Path coloring on the mesh - constructive version

Shira Kritchman

May 4, 2010

In this document I explain how Rabani's result [1] about path coloring on the mesh can be made constructive, using a result by Moser and Tardos [2].

The document follows closely Rabani's paper by its organization, and some of the text is copied as is from Rabani. The main differences from Rabani's paper are:

- Use of constructive version of LLL + necessary computations to show it applies
- Proof sketch for the constructive version of LLL
- Lemma (2), which was missing from Rabani's paper
- Many illustrations
- Completed / simplified a few other things

1 The Problem

An instance of the minimum path coloring problem (MPCP) specifies a graph G, and a list of pairs of vertices of G, (s_1, t_1), (s_2, t_2),... ,(s_n, t_n). These vertices are named *terminals* and the pairs are named *connections*. A solution to the instance specifies n paths, path i connecting s_i and t_i, and a color for each path. The assigned paths and colors must satisfy the condition that paths of the same color are edge disjoint. The objective function in the optimization version of the problem is to minimize the number of colors used.

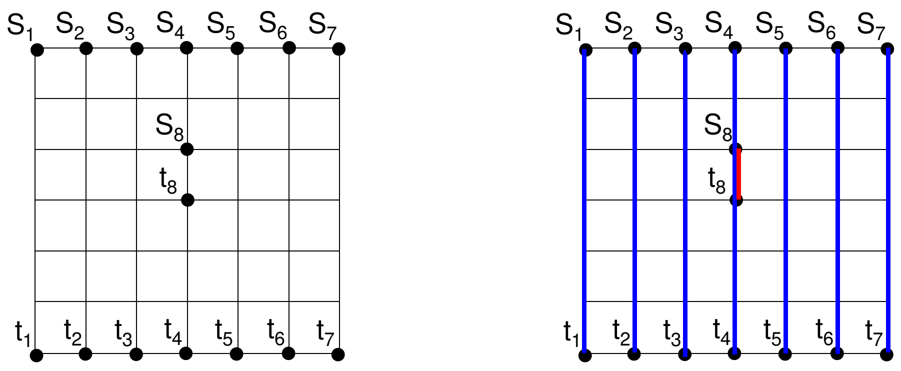
S_1 S_2 S_3 S_4 S_5 S_6 S_7
S_8
t_8
t_1 t_2 t_3 t_4 t_5 t_6 t_7
S_1 S_2 S_3 S_4 S_5 S_6 S_7
S_8
t_8
t_1 t_2 t_3 t_4 t_5 t_6 t_7

Figure 1: Left: an instance of the MPCP problem on a grid. Right: a possible optimal solution, using two colors.

We focus on the MPCP problem on an $N \times N$ grid. We denote such a grid by M_N. Figure 1-left is an example of an instance of the problem. It is quite clear that the minimal number of colors needed in this example is two. Figure 1-right suggests a solution using two colors.

2 Solution scheme

In this section we describe in short the entire flow of Rabani's solution. Given an MPCP instance on M_N, the first stage is a separation of the list of pairs of terminals into long and short connections, depending on the distance between them. This is discussed in Lemma (1). The heart of the solution is algorithm A, which is given as input an instance of MPCP on M_N with long connections only, and outputs an $O(1)$-approximation. The flow of the entire solution is as follows:

- **Input: MPCP instance on M_N, optimal solution:** c_0
- **Lemma (1): Separate the problem into O(1) problems and solve them using separate sets of colors**
  1. Short connections: O(1) disjoint smaller problems
     - Short short connections – optimal solution: c_1, find using exhaustive search

  - Long short connections – optimal solution: c_2, use A to find $\tilde{c_2} \leq O(1) \cdot c_2$

2. Long connections – optimal solution: c_3, use A to find $\tilde{c_3} \leq O(1) \cdot c_3$
3. Combined solution: $c_1 + \tilde{c_2} + \tilde{c_3} \leq O(1) \cdot c_0$

The flow of algorithm A is described by:

1. **Input: MPCP instance on M_N with long connections only**

   Optimal solution: k_0

2. **Reduce to MPCP instance on another graph, G_ν**

   Optimal solution: k_1

   Lemma (2): $k_1 \leq O(1) \cdot k_0$

3. **IP formulation**

   Optimal solution: k_1

4. **Solve corresponding LP**

   Optimal solution: k_2

   Clearly, $k_2 \leq k_1$. Solve using an approximation scheme and get k_3, where $k_2 \leq k_3 \leq O(1) \cdot k_2 \leq O(1) \cdot k_1$

5. **Round using Lovász Local Lemma**

   Rounding results in: k4

   Theorem (5): $k_4 \leq O(1) \cdot k_2$

6. **Translate the solution back to a solution on M_N**

   Translation results in: k_5

   Lemma (4): $k_5 \leq O(1) \cdot k_4$

7. **Altogether,** $k_0 \leq k_3 \leq O(1)k_0$, making k_3 an $O(1)$-approximation of k_0

The non constructive part is the use of the Lovász Local Lemma at stage (5) of algorithm A. We show that a recent result by Moser and Tardos [2] can be used to make this part constructive.

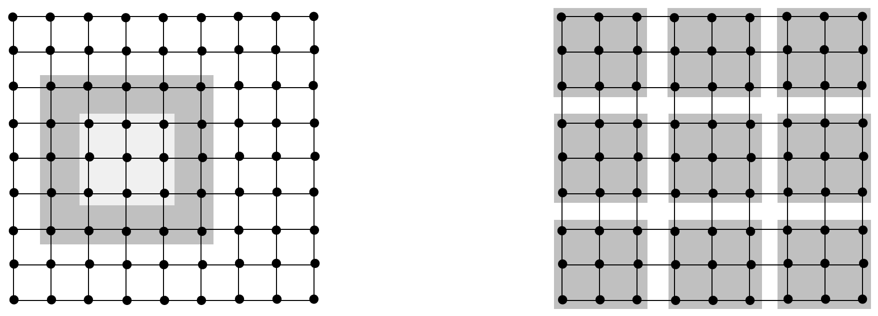

Figure 2: Left: a 1-neighborhood of a 3-tile located at (3,2). Right: a 3-partition of a 9×9 grid

3 The reduction

In this section we discuss shortly the first stage (separation into short and long connections), and the first and last stages of algorithm A (reduction to the graph G_ν and translation back into M_N). We start with a few definitions, illustrated by Figure 2.

Definition 1. *(Tiles): Let x, y, λ, μ be integers. A $\underline{\lambda\text{-tile located at } (x, y)}$ is the subset of vertices of M_N of the form $\{(i, j)|x \leq i \leq x + \lambda \text{ and } y \leq j \leq y + \lambda\}$. A $\underline{\mu\text{-neighborhood}}$ of a λ-tile τ located at (x, y) is the $(\lambda + 2\mu)$ tile located at $(x - \mu, y - \mu)$. The $\underline{\lambda\text{-partition}}$ of M_N is the collection of disjoint λ-tiles, including the one located at $(0, 0)$, which covers the nodes of M_N.*

The following lemma describes the partition into short and long connections.

