

Communication-Aware Processor Allocation for Supercomputers

Michael A. Bender* David P. Bunde† Erik D. Demaine‡ Sándor P. Fekete§
 Vitus J. Leung¶ Henk Meijer|| Cynthia A. Phillips¶

Abstract

This paper gives processor-allocation algorithms for minimizing the average number of communication hops between the assigned processors for grid architectures, in the presence of *occupied* cells. The simpler problem of assigning processors on a *free* grid has been studied by Karp, McKellar, and Wong who show that the solutions have nontrivial structure; they left open the complexity of the problem.

The associated clustering problem is as follows: Given n points in \mathbb{R}^d , find k points that minimize their average pairwise L_1 distance. We present a natural approximation algorithm and show that it is a $\frac{7}{4}$ -approximation for 2D grids. For d -dimensional space, the approximation guarantee is $2 - \frac{1}{2d}$, which is tight. We also give a polynomial-time approximation scheme (PTAS) for constant dimension d , and report on experimental results.

Keywords: Processor allocation, supercomputers, communication cost, Manhattan distance, clustering, approximation, polynomial-time approximation scheme (PTAS).

1 Introduction

This paper gives processor-allocation algorithms for minimizing the average number of communication hops between the processors assigned to a job on a grid architecture. Our problem is: given a set P of n points in \mathbb{R}^d , find a subset S of k points with minimum average pairwise L_1 distance.

Processor Allocation in Supercomputers. As part of the Advanced Simulation and Computing Initiative¹ [18], the Department of Energy Laboratories are purchasing increasingly powerful custom supercomputers. In a parallel effort to increase the scalability of commodity-based supercomputers, Sandia National Laboratories is developing the Computational Plant or Cplant [6, 28]. Sandia’s diverse computing resources rely on scheduling/queueing software such as NQS [10] or PBS [26] to determine which job to run next. This decision is based on policy enforcement (fairness and priority) rather than optimal use of resources. When a job is selected to run, the allocator assigns it to a set of processors, which are exclusively dedicated to this job until it terminates. Security constraints forbid migration, preemption, or multitasking.

*Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, USA. Email: bender@cs.sunysb.edu. Partially supported by Sandia Natl. Labs. and NSF Grants EIA-0112849 and CCR-0208670.

†Department of Computer Science, University of Illinois, Urbana, IL 61801, USA. Email: bunde@uiuc.edu. Supported in part by Sandia National Laboratories.

‡MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139, USA. Email: edemaine@mit.edu. Supported in part by NSF Grant EIA-0112849.

§Dept. of Mathematical Optimization, Braunschweig University of Technology, 38106 Braunschweig, Germany. Email: s.fekete@tu-bs.de. Partially supported by DFG grant FE 407/10.

¶Discrete Algorithms & Math Department, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185-1110, USA. {vjleung, caphill}@sandia.gov.

||Dept. of Computing and Information Science, Queen’s University, Kingston, Ontario, K7L 3N6, Canada. Email: henk@cs.queensu.ca.

¹This program was originally called the Accelerated Strategic Computing Initiative.

To obtain maximum throughput in a network-limited computing system, the processors that are allocated to a single job should be physically near each other. This placement reduces communication costs and avoids bandwidth contention caused by overlapping jobs. Processor locality is particularly important in commodity-based supercomputers, which typically have higher communication latencies and lower bandwidth than supercomputers with custom networks. For example, in Cplant supercomputers, where the switches roughly form two- or three-dimensional meshes with some toroidal wraps, a good processor allocation occupies an approximate subcube of switches.

Experiments have shown that processor allocation affects throughput on a range of architectures [3, 19, 22, 23, 25]. On Cplant in particular, when two high-communication jobs are hand-placed on the machine so that their communication paths overlap significantly, both jobs’ running times approximately double [19].

Several papers suggest that minimizing the *average number of communication hops* is an appropriate metric for job placement [17, 22, 23]. Experiments with a communication test suite demonstrate that this metric correlates with the job’s completion time. See Figure 1, reproduced from [19]. While this correlation may not be strong for all communication patterns, it does seem to hold for the most intense: all-to-all [7].

