Optimal Oblivious Path Selection on the Mesh

Costas Busch Malik Magdon-Ismail Jing Xi
Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA.
{buschc,magdon,xij2}@cs.rpi.edu

Abstract

In the oblivious path selection problem, each packet in the network independently chooses a path, which is an important property if the routing algorithm is to be independent of the traffic distribution. The quality of the paths is determined by the congestion C, the maximum number of paths crossing an edge, and the dilation D, the maximum path length. So far, the oblivious algorithms studied in the literature have focused on minimizing the congestion while ignoring the dilation.

An open question is whether C and D can be controled simultaneously. Here, we answer this question for the d-dimensional mesh. We present an online algorithm for which C and D are both within $O(d^2)$ of optimal. The algorithm uses randomization, and we show that the number of random bits required per packet is within $O(d)$ of the minimum number of random bits required by any algorithm that obtains near-optimal congestion. For fixed d, our algorithm is asymptotically optimal.

1. Introduction

Given a set of packet transfer requests in a communication network, a routing algorithm must select the paths that will be traversed in order to fulfill all the requests. A routing algorithm is *oblivious* if every path that is selected for each request is chosen independently of every other path. Oblivious algorithms are by their nature distributed and capable of solving online routing problems, where packets continuously arrive in the network. Hence, oblivious routing (path selection) is preferred to non-oblivious routing, since one does not need to make assumptions regarding the nature of the traffic.

We study oblivious algorithms in the context of a synchronous routing model in which at most one packet traverses any edge during a time step. Initially, at time zero, a set of packets must simulataneously select their paths. The quality of the paths selected can be measured by two parameters: the *(edge) congestion* C, the maximum number of paths that use any edge in the network, and the *dilation* D, the maximum length of any path. A trivial lower bound for the total time to transfer all the packets along the selected paths is $\Omega(C+D)$, hence $C+D$ is a natural metric by which to measure the quality of the paths output by a routing algorithm.

Traditionally, the bandwidth of the network is usually the major bottleneck, hence optimizing the load distribution, that is, minimizing the congestion, has been the focus of existing research. In particular, Maggs et al. [9] gave the first oblivious path selection algorithm with optimal edge congestion on the Mesh. Since then, generalizations have been given to oblivious path selection algorithms for arbitrary networks that achieve near optimal congestion [3, 4, 7, 11]. However, these algorithms do not control the dilation. For example, a packet path may be *stretched* by an arbitrary amount – a packet that has destination at a neighboring node may traverse the entire network before reaching its destination.

An interesting open problem is to obtain small path stretch in addition to low congestion. Here, we study this problem for the d-dimensional mesh network, for which we show that it is indeed possible to obtain near optimal congestion while maintaining small stretch. For general networks, this is not possible.

Optimal Oblivious Routing. Given a routing problem (collection of sources and destinations), we define C^* to be the optimal congestion attainable by any routing algorithm (oblivious or not). We define C^*_{obl} to be the optimal congestion attainable if the routing algorithm is restricted to being oblivious. For the d-dimensional Mesh with n nodes, Maggs et al. [9] give the lower bound $C^*_{obl} = \Omega(\frac{C^*}{d} \log n)$ in the worst case. We will compare the congestion of our algorithm with the lower bound on C^*_{obl}. The stretch of a path from a source to a destination is the ratio of the path length to the shortest path length. Clearly, the smallest stretch factor is 1, since a packet can follow a shortest path; however, the choice of shortest paths may increase the congestion.

Our Contributions. We show the surprising result that it is possible to obtain small stretch for any routing instance, using an oblivious algorithm, and without sacrificing on the congestion. Specifically, we give an oblivious routing algorithm for the d-dimensional mesh with n nodes, that achieves congestion $O(dC^* \log n)$, and stretch $O(d^2)$. Considering the class of oblivious algorithms, our algorithm is within $O(d^2)$ of optimal for both congestion and dilation, which means that the routing time bound $(C + D)$ is within $O(d^2)$ of optimal. For fixed d, our algorithm is optimal to within constant factors.

Our algorithm is based upon a hierarchical decomposition of the Mesh. Starting at its source node, a packet constructs its path by randomly selecting intermediate points in submeshes of increasing size until the current submesh contains the destination node. Then random intermediate points are selected in submeshes of decreasing size until the destination node is reached. The key new idea that we introduce is the notion of "bridge" submeshes that make it possible to move from a source to a destination more quickly, without increasing the congestion. These bridge submeshes are instrumental in controling the stretch, while maintaining low congestion.

Our algorithm uses randomization, and we show that randomization is essential for obtaining low congestion. In a deterministic algorithm, a packet path is fixed, given its source and destination. It can be shown that for any deterministic algorithm, there exists a routing problem with high congestion. The only alternative is to use randomization: a packet selects a path probabilistically from a choice of κ alternatives, where κ may depend on the source and destination. Such an algorithm requires at least $\log \kappa$ random bits per packet. We give a lower bound on the number of bits required, $\Omega((1 - \frac{1}{d}) \log \frac{D(s,t)}{d})$, where $D(s,t)$ is the shortest path distance between the source and destination. The number of random bits required by our algorithm is within $O(d)$ of this lower bound, and hence is near-optimal.

Related Work. Most related to our work is the original paper by Maggs et al. [9] where they present an oblivious algorithm with congestion $O(dC^* \log n)$. However, the stretch factor in that algorithm is unbounded. To control stretch, we generalize the hierarchical decomposition for the mesh denoted by an *access tree* [9] to a more general *access graph*. In the access tree, only one path exists between a particular source and destination, however, our decomposition offers several paths, in particular much shorter paths. Our algorithm uses randomized dimension by dimension routing, which alone can improve the result in [9] by a factor of d to $O(C^* \log n)$. Following the work in [9], there have been extensions to general networks, [3, 4, 7, 11], where progressively better oblivious algorithms with near optimal congestion are given. Once again, the stretch is unbounded. For general networks, the stretch and congestion cannot independently be controled.

Non-oblivious approaches to optimizing $C + D$ have received considerable attention, and near optimal algorithms are discussed in [1, 2, 12, 13]. As already mentioned, such offline algorithms require knowledge of the traffic distribution *a priori* and generally do not scale well with the number of packets. We show that for the mesh, distributed and oblivious algorithms are within a logarithmic factor from the optimal offline performance, hence there is no significant benefit from using the offline algorithm. Trade offs between stretch and congestion have been studied in wireless networks [6].