Lemma 1. *(Lemma 1 in [1]) Let α, d be constants. Let A be a polynomial time MPCP $f(N)$-approximation algorithm to instances where each pair of terminals is at least $\alpha \ln^d N$ apart (for all N). Then, there is an $O(f(N))$-approximation algorithm to MPCP.*

Proof Sketch. (taken from Rabani with small changes) Call connections whose terminals are at least $\alpha \ln^d N$ apart *long*, and the other connections *short*. We use separate sets of colors for the long and short connections. We route the long connections using A. For the short connections, we use the fact that there is a fixed number of collections of

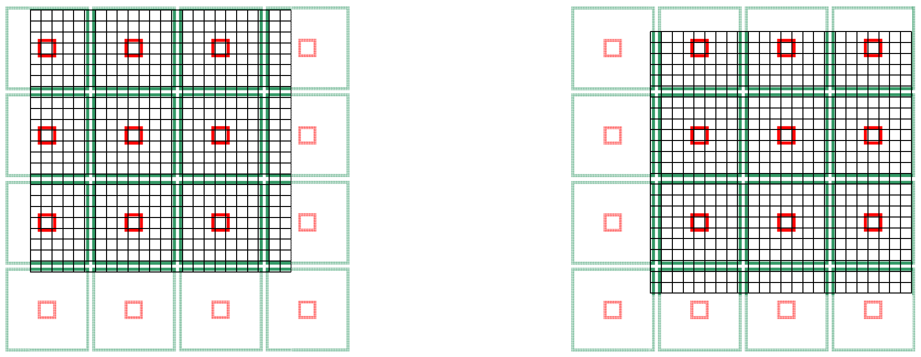

Figure 3: Two different collections of tiles (in red) with disjoint 3-neighborhoods (in green) on the same grid (in black)

$\beta \ln^d N$-tiles ($\beta > \alpha$ a constant), such that (i) for every short connection, both terminals are contained in a single tile in at least one of the collections; and (ii) the $3\beta \ln^d N$-neighborhoods of the tiles in a collection are all disjoint. Figure (3) shows two different collections of tiles with disjoint 3-neighborhoods.

These collections induce a partition of the short connections into classes. Each class contains the connections whose terminals are contained in a single tile of the corresponding collection. (If a connection fits into more than one collection, pick one arbitrarily.) We use a separate set of colors for each class of short connections. In each class, every $\beta \ln^d N$-tile is routed separately within its $3\beta \ln^d N$-neighborhood. (Colors are reused for each tile.) A simple argument, described graphically in figure (4) shows that we do not increase the number of colors needed to route the connections in a tile by restricting the routes to its neighborhood. Routing the connections in each tile is done by calling A recursively. Notice that one level of recursion is enough. The next level has to deal with poly($\ln \ln N$)-tiles, and there the problem can be solved by exhaustive search. $\square$

From now on we concentrate on finding an $O(1)$-approximation algorithm A for the long connections only. We now define the network G_N.

Definition 2. *(The network G_N): Let α, d be constants, and let $\lambda = \ln^d N$. The network G_N is a modification of the mesh M_N. Consider a λ-partition of a mesh M_N. Remove the edges connecting between different tiles. For each tile, add a representative node, and connect it to the boundary nodes of the tile by an edge with capacity 1. Also, connect*

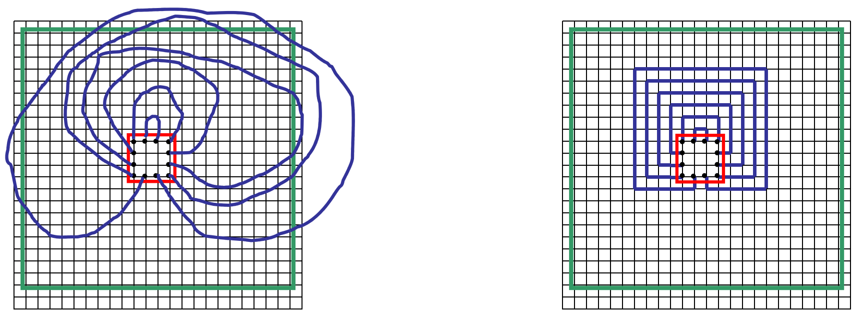
Figure 4: Left: an arbitrary routing of pairs in the inner tile. Right: the routing of the same pairs can be done inside the 3-neighborhood of the inner tile.

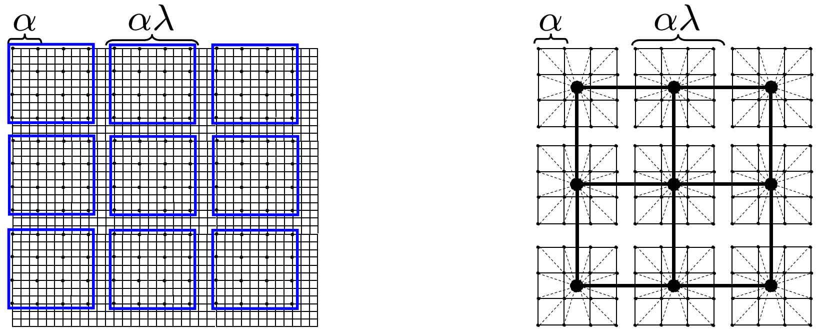

Figure 5: Left: the graph M_N. The blue rectangles are $\alpha\lambda$ tiles. Right: the graph G_ν constructed from M_N.

each two representative nodes of adjacent tiles by an edge with capacity λ. Such an edge 'replaces' the λ edges that connect two adjacent tiles in M_N.

We denote the edges with capacity 1 (inside tiles, and between tile and representative) *thin edges*, and denote the edges with capacity λ *fat edges*.

Let $\nu = N/\alpha$. Given an instance of MPCP on M_N we reduce it into an MPCP instance on G_ν. That is, G_ν is constructed by downsampling M_M to a coarser mesh M_ν, and from there constructing G_ν as in the definition. A terminal in M_N is mapped to the leaf corresponding to the tile containing the terminal in the α-partition of M_N. Figure (5) illustrates the construction of $G_{N/\alpha}$.

The following Lemma guarantees that by reducing the instance to the graph G_ν, the optimal number of colors doesn't increase by more than a constant factor. The proof uses Theorem (5), which will be presented in section (4).