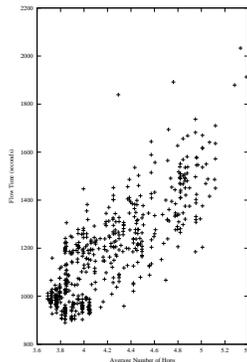


Figure 1: Job flow time (completion time minus arrival time) as a function of the average number of communication hops between processors assigned to that. All plotted jobs used 30 processors on a 128-processor machine configured as a 2D mesh with toroidal wraps.

Early processor-allocation algorithms allocate only a convex set of processors to a job [5, 9, 20, 33]. This design decision means that each job’s communication can be routed entirely within the processors assigned to that job, so jobs contend only with themselves. Unfortunately, this design decision also reduces the achievable system utilization to levels unacceptable for any government-audited system [16, 30].

More recent work [8, 19, 21, 24, 30] allows discontinuous allocations but tries to cluster processors and minimize contention with previously allocated jobs. Mache, Lo, and Windisch [24] propose the MC algorithm for grid architectures. MC assumes that jobs request processors in a particular rectangular shape. Each free processor evaluates the quality of an allocation centered on itself. It counts the number of free processors within a submesh of the requested size centered on itself and within “shells” of processors surrounding this submesh. The processors are weighted by the shell containing them; 0 for the initial submesh, 1 for the first shell out, 2 for the second, and so on. The sum of the weights gives the cost of the allocation. MC chooses the allocation with lowest cost; see Figure 2 reproduced from [24]. Since users of Cplant do not request processors in a particular shape, MC cannot be used. Thus, in this paper, we consider a variant called MC1x1, in which shell 0 is 1×1 and subsequent shells grow in the same way as in MC.

Until recently, processor allocation on the Cplant system was *not* based on the locations of the free processors. The allocator simply verified that a sufficient number of processors were free before dispatching a job. The current allocator uses space-filling curves and 1D bin-packing techniques based upon the results of [19]. The mature version of our research will determine the allocation algorithm for both the next release of Cplant system software [28] and the initial release of Red Storm system software [29].

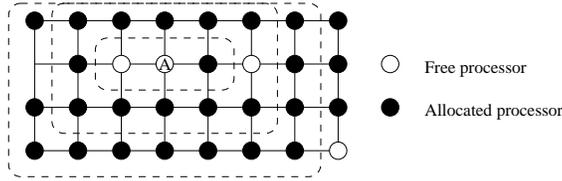


Figure 2: Illustration of MC: Shells around processor A for a 3×1 request.

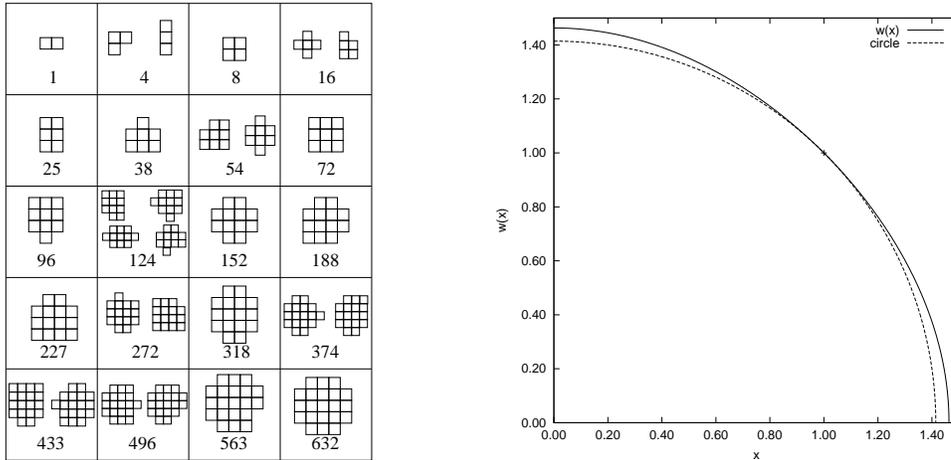


Figure 3: (Left) Optimal unconstrained clusters for small values of k ; numbers shown are the sums of Manhattan distances. (Right) Plot of a quarter of the optimal limiting boundary curve; the dotted line is a circle.