Lower bounds on the competitive ratio of oblivious routing has been studied for various types of networks. Maggs et al. [9] give the $\Omega(\frac{C^*}{d} \log n)$ lower bound on the competitive ratio of an oblivious algorithm on the mesh. Valiant and Brebner [14] perform a worst case theoretical analysis on oblivious routing on specific network topologies such as the hypercube. Borodin and Hopcroft [5] and Kaklamanis et al. [8] showed that deterministic oblivious routing algorithms can not approximate the minimal load on most non-trivial networks, which justifies the necessity for randomization. Here, we give a lower bound on the amount of randomization required.

Paper Outline. We begin with some preliminary definitions (Section 2), and continue with our mesh decomposition algorithm in 2-dimensions (Section 3), which we generalize to d-dimensions in Section 4. We discuss randomization requirements in Section 5, and we conclude in Section 6. Several proofs are relegated to the appendix.

2. Preliminaries

The d-dimensional mesh M is a d-dimensional grid of nodes with side length m_i in dimension i. There is a link connecting a node with each of its $2d$ neighbors (except at the nodes at the boundaries of the mesh). We denote by n the size of M, $n = \mathsf{size}(M) = \prod_{i=1}^{d} m_i$, and by $|E|$ the number of edges in the network. Each node has a coordinate with the top-left node having coordinate $(0,0)$. We refer to specific submeshes by giving its end points in every dimension, for example, $[0,3][2,5]$ refers to a 4×4 submesh, with the x coordinate ranging from 0 to 3 and the y coordinate from 2 to 5.

The input for the path selection problem is a set of N sources and destinations (i.e. packets), $\Pi = \{s_i, t_i\}_{i=1}^{N}$ and the mesh M. The output is a set of paths, $P = \{p_i\}$, where each path $p_i \in P$ is from node s_i to node t_i. The length of path p, denoted $|p|$, is the number of edges it uses. We denote the length of the shortest path from s to t by $\mathsf{dist}(s,t)$. We will denote by D^* the maximum shortest distance, $\max_i \mathsf{dist}(s_i, t_i)$. The *stretch* of a path p_i, denoted $\mathsf{stretch}(p_i)$, is the ratio of the path length to the shortest path length between its source and destination, $\mathsf{stretch}(p_i) = |p_i|/\mathsf{dist}(s_i, t_i)$. The *stretch factor* for the collection of paths P, denoted $\mathsf{stretch}(P)$, is the maximum stretch of any path in P, $\mathsf{stretch}(P) = \max_i \mathsf{stretch}(p_i)$.

For a submesh $M' \subseteq M$, let $\mathsf{out}(M')$ denote the number of edges at the boundary of M', which connect nodes in M' with nodes outside M'. For any routing problem Π, we define the *boundary congestion* as follows. Consider some submesh of the network M'. Let Π' denote the packets (pairs of sources and destinations) in Π which have either their source or destination in M', but not both. All the packets in Π' will cross the boundary of M'. The paths of these packets will cause congestion at least $|\Pi'|/\mathsf{out}(M')$. We define the boundary congestion of M' to be $B(M', \Pi) = |\Pi'|/\mathsf{out}(M')$. For the routing problem Π, the boundary congestion B is the maximum boundary congestion over all its submeshes, i.e. $B = \max_{M' \subseteq M} B(M', \Pi)$. Clearly, $C^* \geq B$.

3. The 2-Dimensional Mesh

Here we show how to select the paths in a 2-dimensional mesh with equal side lengths $m = 2^k,\ k \geq 0$. The path selection algorithm relies on a decomposition of the mesh to submeshes, and then constructing an access graph, as we describe next.

3.1. Decomposition to Submeshes

We decompose the mesh M into two types of submeshes, type-1 and type-2, as follows.

*Type-*1 *Submeshes.* We define the type-1 submeshes recursively. There are $k+1$ levels of type-1 submeshes, $\ell = 0, \ldots, k$. The mesh M itself is the only level 0 submesh. Every submesh at level ℓ can be partitioned into 4 submeshes by dividing each side by 2. Each resulting submesh is a type-1 submesh at level $\ell + 1$. This construction is illustrated in Figure 1. In general, at level ℓ there are $2^{2\ell}$ submeshes each with side $m_\ell = 2^{k-\ell}$. Note that the level k submeshes are the individual nodes of the mesh.

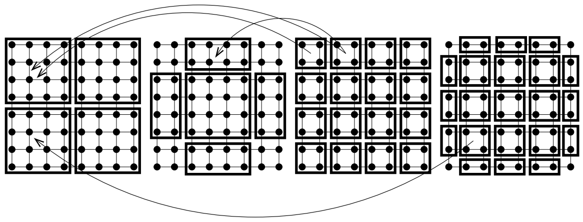


Figure 1. Mesh decomposition for the $2^3 \times 2^3$ mesh. Arrows indicate the parents of a submesh.

*Type-*2 *Submeshes.* There are $k-1$ levels of type-2 submeshes, $\ell = 1, \ldots, k-1$. The type-2 submeshes at level ℓ are obtained by first extending the grid of type-1 meshes by adding one layer of type-1 meshes along every dimension. The resulting grid is then translated by the vector $-(m_\ell/2, m_\ell/2)$. In this enlarged and translated grid, some of the resulting translated submeshes are entirely within M. These are the *internal* type-2 submeshes. For the remaining $external$ type-2 submeshes, we keep only their intersection with M, except that we discard all the "corner" submeshes, because they will be included in the type-1 submeshes at the next level. Notice that all the type-2 submeshes have at least 1 side of length m_ℓ nodes. Figure 1 illustrates the construction.

A submesh of M is *regular* if it is either type-1 or type-2. Unless otherwise stated, we will refer to regular submeshes. The following lemma follows from the construction of the regular submeshes.

Lemma 3.1 *The mesh decomposition satisfies the following properties.*

*(1) The type-*1 *submeshes at a given level are disjoint, as are the type-*2 *submeshes.*

*(2) Every regular submesh at level ℓ can be partitioned into type-*1 *submeshes at level $\ell + 1$.*

(3) Every regular submesh at level $\ell + 1$ is completely contained by a submesh at level ℓ of either type-1 or type-2, or both.