Lemma 2. *(Missing from Rabani's paper) Given an MPCP instance on M_N, let c be the optimal solution. Consider the equivalent MPCP instance on G_ν, achieved by the reduction mentioned above. Let k be the optimal solution for this instance. Then, $k < O(1) \cdot c$.*

Proof. Consider an optimal integral solution f to an MPCP instance on M_N, using c colors. The solution can be translated into a solution on G_ν in a straightforward way. This solution, however, may have a flow of up to α on some of the thin edges, and a flow of $\alpha\lambda$ on the fat edges. We thus duplicate each color α times and decrease the flow by a factor of α. This gives a valid (though non-integral) solution to the LP of definition (1), which uses αc colors. From Theorem (5) we have that $k \leq O(1)\alpha c = O(1)c$. $\square$

The following two Lemmas implicate that any solution on G_ν can be converted to a solution to the original problem on M_N, while increasing the number of colors by a constant factor.

Lemma 3. *(A simplification of Lemma 2 in [1]) Given an MPCP instance on G_N, consider solutions satisfying the following additional constraint: For each color c, for each leaf, at most one path colored c leaves the leaf. Then the optimal solution with this additional constraint is within a factor of $O(1)$ of the optimal solution without this constraint.*

Proof. Given a feasible solution to MPCP on G_N, we convert it to a feasible solution satisfying the additional constraint as follows: For each color c, consider the graph whose nodes are the paths colored c with two paths adjacent if they leave the same terminal. The maximum degree of a node in this graph is bounded by 6, so its chromatic number is at most 7. So, by replacing each color in the original solution by at most 7 distinct colors (according to the coloring of the nodes of the constructed graph), we impose the additional constraint. $\square$

Lemma 4. *(This is Lemma 3 in [1]) Consider any set of edge-disjoint paths in G_ν connecting leaves. Assume that any leaf is a terminal of at most one path. Further assume that for any path, its two terminals are located in two λ-tiles whose representatives are at distance at least two apart. Consider the tiles in the α-partition of M_N which correspond to terminals. Suppose that α is a sufficiently large constant. Then, we can choose any single node in each such tile, so that the collection of pairs of nodes in pairs of tiles that correspond to pairs of terminals connected in G_ν can all be connected by edge-disjoint paths in M_N.*

Proof Sketch. (taken from Rabani, where a more detailed proof sketch can be found) The idea is to simulate G_ν on M_N in a general way (that is, that doesn't depent on the problem instance). Some small adjustments might be needed, depending on the instance. Choose the nodes in the λ-tiles that correspond to terminals. G_ν is simulated as follows. Its leaves are simulated by the tiles in the α-partition of M_N, and its representatives are simulated by the tiles in the $\alpha\lambda$-partition of M_M.

Call the nodes in M_N which lie on the edge of an $\alpha\lambda$ tile, and are also corners of an α tile, *edge-nodes* (these are the nodes connected to representative nodes in figure 5). Also, call the nodes in G_ν which are connected to a representative, *edge-nodes.* In G_ν, We have three connection types to consider: (1) Connections between a leaf and an edge-node (these connections have capacity 1), (2) connections between an edge-node and a representative node (these connections have capacity 1), (3) Connections between two adjacent representative nodes (these have capacity λ).

To simulate type (1) connections we build a network which connects any small α-tile, to any edge-node of the bigger $\alpha\lambda$-tile surrounding it. This is called *the escape network.* To simulate types (2)+(3) connections we build a network which connects any edge-node of an $\alpha\lambda$-tile, to any other edge-node of the same tile which lies on a different side, called the *high capacity network*, and a network which connects any edge-node of an $\alpha\lambda$-tile, to any other edge-node of the same tile (even if they lie on the same side), called the *redirection network*,

If α is a sufficiently large, then the edges of these networks can all be embedded as mutually edge-disjoint paths, and furthermore, none of the networks block the chosen

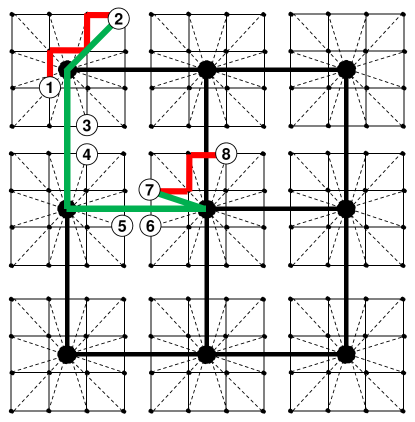

Figure 6: Simulating a path on G_ν (in red and green) using the simulation networks on M_N is done as follows: (1-2) Escape network, (2-3) redirection network, (3-4) edge between them, (4-5) high-capacity network, (5-6) edge between them, (6-7) redirection, (7-8) escpse network.

terminal nodes (to this end we might need to adjust the networks according to the problem instance).

Using these networks, we can simulate any path on G_ν. See figure 6 for an example.

□

4 Main

The MPCP problem on the graph G_N can be formulated as the following (exponential size) integer linear program, denoted $IP[c]$:

Definition 3. *Minimize γ subject to:*

$$\begin{array}{ll} \sum_{j,k} f^i_{j,k} \geq 1 & \forall i \in \{1, 2, \ldots, n\}; \\ \sum_{i,j} f^i_{j,k} q^i_j(e) \leq \gamma c(e) & \forall k \in \{1 \ldots, c\}, \forall e \in E(G); \\ f^i_{j,k} \in \{0, 1\} & \forall i, j, k \end{array} \tag{1}$$

where $f^i_{j,k}$ is the flow of commodity i over path j with color k, q^i_j denotes the characteristic function of the jth path of commodity i, and $c(e) \in \{1, \lambda\}$ is the capacity of the edge e.

The MPCP instance can be solved using c colors iff there is a solution to $IP[c]$ with $\gamma \leq 1$.

We relax the last set of conditions to $f^i_{j,k} \in [0,1]$, obtaining a linear program, which we denote $LP[c]$. Clearly, if the optimal solution to $IP[c]$ is ≤ 1, so is the optimal solution to $LP[c]$. The optimal solution to $LP[c]$ can be found using an approximation scheme, as explained later in this section.

For an integer B, we call a solution *B-integral* if all paths have weight $\frac{1}{B}$ (or zero). A solution with $\gamma < 1$ is denoted *feasible.* A 1-integral feasible solution is denoted *proper.*

Let $\gamma^*_{IP[c]}$ denote the optimal solution to $IP[c]$, and let $\gamma^*_{LP[c]}$ denote the optimal solution to $IP[c]$. We show the following theorem:

Theorem 5. *(Theorem 5 in Rabani) There exists a constant μ such that if $\gamma^*_{LP[c]} \leq 1$, then $\gamma^*_{IP[\mu c]} \leq 1$.*

Before turning to the proof of theorem (5), we introduce some notation. Define $G = G_N$ by taking $\lambda = \ln^2 N$. Fix c, and let $\{\gamma, f\}$ be a solution to $LC[c]$ with $\gamma \leq 1$.