Related Algorithmic Work. A natural special case of the allocation problem is the *unconstrained* problem, in the absence of occupied processors: for any number k , find a set of k grid points of minimal average Manhattan distance. For moderate sizes of k , these allocations can be computed by exhaustive search; see Figure 3. The resulting shapes appear to approximate some “ideal” rounded shape, with better and better approximation for growing k . Karp et al. [15] and Bender et al. [4] study the exact nature of this shape. They show that for large k , the optimal solution does *not* approach any basic simple shape. Instead, it is bounded by a convex curve described by a differential equation; the closed-form solution is unknown. The complexity of even this special case remains open.

Krumke et. al. [17], motivated by processor allocation on the CM5, consider the constrained problem. They prove that for distances obeying the triangle inequality, a greedy algorithm gives a 2-approximation and they prove hardness of approximation for arbitrary distances.

In the context of reconfigurable computing on field-programmable gate arrays (FPGAs), we are faced with a generalization of the problem considered in this paper, as the size of processors may vary: The objective is to place a set of rectangular processors (modules) on a given grid, such that the overall weighted sum of Manhattan distances is minimized. See [1], who give an $O(n \log n)$ algorithm for finding an optimal feasible location for one additional module between a set of n existing modules. At this point, no results are known on the general off-line problem (place n modules simultaneously) or on on-line versions.

Another related problem is called *min-sum k -clustering*. separate a graph into k clusters to minimize the sum of pairwise distances between nodes in the same cluster. For general graphs Sahni and Gonzalez [27] show that this problem is NP-hard to approximate within any constant factor for $k \geq 3$. In a metric space the problem is easier to approximate: Guttmann-Beck and Hassin [13] give a 2-approximation, Indyk [14] gives a PTAS for $k = 2$, and Bartel et al. [2] give an $O(1/\epsilon \log^{1+\epsilon} n)$ -approximation for general k .

The problem of *maximizing* the average Manhattan distance has also been considered: Fekete and Meijer [12] give a PTAS for this *dispersion* problem in \mathbb{R}^d for constant d , and show that for any fixed k , an optimal set of k of n given points can be determined in time linear in n .

Our Results. This paper gives exact and approximate algorithms for minimizing the average Manhattan distance between allocated processors. One of these algorithms has been implemented on Cplant, a super-computer at Sandia National Laboratories. In particular, we have the following results:

- Improving on the previous best factor of 2, we prove that a natural median-based heuristic is a $\frac{7}{4}$ -approximation algorithm for $2D$ grids and show this analysis is tight.
- We present a simple generalization to general d -dimensional space with fixed d and prove that it is a $2 - \frac{1}{2d}$ -approximation algorithm, which is tight.
- We give an efficient polynomial-time approximation scheme (PTAS).
- We present a PTAS for the clustering problem in \mathbb{R}^d for constant d .
- We describe how our results have been applied in practice.

In addition, we have a number of other results whose details are omitted due to space constraints: We have a linear-time exact algorithm for the 1D case based on dynamic programming. We prove that the MC1x1 algorithm is a $2d$ -approximation. We can solve the 2-dimensional case for $k = 3$ in time $O(n \log n)$.

The rest of the paper is organized as follows. Section 2 gives a $\frac{7}{4}$ -approximation for two-dimensional point sets. Section 3 gives a PTAS in the plane. Section 4 sketches how to extend the 2-dimensional results to general, fixed d -dimensional space: We show that a median-based heuristic yields a $2 - \frac{1}{2d}$ approximation, and describe how to get a PTAS. Section 5 uses simulation results to discuss our results in the context of the original allocation problem. We conclude with Section 6.