3.2. Access Graph

The access graph $G(M)$, for the mesh M, is a leveled graph with $k+1$ levels of nodes, $\ell = 0, \ldots, k$. The nodes in the access graph correspond to the distinct regular submeshes. Specifically, every level-ℓ submesh (type-1 or type-2) corresponds to a level ℓ node in $G(M)$. Edges exist only between adjacent levels of the graph. Let u_ℓ, $u_{\ell+1}$ be level ℓ and $\ell+1$ nodes of $G(M)$ respectively. The edge $(u_\ell, u_{\ell+1})$ exists if the regular submesh corresponding to u_ℓ completely contains the regular submesh corresponding to $u_{\ell+1}$. We borrow some terminology from trees. We say that u_ℓ is a *parent* of $u_{\ell+1}$ in $G(M)$; the parent relationship is illustrated in Figure 1, for the corresponding submeshes. Note that the access graph is not necessarily a tree, since a node can have two parents (a consequence of Lemma 3.1, part (3)). The depth of a node is the same as its level ℓ, and its height is $k - \ell$. Nodes at height 0 have no children, and are leaves. The leaves in $G(M)$ correspond to single nodes in the mesh. There is a unique root at level 0, which corresponds to the whole mesh M.

Let $p = (u_1, u_2, \ldots, u_k)$ be a path in $G(M)$. We say that p is *monotonic* if every node is of increasing level (i.e., the level of u_i is higher than the level of u_{i+1}), and the respective submeshes of nodes $u_2, \ldots, u_k$ are all of type-1. If p is monotonic, then we say that u_1 is *ancestor* of u_k. We will use a function g to map nodes in the access graph to submeshes. Let u be a node in the access graph with corresponding submesh M'. We define the function g so that $g(u) = M'$. Denote by g^{-1} the inverse of function g, that is, $g^{-1}(M') = u$. Using induction on the height of $G(M)$, and part (2) of Lemma 3.1, we obtain the following lemma:

Lemma 3.2 *Let v be any node (1×1 submesh) of a regular submesh $M' \subseteq M$, then $g^{-1}(M')$ is an ancestor of $g^{-1}(v)$.*

Let u and v be two leaves of $G(M)$, and let A be their (not necessarily unique) deepest common ancestor; Note that A exists and in the worst case is $g^{-1}(M)$ (a consequence of Lemma 3.2). Let $p = (u, \ldots, A, \ldots, v)$, be the concatenation of two monotonic paths, one from A to u and the other from A to v. We will refer to p as the *bitonic* path between u and v. Submesh $g(A)$ may be type-1 or type-2, all the other submeshes in p are of type-1. We will refer to $g(A)$ as a "bridge" submesh, since it provides the connecting point between two monotonic paths. Note that type-2 submeshes can be used as bridges between type-1 submeshes, when constructing bitonic paths between leaves. Further, only one type-2 submesh is ever needed in a bitonic path. These access graph paths will be used by the path selection algorithm. Suppose that $height(A) = h_A$. The length of a bitonic path from u to v is $2h_A$. We now show that h_A cannot be too large. This will be important in proving that the path selection algorithm gives constant stretch.

Lemma 3.3 *The deepest common ancestor of two leaves u and v has height at most $\lceil \log \mathsf{dist}(g(u), g(v)) \rceil + 2$.*

Proof: Let $s, t \in M$ such that $s = g(u)$ and $t = g(v)$. We show that there is a common ancestor with height at most $\lceil \log \mathsf{dist}(s,t) \rceil + 2$. Assume, for simplicity, that we are on the torus (the same result holds for the mesh, with minor technical details in the proof due to edge effects). In this case, all the type-2 meshes are of the same size. We obtain the regular submeshes in the original mesh after truncation of the submeshes at the borders of the torus.

Let $\mu = 2^{\lceil \log \mathsf{dist}(s,t) \rceil} \geq \mathsf{dist}(s,t)$. If $4\mu \geq 2^k$, then the root, $g^{-1}(M)$, is a common ancestor with height $\lceil \log \mathsf{dist}(s,t) \rceil + 2$, so assume that $4\mu < 2^k$. Node s is contained in some type-1 submesh of side length 4μ. Without loss of generality assume that this submesh is $M_1 = [0, 4\mu - 1]^2$. If M_1 also contains t, then we are done, since by Lemma 3.2, $g^{-1}(M_1)$ is a common ancestor at height $\lceil \log \mathsf{dist}(s,t) \rceil + 2$. So suppose that t is contained in some other (adjacent) type-1 submesh M_2. Without loss of generality, there are two possibilities for M_2.

(i) $M_2 = [4\mu, 8\mu - 1][4\mu, 8\mu - 1]$. Since $\mathsf{dist}(s,t) \leq \mu$, $s \in [3\mu, 4\mu - 1]^2$ and $t \in [4\mu, 5\mu - 1]^2$, and so the type-2 submesh $[2\mu, 6\mu - 1]^2$ contains both s and t.

(ii) $M_2 = [0, 4\mu - 1][4\mu, 8\mu - 1]$. There are four cases:

   (a) $s \in [0, 2\mu - 1][3\mu, 4\mu - 1]$ and $t \in [0, 2\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[-2\mu, 2\mu - 1][2\mu, 6\mu - 1]$ contains s, t;

   (b) $s \in [2\mu, 4\mu - 1][3\mu, 4\mu - 1]$ and $t \in [2\mu, 4\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[2\mu, 6\mu - 1]^2$ contains s, t;

   (c) $s \in [\mu, 2\mu - 1][3\mu, 4\mu - 1]$ and $t \in [2\mu, 3\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[\mu, 3\mu - 1][3\mu, 5\mu - 1]$ at height $\lceil \log \mathsf{dist}(s,t) \rceil + 1$ contains s, t;

   (d) $s \in [2\mu, 3\mu - 1][3\mu, 4\mu - 1]$ and $t \in [\mu, 2\mu - 1][4\mu, 5\mu - 1]$, which is similar to (c).