Let $\bar{\gamma}(f)$ denote an upper bound on the capacity utilization for edges connecting representatives, and let $\hat{\gamma}_\ell(f)$ denote an upper bound on the maximum over all rectangles with boundary capacity at least ℓ contained in any single $\lambda \times \lambda$ tile, over all colors, of the amount of flow of that color leaving the rectangle divided by the total capacity of edges leaving the rectangle. Note that $\bar{\gamma}$ and $\hat{\gamma}$ denote some upper bound, not necessarily a tight one. Also, let $\eta_\ell(f)$ be a common upper bound on $\bar{\gamma}(f), \hat{\gamma}_\ell(f)$.

The following Lemma explains why we consider these quantities, by showing that if we bound them well enough, then we can convert the solution to a solution with bounded γ by solving a maximal flow problem:

Lemma 6. *(A close version of Lemma 8 in Rabani) Given a 1-integral solution $\{\gamma, f\}$ to $LP[c]$, with $\eta_1(f) \leq 1$, one can find in polynomial time a 1-integral solution $\{1, f''\}$ (that is, a proper solution).*

Proof. (This is a more detailed version of Rabani's proof) First note that $\bar{\gamma}(f) \leq \eta_1(f) \leq 1$ is guaranteed by definition, so we only need to worry about the paths inside the λ-tiles, and leave the portion of the paths that use the fat edges as is. Consider one λ-tile and one color c. Consider the set of terminals S in this tile that use the color c. We have to show that we can find $|S|$ disjoint paths that connect the members of S to the tile's boundary. Thus, we consider the tile as an independent graph. We add a vertex s and connect it to each of the terminals in S. We consider the boundary vertices to be one vertex denoted t. We give all edges capacity of 1. We consider the cut-flow problem on this graph. Claim: the minimal $s-t$ cut is $|S|$. Proof: Let C be a minimal $s-t$ cut. We want to show that each $v \in S$ contributes at least one edge to C. Consider a terminal $v \in S$. If the edge (s,v) is in C, then v contributes this edge. Otherwise, v is connected to s by an edge which is not in C. Thus, it must be disconnected from t using tile edges. The edges in C that are part of the original grid can be partitioned to a set of disjoint closed curves. Consider one such closed curve containing k terminals. Denote its length by ℓ. A closed curve of length ℓ in a grid is always contained in a rectangle R of length $\leq \ell$. By definition of $\hat{\gamma}_1(f)$, the number of paths leaving R is $\leq \ell\hat{\gamma}_1(f) \leq \ell$. Thus, the number of terminals inside a connected component of length ℓ is $\leq \ell$. Thus, for each connected component, each terminal inside the component can be associated with a unique edge.

We now see that each terminal contributes at least 1 to the number of edges in C, thus $|C| \geq |S|$. Thus the maximal flow is $\geq |S|$. Since the edge capacities are integral, the maximal integral flow equals the maximal flow. An integral flow with total flow $|S|$ must have a disjoint path for each terminal, because we must have flow of 1 going from s to each of its $|S|$ neighbors, the terminals S. Thus, by solving the maximal flow problem, we find a proper solution. □

The strategy for obtaining an integral solution has five steps:

1. **Find a near-optimal feasible solution $\{\gamma_1, f_1\}$ to $LP[c]$, with minimal c**

   This can be done using binary search on c and an approximation scheme for multicommodity flow based on Lagrangian relaxations (e.g. [3], which can be used since we have small integral capacities). We are guaranteed to get a near optimal solution (sufficient for our purposes) with a polynomial number of flow paths carrying the same amount of flow each.

   Problem: it is not clear to me why the approximated minimal number of

colors c is an $O(1)$-approximation of the truly minimal number of colors for a feasible solution of the LP. This is used in step 4 of algorithm A's flow.

2. **Let B_0 be a constant. Convert the optimal solution into a feasible solution $\{\gamma_2, f_2\}$ to $LP[4B_0c]$ satisfying the following constraints: $\gamma_2 \leq (2B_0)^{-1}$, and f_2 is h-integral for some h. Since f_2 is computed in polynomial time, we have $h \geq 2^{-N^{O(1)}}$.**

   This step is simple and we don't explain it here.

3. **Converting $\{\gamma_2, f_2\}$ into a B_1^{-1}-integral solution $\{\gamma_3, f_3\}$, where $B_1 \leq B_0$, through a series of stages, such that the increase in $\eta_1(f)$ is restricted in each step, and throughout the steps it doesn't increase by more than a factor of two. Thus, $\eta_1(f_3) \leq B_0^{-1}$.**

   In this step the number of colors increases by a constant factor. This can be done using Lemma (7).

4. **For each commodity, selecting an arbitrary path and giving it weight one. This gives a 1-integral solution $\{\gamma_4, f_4\}$ with $\eta_1(f_4) \leq B_1 \cdot \eta_1(f_3) \leq 1$.**

5. **Find a proper solution $\{\gamma_5, f_5\}$ through Lemma (6).**

Lemma 7. *(Lemma 6 in [1]) Let B_0, $a < 2B_0$ be sufficiently large constants. Let $B \geq B_0$ and let $\ell = O(1)$ be a positive integer. Let η be such that $\eta\ell \geq a \ln^{-1} B$. Given a B^{-1}-integral solution $\{\gamma, f\}$ to $LP[c]$ with $\bar{\gamma}_\ell(f), \hat{\gamma}_\ell(f) \geq \eta$, we can find in polynomial time a $\ln^{-3} B$-integral solution $\{\gamma', f'\}$ to $LP[c]$ with $\eta_\ell(f') \leq (1+\varepsilon)^2\bar{\gamma}$, where $\varepsilon = 1/\sqrt{\ln B}$.*

Note: The constraint $\ell = O(1)$ is missing from Lemma 6 in [1], where ℓ is defined as just an $\ell > 1$ integer. This constraint is necessary for the Lemma to be correct, but adding it doesn't harm the proof, because Lemma (7) is only used with $\ell = 1$ or $\ell = B_0$. The constraint $a < 2B_0$ is also missing from [1].

To prove Lemma (7) we use the constructive version of the general LLL by Moser and Tardos.

This is how Lemma (7) is used to implement step (3):

1. Start with the h-integral solution $\{\gamma_2, f_2\}$, where $\gamma_2 \leq (2B_0)^{-1}$.

2. Repeatedly apply Lemma (7) with $\ell = 1$, as long as $B \geq e^{2aB_0}$. This takes $O(\log^* N)$ iterations.

   For the i'th step we apply Lemma (7) with $\eta = \eta^i = (1+\varepsilon)^{2i}\,(2B_0)^{-1}$. As we will see, through the steps η does not grow by more than a factor of two, thus we always have $\eta \leq B_0^{-1}$. Denote by f_2^i the solution after i invocations of Lemma (7). The conditions for using Lemma (7) are always met:

   (a) $\underline{B \geq B_0}$: $B \geq e^{2aB_0} \geq B_0$.