2 The Manhattan Median Algorithm for Two-Dimensional Point Sets

2.1 Median-Based Algorithms

Given a set P of k points in the plane. A point that minimizes the total (Manhattan) distance to these points is called an (L_1) median: It is straightforward to deduce from the nature of Manhattan distances that this is a point both of whose x - and y -coordinates are medians of the given point sets. Clearly, we can always pick a median whose coordinates are from the coordinates in P . If k is odd, then there is a unique median; if k is even, possible median coordinates may form intervals.

The natural greedy algorithm is as follows:

Build the set of all possible medians by drawing a horizontal and vertical line through every point, and consider all the $O(n^2)$ intersection points. For each of these points p do:

1. Take the k points closest to p (using the L_1 metric).
2. Compute the total pairwise distance between all k points.

Return the set of k points with smallest pairwise distance.

We call this strategy MM, for **Manhattan Median** algorithm. We will see in the following that MM yields a performance ratio of $\frac{7}{4}$ on 2D meshes. (Note that this algorithm was shown to be 2-competitive in arbitrary metric spaces by Krumke et al. [17], who called it Gen-Alg.)

2.2 Analysis of the Algorithm

For $S \subseteq P$, let $|S|$ denote the sum of the pairwise L_1 distances between points in S . If p is a point in the plane, we use p_x and p_y to denote the x - and y -coordinate of p respectively.

Lemma 1 *MM is not better than a $7/4$ approximation.*

Proof: For a class of examples establishing the lower bound, consider the situation as shown in Figure 4. For any $\varepsilon > 0$, it consists of a cluster of $k/2$ points at $(0, 0)$, four clusters of $k/8$ points each at $(\pm 1, \pm 1)$, a cluster of $3k/8$ points at $(-1 - \varepsilon, 0)$, three clusters of $k/8$ points each at $(-2 - \varepsilon, 0)$, $(-1 - \varepsilon, \pm 1)$. Choosing $(0, 0)$ as median yields a total distance of $7k^2/16$ for *MM*(and more for any other median), while choosing the points at $(0, 0)$, $(-1, 0)$, and $(-1 - \varepsilon, 0)$ yields a total distance $k^2/4(1 + \Theta(\varepsilon))$. \square

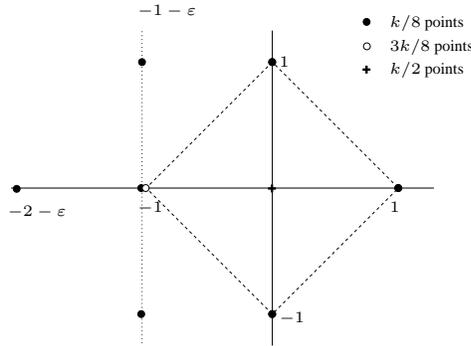


Figure 4: A class of examples where *MM* yields a ratio of $7/4$.

Now we will show that $7/4$ is indeed worst case. We will focus on possible worst-case arrangements and use local optimality to restrict the possible arrangements until the claim follows.

Let OPT be a subset of P of size k for which $|OPT|$ is minimal. Without loss of generality assume that the origin is a median point of OPT . This means that the number of points of OPT with positive or negative x - or y -coordinates is at most $k/2$. Let MM be the set of k points closest to the origin. (Since this is one candidate solution for the algorithm, its sum of pairwise distance is at least as high as that of the solution returned by the algorithm.)

Without loss of generality, assume that the largest distance of a point in MM to the origin is equal to 1, so MM lies in the unit circle C . We say that points are either inside C , on C or outside C . All points of S inside C are in MM and at least some points on C are in MM . If there are more than k points on and inside C , we select all points inside C plus those points on C that give the largest value for $|MM|$.

Clearly $1 \leq |MM|/|OPT|$. Let ρ_k be the supremum of $|MM|/|OPT|$. By assuming that ties are broken badly, we can assume that there is a configuration S for which $|MM|/|OPT| = \rho_k$.

Lemma 2 *For any n and k , there are point sets S with $|S| = n$ for which $|MM|/|OPT|$ attains the value ρ_k .*

Proof: The set of arrangements of n points in C is a compact set in $2d$ -dimensional space. By our assumption on breaking ties, $|MM|/|OPT|$ is upper semi-continuous, so it attains a maximum. \square

The following observation allows us to restrict our attention to a subset of possible k .