In all cases, s, t are contained in a submesh of height at most $\lceil \log \mathsf{dist}(s,t) \rceil + 2$. ■

3.3. Path Selection

Given the access graph, the procedure to determine a path from a given source s to a destination t is summarized in the following algorithm.

Input: Source s and destination t in the mesh M;
Output: Path $p(s,t)$ from s to t in M;
Let $(u_0, \ldots, u_l)$ denote a bitonic path in $G(M)$ from $g^{-1}(s)$ to $g^{-1}(t)$;
for $i = 0$ to l **do**
  Select a node v_i in $g(u_i)$ uniformly at random; $\{v_0 = s$ and $v_l = t\}$
  if $1 \leq i \leq l$ **then**
    Construct subpath r_i from v_{i-1} to v_i by picking a dimension by dimension shortest path (an at most one-bend path), according to a random ordering of the dimensions;
The path $p(s,t)$ is obtained by concatenating the subpaths r_i, $p(s,t) = r_0 r_1 \cdots r_{l-1}$;

Note that the algorithm is oblivious and local, since each source-destination pair can obtain a path independently of the other paths. We will now show that our algorithm with the generalized access graph, in addition to obtaining optimal congestion, also controls the stretch. First we show the constant stretch property of the selected paths.

Theorem 3.4 *For any two distinct nodes s and t of the mesh,* $\mathsf{stretch}(p(s,t)) \leq 64$.

Proof: Let h be the height of the deepest common ancestor of s and t. Then $p(s,t)$ is the concatenation of paths constructed by the dimension to dimension paths in meshes of sides $2^1, \ldots, 2^{h-1}, 2^h, 2^{h-1}, \ldots, 2^1$. By adding the length of the paths we have that $|p(s,t)| \leq 2(2^1 + \cdots + 2^h + 2^h + \cdots + 2^1 - 2h)$ which implies that $|p(s,t)| \leq 2^{h+3} - 4h$. Since s and t are distinct, $h \geq 1$. By Lemma 3.3, $h \leq \log \mathsf{dist}(s,t) + 3$, and the theorem follows. ∎

We now relate the congestion of the paths selected to the optimal congestion C^*. Let e denote an edge in M. Let $C(e)$ denote the load on e, i.e., the number of times that edge e is used by the paths of all the packets. We will get an upper bound on $E[C(e)]$, and then using a Chernoff bound we will obtain a concentration result.

We start by bounding the probability that some particular subpath formed by the path selection algorithm uses edge e. Consider the formation of a subpath r_i from a submesh M_1 to a submesh M_2, such that M_2 completely contains M_1, and e is a member of M_2. According to the path selection algorithm, mesh M_1 is of type-1, thus all of its sides are equal to m_ℓ, where ℓ is the level of M_1. We show the following lemma.

Lemma 3.5 *Subpath r_i uses edge e with probability at most $2/m_\ell$.*

Proof: For subpath r_i, let v_1 be the starting node in M_1 and v_2 the ending node in M_2 for subpath r_i. Suppose $e = (v_3, v_4)$. Without loss of generality, suppose e is vertical. Since the subpath is a one-bend path, edge e is used either when v_1 or v_2 have the same x coordinate with e. This event occurs with probability at most $2/m_\ell$. ∎

Let P' be the set of paths that go from M_1 to M_2 or vice-versa. Let $C'(e)$ denote the congestion that the packets P' cause on e. We show:

Lemma 3.6 $E[C'(e)] \leq 2|P'|/m_\ell$.

Proof: We can write $P' = P_1 \cup P_2$, where P_1 is the set of subpaths from M_1 to M_2, and P_2 is the subpaths from M_2 to M_1. Then, from Lemma 3.5, the expected congestion at edge e due to the subpaths in P_1 is bounded by $2|P_1|/m_\ell$. With a similar analysis, we have that the expected congestion at e is bounded by $2|P_2|/m_\ell$. Since P_1 and P_2 are disjoint, we obtain $E[C'(e)] \leq 2(|P_1| + |P_2|)/m_\ell = 2|P'|/m_\ell$. ∎

From the definition of the boundary congestion, we have that $B \geq B(M_1, \Pi) \geq |P'|/\mathsf{out}(M_1)$. Therefore, $C^* \geq |P'|/\mathsf{out}(M_1)$. Since each side of M_1 has m_ℓ nodes, we have that $\mathsf{out}(M_1) \leq 4m_\ell$. From Lemmas 3.6 we obtain:

Lemma 3.7 $E[C'(e)] \leq 8C^*$.

We "charge" this congestion to submesh M_2. By Lemma 3.3, only submeshes up to height $h < \log D^* + 3$ can contribute to the congestion on edge e (submeshes of type-1). By summing the congestions due to these at most $2(\log D^* + 3)$ submeshes (a type-1 and a type-2 submesh at each level), and by using Lemma 3.7, we arrive at an upper bound for the expected congestion at edge e:

Lemma 3.8 $E[C(e)] \leq 16C^*(\log D^* + 3)$.

Note that without increasing the expected congestion, we can always remove any cycles in a path, so without loss of generality, we will assume that the paths obtained are acyclic. We now obtain a concentration result on the congestion C obtained by our algorithm, using the fact that every packet selects its path independently of every other packet.

Theorem 3.9 $C = O(C^* \log n)$ *with high probability.*

Proof: Let $X_i = 1$ if path p_i uses edge e, and 0 otherwise. Then $E[C(e)] = E[\sum_i X_i] \le 16C^*(\log D^* + 3)$. Let $|E|$ be the number of edges in the mesh. Asymptotically in $|E|$, $E[C(e)] \le 16C^* \log(|E|D^*)$. Let $\kappa > 2e$, then applying a Chernoff bound [10], and using the fact that $C^* \ge 1$ we find that $P[C(e) > 16\kappa C^* \log(|E|D^*)] < (|E|D^*)^{-16\kappa}$. Taking a union bound over all the edges, we obtain

$$P[\max_{e \in E} C(e) > 16\kappa C^* \log(|E|D^*)] < \frac{1}{(|E|D^*)^{16\kappa - 1}}.$$

Using the fact that $D^* = O(|E|)$, $|E| = O(n^2)$, and choosing $\kappa = 2e + 1$, we get $C = O(C^* \log n)$ with high probability. ■

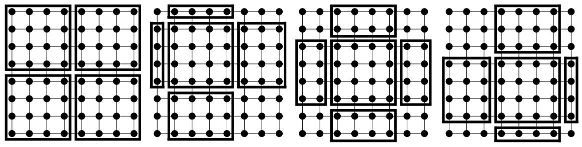
4. The d-Dimensional Mesh

The 2-dimensional decomposition can be directly generalized to a d-dimensional mesh with equal side lengths (2^k, $k \ge 0$) at each dimension. However, the stretch becomes $O(2^d)$, which is excessively high for large d. In order to alleviate the problem, we present an alternative decomposition for which the path selection algorithm has congestion $O(d^2 C^* \log n)$, and stretch $O(d^2)$. For fixed d, these are constant factors from optimal.