   (b) $\underline{\eta\ell \geq a\ln^{-1} B}$: for $\ell = 1$, $\eta = \eta^i \geq (2B_0)^{-1} = a\,(2aB_0)^{-1} \geq a\ln^{-1} B$.

   (c) $\underline{\eta_1(f_2^i) \leq \eta^i}$, that is, η^i is a proper upper bound on the capacity utilization of the flow f_2^i that we get after i iterations: Recall that $\gamma_2 \leq (2B_0)^{-1}$. Thus, $\eta_1(f_2) \leq (2B_0)^{-1}$. Lemma (7) guarantees that $\eta_1(f_2^i) \leq \eta_1(f_2)(1+\varepsilon)^{2i} \leq \eta^i$.

3. In the following steps we will no longer be able to use $\ell = 1$ but only $\ell = B_0$. Thus, in this step, we take care of all the 'small' rectangles. The previous step left us with a solution f_2' with $B < e^{2aB_0}$, and with $\hat{\gamma}_{B_0}(f_2') \leq \eta_1(f_2') \leq B_0^{-1}$. This means that for each rectangle Q with boundary capacity B_0 and for each color k there are at most e^{2aB_0} fractional paths of color k that leave Q. By duplicating each color a (large) constant number of times, we can partition the paths among the duplicates so that at most a single path leaves each such rectangle Q. Note that this step does not increase η_ℓ for any ℓ.

4. Repeatedly apply Lemma (7) with $\ell = B_0$. After $O(1)$ iterations we end up with a $(B_0)^{-1}$-integral solution (B_0 is chosen large enough so that this is possible). The conditions for using Lemma (7) are always met:

   (a) $\underline{B \geq B_0}$: By definition.

   (b) $\underline{\eta\ell \geq a\ln^{-1} B}$: for $\ell = B_0$, $\eta^i\ell \geq (2B_0)^{-1}\,B_0 = 2 \geq a\ln^{-1} B_0 \geq a\ln^{-1} B$.

   (c) $\underline{\eta_{B_0}(f_2^i) \leq \eta^i}$: Same as before.

5. We now have a B_0-integral solution $\{\gamma_3, f_3\}$ with $\eta(f_3) \leq B_0^{-1}$, and step 3 is completed.

Throughout the stages of step 3 we increased η_{B_0} by a factor of at most

$$\prod_{i=1}^{t}\left(1+\frac{1}{\sqrt{\ln B_i}}\right)^2,$$

where the B_i's are given by the recurrence $B_i = e^{\sqrt[3]{B_{i-1}}}$, and $t = O(\log^* N)$. For a sufficiently large choice of B_0, this product is at most 2.

4.1 Proving Lemma (7)

Consider the following randomized rounding procedure: select path q_j^i and color k with probability $f_{j,k}^i(1+\varepsilon)\ln^3 B$, and assign to it a new flow $\bar{f}_{j,k}^i = ln^{-3}B$, independently for all i, j, k. This defines a probability space over $ln^{-3}B$-integral (not necessarily feasible) flows. We shall show that there is a point in this probability space that is both feasible and within the claimed capacity utilization bounds. In proving this we use the following version of the Local Lemma:

Theorem 8. *[Constructive Lovász Local Lemma - Constructive Version] Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables, where event A is determined by vbl(A). Let $G = (\mathcal{A}, E)$ be a dependency graph for the set of events $\mathcal{A}$, where $(A,B) \in E$ iff $A \neq B$ and vbl$(A) \bigcap$ vbl$(B) \neq \Phi$. Then, if there exists an assignment of reals $x : \mathcal{A} \to \{0,1\}$ such that*

$$\Pr[A] \leq x(A) \prod_{(A,B)\in E} (1 - x(B)) \quad \forall A \in \mathcal{A}, \tag{2}$$

then there exists an assignment of values to the variables $\mathcal{P}$ not violating any of the events in $\mathcal{A}$.

Moreover, there exists a randomized algorithm that resamples an event $A \in \mathcal{A}$ at most an expected $x(A)/(1-x(A))$ times before it finds such an evaluation. Thus the expected total number of resampling steps of the algorithm is at most $\sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$.

See the proof sketch at the appendix.

We rephrase Lemma (7) in the terminology of Theorem (8). The set of mutually independent random variables $\mathcal{P}$ is defined as the set $\xi_{j,k}^i$, where $\xi_{j,k}^i$ denotes whether path q_j^i with color k was chosen in the randomized procedure. Define the following events:

- Let F_i denote the event that there are less than $\ln^3 B$ flow paths picked for commodity i.
- For a fat edge e, let $A_{e,k}$ denote the event that the total flow of color k over e is more than $(1+\varepsilon)^2\eta\lambda$ after rounding.
- For a rectangle Q with boundary capacity $\geq \ell$ contained in a big tile, let $E_{Q,k}$ denote the event that the total flow of color k out of Q is more than $(1+\varepsilon)^2\eta\nabla Q$, where ∇Q denotes the total capacity of the edges leaving Q.

We define $\mathcal{A} = \{F_i\}_i \bigcup \{A_{e,k}\}_{e,k} \bigcup \{E_{Q,k}\}_{Q,k}$. An assignment for $\mathcal{P}$ which doesn't violate any of the events in $\mathcal{A}$ gives a $\ln^{-3} B$-integral flow which fulfills the conditions in Lemma (7). We now have to show that the conditions in Theorem(8) are all met; that is, we have to find an assignment $x : \mathcal{A} \to \{0,1\}$ which satisfies Eq. (2). This is exactly the same thing that Rabani showed in his paper, by using the assignment: for all i, $x_{F_i} = X_F = B^{-2}$, for all e,k, $x_{A_{e,k}} = x_A = B^{-1}N^{-2}$, for all Q,k, $x_{E_{Q,k}} = x_Q = B^{-2}e^{-\nabla Q}$. Rabani showed the computations only for events of type F and mentioned that for the other events this is similar.

To use the algorithm suggested by Moser and Tardos we also have to show that the number of random variables defining the bad events is polynomial, that the number of bad events is polynomial, that to check whether a bad event occurs takes polynomial time, and that to resample the random variables associated with a bad event takes polynomial time. Most of these are trivial, but we show how to bound the number of bad events:

- # of events of type F: n
- # of events of type A: $2N^2c$.
- # of events of type E: for $\nabla Q = m$, $\leq N^2mc$. m can go from ℓ to $4\ln^2 N$, so altogether we have a bound of $O(N^2 \ln^4 Nc)$.