Lemma 3 *Let $k_1 < k_2$. Then $\rho_{k_1} \leq \rho_{k_2}$*

Proof: Let P be a configuration that attains the worst-case ratio ρ_{k_1} . Placing $k_2 - k_1$ points at the corresponding median yields at least the same performance for k_2 selected points, proving the claim. \square

Thus, the following suffices for the overall claim.

Lemma 4 For values of k that are multiples of 8 we have $\rho_k = 7/4$.

Proof: For ease of presentation, we assume without loss of generality that $S = MM \cup OPT$. Let $B = OPT \cap MM$, $O = OPT - B$ and $A = MM - B$. See Figure 5 (a).

Claim 0: No $p \in O$ lies outside C .

If a point $p \in O$ lies outside C we can move it a little closer to the origin without entering C . Since it remains outside C , the point does not become part of MM , so $|OPT|$ is reduced, $|MM|$ remains the same and the ratio $|MM|/|OPT|$ increases, which is impossible.

Claim 1: All points inside C are in MM .

It follows from the definition of MM that all points inside C are in MM . Notice that this implies that no point $p \in O$ can lie inside C .

Claim 2: The origin is also a median of MM .

Suppose that the origin is not a median of MM . Without loss of generality assume that there are more than $k/2$ points from MM with positive y -coordinate. So any median of MM has a positive y -coordinate. Moving the point of MM with the smallest positive y -coordinate downward moves it away from all medians of MM . So $|MM|$ increases and $|OPT|$ does not increase. So $|MM|/|OPT|$ increases, which is impossible.

Claim 3: No $p \in A$ lies inside C .

Suppose there is a $p \in A$ that lies inside C . Moving p away from the origin increases MM because p is moved further away from the median of MM . Since $p \notin OPT$, OPT does not increase, although it may decrease. So $|MM|/|OPT|$ increases, which is impossible. This implies that all points inside C are in B and that points from A and O lie on the boundary of C .

Claim 4: Without loss of generality we may assume that all points $p \in A$ on C lie in a corner of C .

Suppose $p \in A$ lies on an edge of C but not in a corner. Let D be the sum of the distances from p to all points in $MM - p$. Consider the set of all points q for which the sum of the distances from q to all points in $MM - p$ is equal to D . The set is a convex polygon P through p . Therefore we can move p along the edge of C on which it lies, so that it either moves outside of P , in which case $|MM|$ increases, or it remains on the boundary of P , in which case $|MM|$ remains equal or possibly increases if p leaves MM . In either case $|OPT|$ stays the same or decreases. If $|MM|$ increases and/or $|OPT|$ decreases, $|MM|/|OPT|$ increases which is impossible. If both stay the same, we can move p until it reaches a corner of C . For an illustration of what the configuration may look like see Figure 5(a).

Claim 5: Without loss of generality we may assume that all points in $O \cup B$ lie in a corner of C or on the origin.

We prove the claim by contradiction. Suppose there is a set of points S for which the claim is false. Let $p \in O \cup B$ be a point that does not lie in a corner of C or on the origin. Without loss of generality assume that $0 < p_y < 1$. We now define up to four sets of points. Let $Y^+(p)$ be the points in S with y -coordinate equal to p_y . If there is a point $q \in Y^+(p)$ with $q_x + q_y = 1$ then let $X^+(p)$ be the points in S with x -coordinate equal to q_x , otherwise $X^+(p)$ is empty. If there is a point $q \in Y^+(p)$ with $q_y - q_x = 1$ then let $X^-(p)$ be the points in S with x -coordinate equal to q_x . If there is a point $q \in X^+(p) \cup X^-(p)$ with $q_x + q_y = -1$ or $q_x - q_y = 1$ then let $Y^-(p)$ be the points in S with y -coordinate equal to q_y . So the four sets of points lie on four axis-parallel line segments in the disk that meet on C . Let $XY(p) = X^-(p) \cup X^+(p) \cup Y^-(p) \cup Y^+(p)$. The set $XY(p)$ is illustrated in Figure 5(b). We move the points in $XY(p)$ simultaneously, in such a way that they stay on four axis-parallel line segments meeting on C . We move all points in $Y^+(p)$ not on C upwards by ε . We move all points in $Y^+(p) \cap C$ upwards while remaining on C . Points in $X^-(p)$ move to the right, points in $X^+(p)$ move to the left and points in $Y^-(p)$ move down. We choose ε small enough