4.1. Decomposition

As in 2 dimensions, we have the type-1 meshes, and the translated meshes. However, we now introduce $\Theta(d)$ types of translated meshes at each level. To be specific, consider the level ℓ type-1 mesh with side length m_ℓ. Set $\lambda = \max\{1, m_\ell / 2^{\lceil \log(d+1) \rceil}\}$. We shift the type-1 submeshes by $(j-1)\lambda$ nodes in each dimension to get the type-j submeshes, for $j > 1$. Notice that the number of different types of submeshes at any level is at most $2(d+1)$. If $m_l \ge d+1$, then there are at least $(d+1)$ different types of submeshes. Thus, an edge is contained in $O(d)$ submeshes at every level. Figure 2 shows an example for $d = 3$, $m_l = 4$, and $\lambda = \frac{m_l}{4} = 1$. The access graph can be constructed using these submeshes. The path selection is similar to the 2-dimensional case, where the bridge is some type-j submesh, while all the other submeshes in the bitonic path are of type-1.

In particular, suppose we are given two nodes s and t in the mesh, we show that in the access graph there is some regular submesh (of some type-j) that completely contains s and t and has side length $O(d \cdot \mathsf{dist}(s,t))$. As we did for $d = 2$, assume, for simplicity, that we are on the torus. Suppose that $s = (s_1, \ldots, s_d)$ and $t = (t_1, \ldots, t_d)$, with $|t_i - s_i| \le \mathsf{dist}(s,t)$. Let $R = [a_1, b_1] \cdots [a_d, b_d]$ be a region of the mesh defined by s and t, such that $[a_i, b_i] = [\min(s_i, t_i), \max(s_i, t_i)]$. Note that R contains s and t. We will find a regular submesh which completely contains R. Consider now the deepest level that has submeshes of side at least $2(d+1) \cdot \mathsf{dist}(s,t)$. Let h be the height of this level. The side length of submeshes at height h is $m_h = 2^h$, where $4(d+1) \cdot \mathsf{dist}(s,t) \ge m_h \ge 2(d+1) \cdot \mathsf{dist}(s,t)$, and let the shift amount at this height be $\lambda_h = m_h / 2^{\lceil \log(d+1) \rceil} \ge \mathsf{dist}(s,t)$. We have:



Type 1. Type 2. Type 3. Type 4.

Figure 2. Mesh decomposition for the $3-$dimensional mesh. Only 2 of the 3 dimensions are depicted

Lemma 4.1 *For every height $\ge h$, R is completely contained by some regular submesh.*

Proof: First consider height h. The *anchor* node of a submesh is the node with the smallest coordinate in each dimension. Consider dimension i. Let $\Delta_i = [a_i, b_i]$, be the side of R in dimension i. By construction of the different types of submeshes, we have that Δ_i contains the ith dimension of anchors of at most one type of submesh at height h. This is because in each dimension the distance between anchors of different mesh types is at least Δ_i. Considering now all the dimensions, since we have d dimensions and at least $d+1$ different types of submeshes, by the pigeonhole principle, there exists at least one type of submesh, say type-ζ, such that R does not contain any anchors of any type-ζ submesh in any dimension. This implies that R is completely contained by a type-ζ submesh M' at height h. The same argument also holds for heights $> h$. ■

From the proof of Lemma 4.1, it follows that there is some regular submesh M_2 at level $h+1$ that contains R. On the access graph, the bitonic path is constructed so that it goes from s to t through the bridge submesh at height $h+1$. Expect possibly for the bridge, the other submeshes on the path are of type-1. By construction, it is guaranteed that M_2 is decomposed into type-1 submeshes at height $h' = \lfloor \log \mathsf{dist}(s,t) \rfloor$. Let M_1 be the type-1 submesh at height h' that contains s, and M_3 the submesh that contains t. The path is formed by first using submeshes of type-1 from s up to submesh M_1, then to bridge M_2, down to M_3, and then down to t by using type-1 submeshes.[1]

4.2. Stretch and Congestion

We now compute the stretch of the path selection algorithm in the d dimensions.

Theorem 4.2 (Stretch for d Dimensions) *For any two distinct nodes s and t of the mesh,* $\mathsf{stretch}(p(s,t)) = O(d^2)$.

Proof: Let p denote the path from s to t. Let r_1, r_2, and r_3, denote the respective subpaths from s to M_1, from M_1 to M_2 to M_3, and from M_3 to t. First we compute $|r_1|$. Subpath r_1 consists of h' subpaths where the subpath from height $i-1$ to height i has length at most $d(2^i-1)$. Therefore, $|r_1| \leq 2d\sum_{i=0}^{h'}(2^i-1) = 2d(2\cdot 2^{h'}-1-h')$. Since $2^{h'} \leq \mathsf{dist}(s,t)$, we have $|r_1| \leq 2d(2\cdot\mathsf{dist}(s,t)-1-h) = O(d\cdot\mathsf{dist}(s,t))$. The path length of r_2 is no more than $2d(2^{h+1}-1)$. Since $2^{h+1} \leq 8(d+1)\cdot\mathsf{dist}(s,t)$, we have that $|r_2| \leq 2d(8(d+1)\cdot\mathsf{dist}(s,t)+1) = O(d^2\cdot\mathsf{dist}(s,t))$. Further, $|r_3| = |r_1|$. Thus, $|p| = |r_1|+|r_2|+|r_3| = O(d^2\cdot\mathsf{dist}(s,t))$. ■

With an analysis similar to the 2-dimensional case, we can show that the expected congestion caused on an edge e from subpaths between a mesh of a lower height and a mesh of a higher level, is $E[C'(e)] \leq 4C^*$) (the analysis can be found in the appendix). We charge this congestion to the higher height mesh. Since paths use heights up to $O(\log(dD^*))$, and each height has $O(d)$ different types of submeshes, at most $O(d\log(dD^*))$ different submeshes can contribute to the congestion on e. Thus $E[C(e)] = O(dC^*\log(dD^*))$. As with the 2-dimensional analysis, we get a concentration result by applying a Chernoff bounding argument and the facts that $dD^* = O(|E|)$, $|E| = O(dn)$, and $d = O(n)$:

Theorem 4.3 (Congestion for d Dimensions) $C = O(dC^*\log n)$ *with high probability.*

5. Random Choices

Here we show that randomization is unavoidable for oblivious algorithms, if they are to obtain near optimal congestion: such algorithms need access to a substantial number of random bits, and we show that our algorithm uses a near minimal number of random bits.