Moser and Tardos also suggest a parallel version of their algorithm. The conditions for using it and its properties are stated in the following Theorem:

Theorem 9 (Constructive Lovász Local Lemma - Parallel Version)**.** *Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables, where event A is determined by vbl(A). Let $G = (\mathcal{A}, E)$ be a dependency graph for the set of events $\mathcal{A}$, where $(A, B) \in E$ iff $A \neq B$ and vbl$(A) \bigcap$ vbl$(B) \neq \Phi$. Then, if there exists an assignment of reals $x : \mathcal{A} \to \{0, 1\}$ such that*

$$\Pr[A] \leq (1-\delta)x(A) \prod_{(A,B)\in E} (1 - x(B)) \quad \forall A \in \mathcal{A}, \tag{3}$$

then the parallel version of our algorithm takes an expected $O\left(\frac{1}{\delta} \log \sum_{a\in\mathcal{A}} \frac{x(A)}{1-x(A)}\right)$ steps before it finds an evaluation violating no event in $\mathcal{A}$.

We don't give here a proof of this Theorem. In the appendix we show computations which prove that the parallel version can be applied to our problem, with $\delta = \frac{1}{2}$.

Note: the conditions for applying the deterministic version of the algorithm are *not* met.

5 Proving LLL

The original proof of the Local Lemma, given in 1975 by László Lovász and Paul Erdős [4], is non-constructive and does not yield an efficient procedure for searching the probability space for a point with the desired property. Moser and Tardos were the first to give a full algorithmic proof that provides such a procedure. Here we sketch the proof given in [2], while trying to give some intuition.

Let $\Gamma(A)$ be the set of neighbors of A in G, and let $\Gamma^+(A) = \Gamma(A) \bigcup \{A\}$. If $B \in \Gamma^+(A)$ we say that A and B are inclusive neighbors.

Proof Sketch for Theorem (8). We will show that the following randomized algorithm fulfills the conditions listed in the Theorem:

```
    function LLL($\mathcal{P}, \mathcal{A}$)
```

```
for all $P \in \mathcal{P}$ do
    $v_P \leftarrow$ a random evaluation of $P$;
while $\exists A \in \mathcal{A}$ is violated when $(P = v_P : \forall P \in \mathcal{P})$ do
    pick an arbitrary violated event $A \in \mathcal{A}$;
    for all $P \in \mathrm{vbl}(A)$ do
    $v_P \leftarrow$ a new random evaluation of $P$;
return $(v_P)_{P \in \mathcal{P}}$
```

We fix some (randomized or deterministic) procedure for picking an arbitrary violated event.

To describe an execution of the algorithm we define the *execution trace.* This is the list $R : \mathcal{N} \to \mathcal{A}$ of violated events that were picked by the algorithm, in the order in which they were picked.

A *witness tree* is a pair (τ, σ_τ), where τ is a rooted tree and $\sigma_\tau : V(\tau) \to \mathcal{A}$ is a labelling of its vertices with events. We denote $[v] = \sigma_T(v)$. A witness tree is said to be *proper* if distinct children of the same vertex always receive distinct labels. Given the trace, we will associate with each resampling step t carried out a witness tree $\tau_R(t)$ that can serve as a 'justification' for the necessity of that correction step.

We construct $\tau_R(t)$ step by step, starting with a root vertex labelled $R(t)$. We follow the trace backwards from t. After j steps, we consider $R(t-j)$. If $R(t-j)$ is an inclusive neighbor (in the dependency graph) of some event which already appears in the graph, we add a vertex labelled $R(t-j)$ to the graph, as far from the root as possible. The only restriction is that it should be added as a child of a vertex which is its exclusive neighbor (in the dependency graph). For an example of building such a tree, see figure (7).

We say that the witness tree τ *occurs* in the trace R if there exists $t \in \mathcal{N}$ such that $\tau_R(t) = \tau$. We will use the following claim without a proof:

Claim. *Let τ be a fixed witness tree and R the (random) trace produced by the algorithm.*
(i) If τ occurs in R, then τ is proper.
(ii) The probability that τ appears in R is at most $\prod_{v \in V(\tau)} \Pr([v])$.

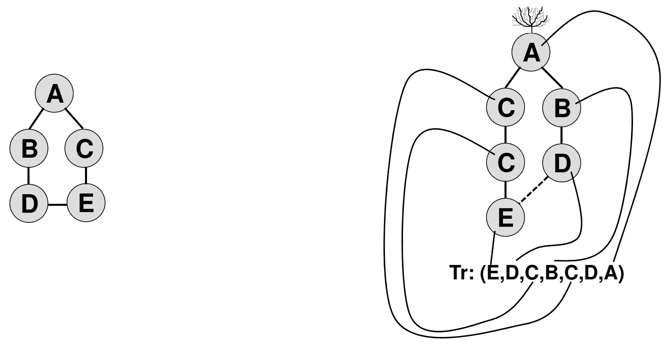

Figure 7: Left: dependency tree for the events A, B, C, D, E. Right: witness tree compatible with the trace $EDCBCDA$. The dashed line offers an alternative way to connect the vertex denoted E.

Part (i) is rather straightforward. To convince yourself with part (ii), observe that in a witness tree, in each level, all the events are independent of each other. Consider the deepest (farthest from the root) level in a witness tree. For an event A in the deepest level, when the algorithm picks A, the assignment for $\mathrm{vbl}(A)$ is the original assignment given by the algorithm. Thus, all the events in this level are violated in the original assignment sampled by the algorithm. Since these events are independent, the probability for this to happen is $\leq \prod_{v \text{ in deepest level}} \Pr([v])$. After resampling the deepest level, the same holds for the assignment of the next level, and so on until we reach the root.

For an event $A \in \mathcal{A}$, let N_A be the random variable which counts the number of occurrences of A in the trace. We want to compute $\mathbb{E}[N_A]$. We denote by $\mathbb{T}_A$ all the proper witness trees rooted at A. For any $\tau \in \mathbb{T}_A$, let ξ_τ be the characteristic function indicating whether τ occurs in the trace. An important observation is that a tree cannot occur more than once in a trace. This is true because for the tree $\tau_R(t)$ associated with time t, its root is labelled $R(t)$, and $R(t)$ appears in the tree exactly the same number of times that $R(t)$ appears in the t-suffix of R. Thus, for each occurrence of an event A in the trace, there is a unique tree $\tau \in \mathbb{T}_A$ that occurs in R. Thus, $N_A = \sum_{\tau \in \mathbb{T}_A} \xi_\tau$, and

$$\mathbb{E}[N_A] = \sum_{\tau \in \mathbb{T}_A} \Pr[\tau \text{ occurs in the trace}] \leq \sum_{\tau \in \mathbb{T}_A} \prod_{u \in V(\tau)} \Pr([u]) \leq \sum_{\tau \in \mathbb{T}_A} \prod_{u \in V(\tau)} x'([u]),$$

where for an event B, $x'(B) = x(B) \prod_{C \in \Gamma(B)} (1 - x(C))$, and the last inequality follows from the condition (2) we have on x.

