to a real network (host H).

$$\phi : V(G) \rightarrow V(H)$$

A set of edges in guest network represent a set of paths in host network.

$$\phi(E(G)) = \{(\phi(u), \phi(v)) : (u, v) \in E(G)\}$$

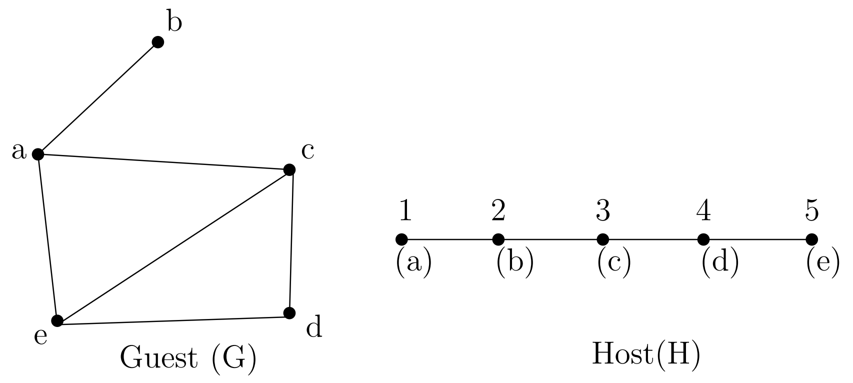

Figure 4: Example of a guest and host network and the embedding

5.4.1 Types of Embeddings

Consider the guest and the host network shown in Figure 4. There are several ways in which we can embed the guest network in the host network. We shall consider them one by one.

1. **Simple Embedding:** In a simple embedding, the mapping between the guest and the host network may be one-to-one or many-to-one. For example the embedding shown in Figure 4 is a simple embedding. Figure 5 shows another case of simple embedding when many guest nodes are mapped to a single host node. This kind of mapping leads to increased load in the host node that hosts more than one guests. As a result, the time taken by packets to travel through the network increases.

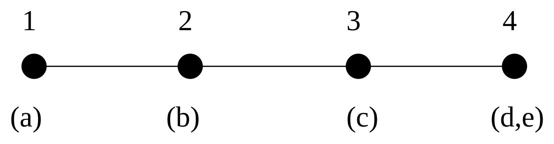

Figure 5: Simple embedding

2. **Multiple embedding:** In multiple embedding, one node in the guest

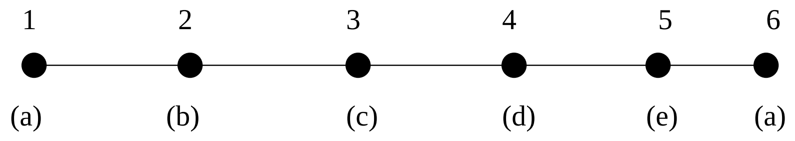

Figure 6: Multiple embedding

network may be mapped to many nodes in the host network. An example is shown in Figure 6. However, this setting leads to problem of maintaining consistency between identical guest nodes. Synchronization between these nodes becomes an issue.

3. **Dynamic Embedding:** Dynamic embeddings are those embeddings in which the guest to host mapping keeps on changing from time to time. In these kinds of embeddings there are additional overheads of maintaining consistency throughout the network about the current configuration. This is normally achieved by additional information packets relayed throughout the network according to some fixed protocol.

5.4.2 A Model of Parallel Computation

We are now in a position to describe a simple model for parallel computation. We build upon our earlier network model of synchronous packet transfer. Now, we add the criterion that in each step a node can also do some computation on the packets. Notice that this criterion adds several synchronization issues with the network, such as, before computation the input packet must arrive at the node and that output cannot be produced without input.
Parameters of Embeddings: Embeddings add an additional parameter, load (L), to the parameters congestion and dilation. Load may be defined as the maximum over all nodes, the number of packets that any node has to process. Hence the parameters of embeddings are:

1. Load(L)
2. Congestion(C)
3. Dilation(D)

An Example: Consider the $\sqrt{n}$ by $\sqrt{n}$ grid mesh G as a guest network. Suppose we want to map this mesh network to the serial network H of n nodes (Figure 7).We define a simple embedding as follows:

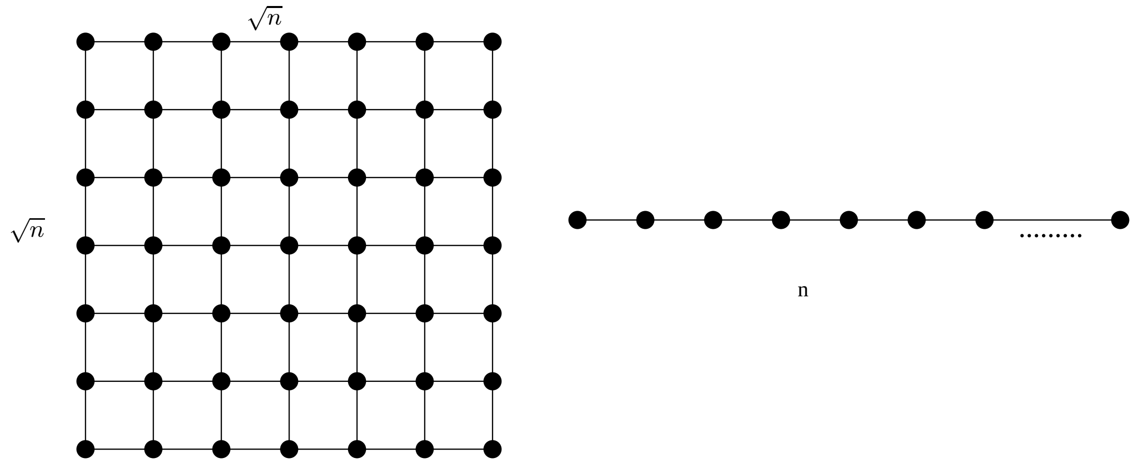

Figure 7: Embedding from mesh to line network

Map the first $\sqrt{n}$ row nodes of G to the first $\sqrt{n}$ nodes of H, the second $\sqrt{n}$ row nodes of to the next $\sqrt{n}$ nodes of H and so on. Thus each row maps to $\sqrt{n}$ contiguous nodes in H.

This is an example of a simple embedding. Clearly since this is a one-to-one mapping the load $L = 1$, congestion $C = \sqrt{n}$. Also, notice that while the horizontal edges are mapped to edges in the host, vertical edges are mapped to $\sqrt{n}$ length paths in the host. Hence the new dilation will be $D = \sqrt{n}$.

5.4.3 The Routing Problem: The Complete Picture

We are now in a position to define the complete routing problem. Given a path system P of host network H where,

$$P = \{(u, v, p) : u, v \in V(H), p \text{ is a path from u to v in H}\}$$

and an embedding $\phi : V(G) \to V(H)$, a routing problem is defined by

$$R_H^G = \{(x, p) : x \in \phi(E(G)), p \text{ is the path for } x \text{ in } P\}$$

Also note that it takes $O(L+C+D)$ steps in H for all packets to be successfully dispatched because it takes at most L steps to compute packets and at most O(C+D) steps for routing. ■

References

[1] F. Leighton, B. Maggs, and S. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.