Consider the d-dimensional mesh, with equal side lengths (2^k, $k \geq 0$) in each dimension. We say that a path selection algorithm A is a *κ-choice* algorithm, if for every source-destination pair (s,t), A chooses the resulting path from κ possible different paths from s to t. The path choice is randomized according to some probability distribution which may be specific to each source/destination pair. For the case in which $\kappa = 1$, the algorithm is deterministic. A κ-choice algorithm requires $\log\kappa$ bits of randomization per packet to select any particular path.

5.1. Path Choices and Congestion

Consider a κ-choice algorithm A. We now construct a routing problem Π_A which gives a lower bound on κ in terms of the congestion. The routing problem construction uses algorithm A itself. Suppose that $l = 2^s$, for some $s \geq 0$. Each node in the network is the source of one packet and the destination of one packet (permutation). The distance of every packet to its destination is l. Such a problem can be constructed by dividing the network into submeshes of side length l, and then forming pairs of submeshes which exchange their packets at the respective nodes. However, the problem construction has not finished yet. Every packet uses the path from its source to its destination so that it is the path with maximum probability among the κ possible paths provided by algorithm A for the particular source and destination. This way, we obtain paths for all the packets in the network. The average number of paths crossing an edge is at least $\frac{m^d l}{dm^d} = \frac{l}{d}$. Therefore, there is an edge e which is crossed by at least l/d packets. Let Π_A denote a set consisting of l/d packets crossing e. This concludes the construction of Π_A. Note that Π_A keeps only a subset of the packets in the original permutation problem. We have the following result that relates path choices and congestion.

Lemma 5.1 *For any κ-choice algorithm A and corresponding routing problem Π_A with $D^* = l$, the expected congestion Y is at least $\frac{l}{d\kappa}$.*

Proof: By construction of Π_A, there is an edge e which is used by each packet in Π_A with probability at least $1/\kappa$. The expected congestion Y_e on edge e is bounded by $Y_e \geq \frac{|\Pi_A|}{\kappa} = \frac{l}{d\kappa}$. The expected congestion $Y = E[\max_{e_i} C(e_i)] \geq E[C(e)] = Y_e$, since $\max_{e_i} C(e_i) \geq C(e)$. ■

1 The reason of using the height $h+1$ for bridges instead of height h, is due to technical reasons explained in the appendix.

5.2. Lower Bound on Randomization

Here, we establish a lower bound on the number of bits per packet required by any algorithm that achieves congestion comparable to our algorithm. Let algorithm H denote our d-dimensional algorithm presented in Section 4. Consider an arbitrary κ-choice algorithm A and its corresponding routing problem Π_A as described in the previous section. We will give an upper bound on the performance of H for routing problem Π_A (notice that H and Π_A are independent unless $H = A$). Let C_H denote the expected congestion of H for Π_A. We will give an upper on C_H in terms of l and d.

Lemma 5.2 $C_H = O\left(\left(\frac{l}{d}\right)^{\frac{1}{d}} \cdot \log n\right)$.

Proof: From the analysis in Sections 3 and 4, $C_H = O(dB \log n) = O(dC^* \log n)$, where B is the boundary congestion. We give an upper bound on B in terms of l and d.

For any arbitrary submesh M', we give an upper bound on $B(M', \Pi_A)$, which is also an upper bound for B. Assume M' is a $m_1 \times \cdots \times m_d$ with n' nodes. It can be shown that $\mathsf{out}(M') \geq d \cdot n'^{\frac{d-1}{d}}$ (see a proof in the appendix). Let Π' denote the packets that have to cross the border of M' (have source outside and destination inside M' or vice versa). Clearly, $|\Pi'| \leq \min\{\frac{l}{d}, n'\}$, since $|\Pi'| \leq |\Pi_A| = \frac{l}{d}$, and each node is the source/destination of at most one packet. By definition of the boundary congestion, $B(M', \Pi_A) = \frac{|\Pi'|}{\mathsf{out}(M')}$. There are two cases:

(1) $\frac{l}{d} \leq n'$: $\frac{|\Pi'|}{\mathsf{out}(M')} \leq \frac{l/d}{d \cdot n'^{\frac{d-1}{d}}} \leq \frac{l}{d^2}\left(\frac{d}{l}\right)^{\frac{d-1}{d}} = \frac{l^{\frac{1}{d}}}{d^{(1+\frac{1}{d})}}$;

(2) $\frac{l}{d} > n'$: $\frac{|\Pi'|}{\mathsf{out}(M')} \leq \frac{n'}{d \cdot n'^{\frac{d-1}{d}}} = \frac{n'^{1/d}}{d} < \frac{\left(\frac{l}{d}\right)^{\frac{1}{d}}}{d} = \frac{l^{\frac{1}{d}}}{d^{(1+\frac{1}{d})}}$.

In both cases, $B(M', \Pi_A) \leq \frac{l^{\frac{1}{d}}}{d^{(1+\frac{1}{d})}}$. Since M' was arbitrary, $B \leq \frac{l^{\frac{1}{d}}}{d^{(1+\frac{1}{d})}}$, consequently $C_H = O\left(\left(\frac{l}{d}\right)^{\frac{1}{d}} \cdot \log n\right)$. ∎

Suppose that algorithm A, is at least as good as H for arbitrary routing problems with respect to congestion. Let C_A denote the expected congestion of algorithm A on a routing problem. Then $C_A \leq C_H$ for routing problem Π_A. From Lemmas 5.2, and 5.1, algorithm A requires $\kappa = \Omega\left(\left(\frac{l}{d}\right)^{1-\frac{1}{d}} \frac{1}{\log n}\right)$ path choices, or equivalently $\Omega\left((1 - \frac{1}{d}) \log \frac{l}{d} - \log\log n\right)$ random bits per packet. Since $D^* = l$ we have the following result.