We now wish to bound the last term. To this end, consider the Galton-Watson branching process for generating witness trees rooted at A. We start with a single vertex labelled A. In each subsequent round, we consider each vertex v produced in the previous round independently. For each event $B \in \Gamma^+([v])$, we add a vertex labelled B with probability $x(B)$. This process creates proper witness trees, and a tree τ is created with probability

$$p_\tau = \frac{1-x(A)}{x(A)} \prod_{v \in V(\tau)} x'\left([v]\right),$$

and hence $\sum_{\tau \in \mathbb{T}_A} p_\tau \leq 1$. Therefore,

$$\sum_{\tau \in \mathbb{T}_A} \prod_{v \in V(\tau)} x'\left([v]\right) = \sum_{\tau \in \mathbb{T}_A} \frac{x(A)}{1-x(A)} p_\tau = \frac{x(A)}{1-x(A)} \sum_{\tau \in \mathbb{T}_A} p_\tau \leq \frac{x(A)}{1-x(A)}.$$

We conclude that

$$\mathbb{E}[N_A] \leq \sum_{\tau \in T_A} \prod_{u \in V(\tau)} x'([u]) \leq \frac{x(A)}{1-x(A)}.$$

□

6 Computations

In order to use the parallel version of the GLLL algorithm, one needs to show that for some $\delta > 0$ (which we want to be large), for any bad event A,

$$\Pr[A] \leq (1-\delta) x(A) \prod_{B \in \Gamma_{\mathcal{A}}(A)} (1 - x(B)). \tag{4}$$

where we use the same assignments as before: for all i, $x_{F_i} = X_F = B^{-2}$, for all e, k, $x_{A_{e,k}} = x_A = B^{-1}N^{-2}$, for all Q, k, $x_{E_{Q,k}} = x_Q = B^{-2}e^{-\nabla Q}$. We show here that we can have $\delta = \frac{1}{2}$.

To simplify notation, let $f(\varepsilon) = \frac{e^\varepsilon}{(1+\varepsilon)^{1+\varepsilon}}$. We will use the following inequality, which holds for $0 < \varepsilon < \frac{1}{2}$:

$$f(\varepsilon) \leq e^{-\frac{\varepsilon^2}{4}}. \tag{5}$$

Proof. First note that $f(\varepsilon) = e^{\varepsilon - \ln(1+\varepsilon)(1+\varepsilon)}$. Using the inequality $\ln(1+x) > x - \frac{x^2}{2}$ we get

$$\begin{aligned}
\varepsilon - \ln(1+\varepsilon)(1+\varepsilon) &< \varepsilon - \left(\varepsilon - \frac{\varepsilon^2}{2}\right)(1+\varepsilon) \\
&= \frac{\varepsilon^2}{2}(-1+\varepsilon) \\
&< -\frac{\varepsilon^2}{4}
\end{aligned}$$

where the last inequality uses $\varepsilon < \frac{1}{2}$. □

We will use the following bounds, given by Rabai, on the probabilities of the bad events:

$$\begin{aligned}
\Pr[F_i] &< \exp\left[-\ln^2 B/2(1+\varepsilon)\right] \\
\Pr[A_{e,k}] &< f(\varepsilon)^{\bar{\gamma}\ln^2 N \ln^3 B} \\
\Pr[E_{Q,k}] &< f(\varepsilon)^{\hat{\gamma}\nabla Q \ln^3 B},
\end{aligned}$$

where $\varepsilon = 1/\sqrt{\ln B}$. The bounds are achieved by applying Chernoff bounds.

6.1 Computation for type F events

In [1] we have the following computation:

$$\begin{aligned}
& x_F(1-x_A)^{\frac{4N^2}{\ln^4 N}} \prod_{\nabla Q=\ell}^{4\ln^2 N} (1-x_{\nabla Q})^{2(\nabla Q)^3 B} \\
\geq\ & B^{-2}\left(1 - \frac{5}{\ln^4 N}\right)\left(1 - \frac{18}{B}\right)
\end{aligned}$$

We continue from here:

$$\begin{aligned}
& B^{-2}\left(1-\frac{5}{\ln^4 N}\right)\left(1-\frac{18}{B}\right) \\
= \quad & \left[e^{-2}\left\{\left(1-\frac{5}{\ln^4 N}\right)\left(1-\frac{18}{B}\right)\right\}^{1/\ln B}\right]^{\ln B} \\
\geq \quad & \left[e^{-2}e^{-\ln 2}\right]^{\ln B} \\
\geq \quad & \left[2e^{-\frac{\ln B}{4}}\right]^{\ln B} \\
\geq \quad & \left[2e^{-\frac{\ln B}{2(1+\varepsilon)}}\right]^{\ln B} \\
\geq \quad & 2e^{-\frac{\ln^2 B}{2(1+\varepsilon)}} \\
\geq \quad & 2\Pr[F_i]
\end{aligned}$$

6.2 Computation for type A events

We now show that Eq. (4) holds for events of type $A_{e,k}$ with $\delta = \frac{1}{2}$.

From Eq. (5) it follows that

$$\Pr[A_{e,k}] \leq f(\varepsilon)^{\bar{\gamma}\ln^2 N \ln^3 B} \leq e^{-\frac{\varepsilon^2}{4}\bar{\gamma}\ln^2 N \ln^3 B} = e^{-\frac{\bar{\gamma}\ln^2 N \ln^2 B}{4}}$$

We compute:

$$\begin{aligned}
& x_A \prod_{(A,B)\in E} (1-x(B)) \\
\geq\ & x_A(1-x_A)^{4N^2/\ln^4 N}(1-x_F)^{B\bar\gamma\ln^2 N} \prod_{\nabla Q=\ell}^{4\ln^2 N} (1-x_Q)2(\nabla Q)^3\bar\gamma\ln^2 N \\
=\ & \frac{1}{BN^2}(1-B^{-1}N^{-2})^{4N^2/\ln^4 N}(1-B^{-2})^{B\bar\gamma\ln^2 N}\left[\prod_{\nabla Q=\ell}^{4\ln^2 N}(1-B^{-2}e^{-\nabla Q})^{2(\nabla Q)^3}\right]^{\bar\gamma\ln^2 N} \\
\geq\ & \frac{1}{BN^2}(1-B^{-1}N^{-2})^{(BN^2-1)(5/\ln^4 NB)}(1-B^{-2})^{(B^2-1)(2\bar\gamma\ln^2 N/B)}\left(1-\frac{18}{B}\right)^{\bar\gamma\ln^2 N} \\
\geq\ & \frac{1}{BN^2}e^{-5/(\ln^4 NB)}e^{-2\bar\gamma\ln^2 N/B}\left(1-\frac{18}{B}\right)^{\bar\gamma\ln^2 N} \\
=\ & \frac{1}{BN^2}\left\{e^{-5/(\bar\gamma\ln^6 NB)}e^{-2/B}\left(1-\frac{18}{B}\right)\right\}^{\bar\gamma\ln^2 N} \\
\geq\ & \left[\left(BN^2\right)^{-1/(\bar\gamma\ln^2 N)}e^{-5/(\bar\gamma\ln^6 NB)}\left(1-\frac{2}{B}\right)\left(1-\frac{18}{B}\right)\right]^{\bar\gamma\ln^2 N} \\
\geq\ & \left[\left(BN^2\right)^{-\ln N\ln B/(a\ln^2 N)}e^{-5\ln N\ln B/(a\ln^6 NB)}\left(1-\frac{2}{B}\right)\left(1-\frac{18}{B}\right)\right]^{\bar\gamma\ln^2 N} \\
=\ & \left[\left(BN^2\right)^{-1/(a\ln N)}e^{-5/(a\ln^5 NB)}\left\{\left(1-\frac{2}{B}\right)\left(1-\frac{18}{B}\right)\right\}^{1/lnB}\right]^{\bar\gamma\ln^2 N\ln B} \\
=\ & \left[\left(e^{\ln B+2\ln N}\right)^{-1/(a\ln N)}e^{-5/(a\ln^5 NB)}\left\{\left(1-\frac{2}{B}\right)\left(1-\frac{18}{B}\right)\right\}^{1/lnB}\right]^{\bar\gamma\ln^2 N\ln B} \\
=\ & \left[e^{-\frac{\ln B/\ln N+2+5/(\ln^5 NB)}{a}}\left\{\left(1-\frac{2}{B}\right)\left(1-\frac{18}{B}\right)\right\}^{1/\ln B}\right]^{\bar\gamma\ln^2 N\ln B} \\
\geq\ & \left[e^{-\frac{\ln B/\ln N+3}{a}}e^{-\ln 2}\right]^{\bar\gamma\ln^2 N\ln B} \\
=\ & \left[e^{-\frac{\ln B/\ln N+3+a\ln 2}{a}}\right]^{\bar\gamma\ln^2 N\ln B} \\
\geq\ & \left[e^{-\frac{\ln B}{4}+\frac{\ln 2}{\bar\gamma\ln^2 N\ln B}}\right]^{\bar\gamma\ln^2 N\ln B} \\
=\ & 2\left[e^{-\frac{\ln B}{4}}\right]^{\bar\gamma\ln^2 N\ln B} \\
\geq\ & 2\Pr[A_{e,k}]
\end{aligned}$$

6.3 Computation for type E events

We now show that Eq. (4) holds for events of type $B_{Q,k}$ with $\delta = \frac{1}{2}$. Denote $q = \nabla Q$.

From Eq. (5) it follows that

$$\Pr[E_{Q,k}] \le f(\varepsilon)^{\hat{\gamma} q \ln^3 B} \le e^{-\frac{\varepsilon^2}{4}\hat{\gamma} q \ln^3 B} = e^{-\frac{\hat{\gamma} q \ln^2 B}{4}}$$

We compute:

$$\begin{aligned}
& x_Q \prod_{(Q,B)\in E} (1 - x(B)) \\
\ge\ & x_Q (1-x_A)^{\frac{4N^2}{\ln^4 N}} (1-x_F)^{B\hat{\gamma} q} \prod_{\nabla Q'=\ell}^{4\ln^2 N} (1 - x_{\nabla Q'})^{2(\nabla Q')^3 \hat{\gamma} q} \\
=\ & B^{-2} e^{-q} (1-x_A)^{\frac{4N^2}{\ln^4 N}} (1-x_F)^{B\hat{\gamma} q} \left[\prod_{\nabla Q'=\ell}^{4\ln^2 N} (1 - B^{-2} e^{-\nabla Q'})^{2(\nabla Q')^3} \right]^{\hat{\gamma} q} \\
\ge\ & e^{-2\ln B - q} e^{-5/(\ln^4 NB)} \left\{ \left(1 - \frac{2}{B}\right)\left(1 - \frac{18}{B}\right) \right\}^{\hat{\gamma} q} \\
\ge\ & e^{-2\ln B} e^{-q} e^{-5/(\ln^4 NB)} \left\{ \left(1 - \frac{2}{B}\right)\left(1 - \frac{18}{B}\right) \right\}^{\hat{\gamma} q} \\
=\ & \left[\exp\left\{ -\frac{2\ln B + q + 5/(\ln^4 NB)}{\hat{\gamma} q} \right\} \left\{ \left(1 - \frac{2}{B}\right)\left(1 - \frac{18}{B}\right) \right\}^{1/\ln B} \right]^{\hat{\gamma} q \ln B} \\
\ge\ & \left[\exp\left\{ -\frac{2\ln B + q + 5/(\ln^4 NB) + \mathit{const}}{a} \right\} \right]^{\hat{\gamma} q \ln B} .
\end{aligned}$$

Now recall that $B \ge B_0$, where B_0 is a large enough constant, and a is also a large enough constant, restricted only by $a < 2B_0$, and that $q \ge 1$. Thus, for large enough B_0, a and N, we can have

$$\frac{2\ln B + q + 5/(\ln^4 NB) + \mathit{const}}{a} \le \frac{\ln B}{4} - \frac{\ln 2}{\hat{\gamma} q \ln B},$$

which implies

$$\begin{aligned}
& x_Q \prod_{(Q,B)\in E} (1 - x(B)) \\
\ge\ & \left[\exp\left\{ -\frac{\ln B}{4} + \frac{\ln 2}{\hat{\gamma} q \ln B} \right\}^{\hat{\gamma} q \ln B} \right] \\
=\ & 2 e^{-\frac{\hat{\gamma} q \ln^2 B}{4}} \\
\ge\ & 2 \Pr[E_{Q,k}].
\end{aligned}$$

References

[1] Y. Rabani, ”Path coloring on the mesh”, *In Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science.* , October 1996, pages 400-409

[2] R. Moser and G. Tardos, ”A constructive proof of the general Lovasz Local Lemma”, *Journal of the AMS* , to appear.

[3] P. Klein, S. Plotkin, C. Stein, and É. Tardos, ”Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts”, *SIAM J. Comput.*, 23(3):466487, 1994.

[4] N. Alon and J. H. Spencer, ”The probabilistic method”, second edition, *John Wiley & Sons*, Inc. 2000.