Lemma 5.3 *There is a routing problem with $D^* = \Omega(\log n)$ for which any algorithm A with $C_A = O(C_H)$ requires $\Omega\left((1 - \frac{1}{d}) \log \frac{D^*}{d}\right)$ random bits per packet.*

i.e., there exist routing problems for which significant randomization is unavoidable for any algorithm with congestion at least as good as algorithm H.

5.3. Upper Bound on Randomization

Here we compute an upper bound on the number of random bits per packet required by our d-dimensional algorithm H. Consider a path formation from submesh M_1 to a submesh M_2 (we will refer to this as a "step" in the algorithm). The algorithm makes the following random selections:

i. A random ordering of the dimensions from the $d!$ possible orderings, requiring $O(d \log d)$ random bits each time.

ii. A random node in M_2. The largest submesh in the bitonic path has size $O((dD^*)^d)$ (Lemma 4.1), so the number of random bits required each time is $O(d \log(dD^*))$.

Consequently, in a single step of the algorithm, the number of bits required per step is $O(d \log(dD^*))$. Since, the number of steps to select a path is at most $2 \log D^* + 1$, the total number of random bits needed per packet is $O(d \log^2(dD^*))$. We can decrease the number of random bits needed by a factor of $\log(dD^*)$, as follows:

i. We select the random order of dimensions only once at the beginning of the path creation and we use the same order at the subsequent steps of the algorithm for the same path.

ii. We compute two random nodes, namely v_1 and v_2, at the largest submesh in the bitonic path. For the path formation we only use bits from v_1 and v_2 to compute the (random) nodes in the submeshes of the bitonic path. We use as many bits as necessary from v_1 or v_2, depending on the size of the submesh considered each time. We alternate the use of nodes v_1 and v_2 in the path formation, so that if the path is formed on submeshes $M_1, \ldots, M_k$, we use bits from v_1 for $M_1, M_3, \ldots$, (odd indeces) and from v_2 for $M_2, M_4, \ldots$ (even indeces).

Therefore, only $O(O(d \log d))$ random bits are needed for the random order of dimensions, and $O(d \log(dD^*))$

random bits for the randomization of v_1 and v_2. This gives us in total $O(d\log(dD^*))$ bits.

Lemma 5.4 *For any routing problem, algorithm H requires $O(d\log(dD^*))$ random bits.*

From Lemma 5.3 we have that there exist routing problems with $D^* = \Omega(\log n)$ such that any oblivious routing algorithm which is at least as good as H in expected congestion, requires $\Omega((1-\frac{1}{d})\log\frac{D^*}{d})$ random bits. The same result holds for H too. Therefore, when $D^* = \Omega(d+\log n)$, algorithm H uses a number of bits which is only a factor $O(d)$ from optimal. We have the following theorem:

Theorem 5.5 (Approximation of Random Bits) *The number of random bits used by Algorithm H is within $O(d)$ of optimal.*

6. Discussion

We have shown that for the mesh, one can *simultaneously* control both the stretch and the congestion. As already mentioned, this cannot be done for general networks. Interesting open issues that remain are to develop similar algorithms for other specific networks, and to develop oblivious algorithms that minimize $C+D$ for general networks.

References

[1] J. Aspens, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. Online load balancing with applications to machine scheduling and virtual circuit routing. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 623–631, 1993.

[2] B. Awerbuch and Y. Azar. Local optimization of global objectives: competitive distributed deadlock resolution and resource allocation. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 240–249, Santa Fe, New Mexico, 1994.

[3] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 383–388, San Diego, CA, June 2003. ACM Press.

[4] Marcin Bienkowski, Miroslaw Korzeniowski, and Harald Räcke. A practical algrorithm for constructing oblivious routing schemes. In *Proceedings of the 15th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 24–33, Jun. 2003.

[5] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Science*, 30:130–145, 1985.

[6] Jie Gao and Li Zhang. Tradeoff between stretch factor and load balancing ratio in wireless network routing. In *Proceedings of th Symposium on Principles of Distributed Computing (PODC)*, page to appear, 2004.

[7] Chris Harrelson, Kristen Hildrum, and Satish Rao. A polynomial-time tree decomposition to minize congestion. In *Proceedings of the 15th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 34–43, Jun. 2003.

[8] Christos Kaklamanis, Danny Krizanc, and Thanasis Tsantilas. Tight bounds for oblivious routing in the hypercube. In *Proceedings of 2nd IEEE Symposium on Parallel and Distributed Processing (2nd SPAA 90)*, pages 31–36, Crete, Greece, July 1990.

[9] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westerman. Exploiting locality in data management in systems of limited bandwidth. In *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science*, pages 284–293, 1997.

[10] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 2000.

[11] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Annual Symposium on the Foundations of Computer Science*, pages 43–52, Nov. 2002.

[12] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[13] A. Srinivasan and C-P. Teo. A constant factor approximation algorithm for packet routing, and balancing local vs. global criteria. In *Proceedings of the ACM Symposium on the Theory of Computing (STOC)*, pages 636–643, 1997.

[14] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 263–277, May 1981.

A. Appendix

A.1. The d-Dimensional Mesh

Here, we consider the d-dimensional path selection algorithm of Section 4. We compute an upper bound on the expected congestion on an edge.

Let M_1 and M_2 be two submeshes with respective side lengths $a_1, \ldots, a_d$ and $b_1, \ldots, b_d$, such that: (i) M_1 is of type-1, (ii) M_2 completely contains M_1, (iii) each side of M_2 is at least twice the side of M_1, $b_i \geq 2a_i$, for all $1 \leq i \leq d$. The d-dimensional algorithm preserves conditions (i)-(iii), for every pair of consecutive submeshes of the bitonic path (condition (iii) is preserved even when M_2 is a bridge, since the height of the bridge is $h + 1$).

Consider the formation of a subpath r from a submesh M_1 to M_2, where r is formed by following a shortest path from a randomly chosen node v_1 in M_1 to a randomly chosen node v_2 in M_2. The path is formed by following a dimension by dimension path, with a random ordering of dimensions. Since M_1 is of type-1, it holds that all of its sides are equal, $a_1 = a_2 = \cdots = a_d = a$. Let e be an edge of M_2. We show the following result:[2]

Lemma A.1 *Subpath r uses edge e with probability at most* $\dfrac{2}{da^{d-1}}$.

Proof: Without loss of generality, suppose that e is an edge in M_2 along dimension l, from $(x_1, \ldots, x_l, \ldots, x_d)$ to $(x_1, \ldots, x_l + 1, \ldots, x_d)$. Suppose $v_1 = (y_1, \ldots, y_d)$ and $v_2 = (z_1, \ldots, z_d)$. Let ρ be a random permutation which determines the order of dimensions. Suppose that dimension l occurs in position i, i.e., $\rho(i) = l$. Then the probability $P[e]$ that edge e is used by r is bounded from above by the probability that, $z_{\rho(1)} = x_{\rho(1)}, \ldots, z_{\rho(i-1)} = x_{\rho(i-1)}$, and, $y_{\rho(i+1)} = x_{\rho(i+1)}, \ldots, y_{\rho(d)} = x_{\rho(d)}$. We obtain: $P[e] = \frac{1}{b_{\rho(1)} \cdots b_{\rho(i-1)}} \cdot \frac{1}{a_{\rho(i+1)} \cdots a_{\rho(d)}} \leq \frac{1}{2a_{\rho(1)} \cdots 2a_{\rho(i-1)}} \cdot \frac{1}{a_{\rho(i+1)} \cdots a_{\rho(d)}} = \frac{1}{2^{i-1} a^{d-1}}$.

Since ρ is a random permutation, the probability that $\rho(i) = l$ for any i is $1/d$, so multiplying by $1/d$, summing, and using the fact that $\sum_{i=1}^{d} \frac{1}{2^{i-1}} \leq 2$, we obtain:

$$P[e] \leq \frac{1}{d} \sum_{i=1}^{d} \frac{1}{2^{i-1} a^{d-1}} = \frac{1}{da^{d-1}} \sum_{i=1}^{d} \frac{1}{2^{i-1}} \leq \frac{2}{da^{d-1}}.$$

■

Let P' be the set of paths that go from M_1 to M_2 or vice-versa. Let $C'(e)$ denote the congestion that the packets P' cause on e. We show:

Lemma A.2 $E[C'(e)] \leq \dfrac{2|P'|}{da^{d-1}}$.

Proof: We can write $P' = P_1 \cup P_2$, where P_1 is the set of subpaths from M_1 to M_2, and P_2 is the subpaths from M_2 to M_1. Then, from Lemma A.1, the expected congestion at edge e due to the subpaths in P_1 is bounded by $2|P_1|/(da^{d-1})$. With a similar analysis, we have that the expected congestion at e is bounded by $2|P_2|/(da^{d-1})$. Since P_1 and P_2 are disjoint, we obtain $E[C'(e)] \leq 2(|P_1| + |P_2|)/(da^{d-1}) = 2|P'|/(da^{d-1})$. ■

We have that $C^* \geq B(M_1, \Pi) \geq |P'|/\mathsf{out}(M_1)$. Since each side of M_1 has a nodes, we have that $\mathsf{out}(M_1) \leq 2da^{d-1}$. From Lemma A.2, we obtain:

Lemma A.3 $E[C'(e)] \leq 4C^*$.

A.2. Lower Bound on Random Choices

Here we show that any submesh with a particular number of nodes must have a lower bound on its outgoing edges.

Lemma A.4 *For any submesh M' with n' nodes,* $\mathsf{out}(M') \geq dn'^{\frac{d-1}{d}}$.

Proof: Let m_i $1 \leq i \leq d$ be the sides of M'. Let z_i denote the product $m_1 \cdots m_{i-1} m_{i+1} \cdots m_d$. Each z_i represents a surface of M'. At most d different surfaces could be at the border of M and thus may not be used at $\mathsf{out}(M')$. Therefore, $\mathsf{out}(M') \geq \sum_i z_i$. We want to find which m_i minimize $\sum_i z_i$, given that $\prod_i m_i = n'$. This will give us a lower bound on $\mathsf{out}(M')$.

Consider the Lagrangian $L = \sum_i z_i - \lambda(\prod_i m_i - n')$. Then, $\sum_i z_i$ is minimized when $\frac{\partial L}{\partial m_i} = 0$, for all $1 \leq i \leq d$. We have,

$\frac{\partial L}{\partial m_1} = ((m_3 \cdots m_d) + (m_2 m_4 \cdots m_d) + \cdots + (m_2 \cdots m_{d-1})) - \lambda \prod_{i=2}^{d} m_i = 0$, and

$\frac{\partial L}{\partial m_2} = ((m_3 \cdots m_d) + (m_1 m_4 \cdots m_d) + \cdots + (m_1 m_3 \cdots m_{d-1})) - \lambda m_1 \prod_{i=3}^{d} m_i = 0$,

which gives,

$$\begin{aligned} \lambda &= \frac{1}{m_2} + \frac{1}{m_3} + \cdots + \frac{1}{m_d}, \text{ and} \\ \lambda &= \frac{1}{m_1} + \frac{1}{m_3} + \cdots + \frac{1}{m_d}. \end{aligned}$$

Therefore, $m_1 = m_2$. By computing the rest of the partial derivatives, we can show $m_{i-1} = m_i$ for $2 \leq i \leq d$. Therefore, $\sum_i z_i$ is minimized when $m_i = m_j$, for all $1 \leq i, j \leq d$. Consequently, $\sum_i z_i \geq d{m_i}^{(d-1)} = d(n'^{\frac{1}{d}})^{(d-1)} = dn'^{\frac{d-1}{d}}$. Which implies that $\mathsf{out}(M') \geq dn'^{\frac{d-1}{d}}$. ■

2 This result cannot be used for the 2-dimensional analysis of Section 3.3, since the condiction (iii) listed above does not hold for that algorithm.