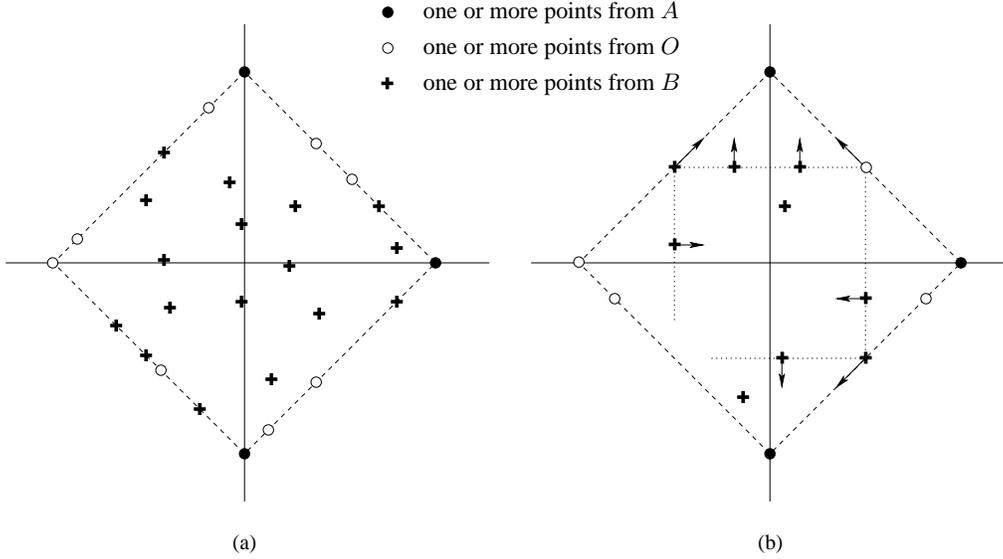


Figure 5: Points of A , O and B (a) after claim 4 and (b) during motion used in claim 5.

such that no point from $S \setminus XY(p)$ enters one of the four line segments defining $XY(p)$. This move changes $|MM|$ by some amount $\delta_a \varepsilon$ and $|OPT|$ by some amount $\delta_o \varepsilon$. However if we move all points in the opposite direction (i.e. point in $Y^+(p)$ downwards, etc.) $|MM|$ and $|OPT|$ change by $-\delta_a \varepsilon$ and $-\delta_o \varepsilon$ respectively. So if $\delta_a / \delta_o \neq \rho_k$, one of these two moves increases $|MM|/|OPT|$, which is impossible. If $\delta_a / \delta_o = \rho_k$ we keep moving the points in the same direction until there is a combinatorial change, i.e. a point in $XY(p)$ reaches C , a point in $XY(p)$ reaches a corner, or a point from $S \setminus XY(p)$ enters one of the line segments defining $XY(p)$. We can then repeat this argument until all points of S lie on C or on the origin.

We can now complete the proof. Let b denote the number of points at the origin. Let a_0, a_1, a_2, a_3 and o_0, o_1, o_2, o_3 be the points of MM and OPT at the north, east, south and west corners of C respectively. The value of $|MM|$ is $2 \sum_{0 \leq i < j \leq 3} a_i a_j + \sum_{0 \leq i \leq 3} b a_i = 2 \sum_{0 \leq i < j \leq 3} a_i a_j + b(k-b)$ which is maximal when all values a_i are equal to $\lfloor (k-b)/4 \rfloor$ or $\lceil (k-b)/4 \rceil$. The value of $|OPT|$ is $2 \sum_{0 \leq i < j \leq 3} o_i o_j + b(k-b)$ which is minimal when $o_0 = k-b$ and $o_1 = o_2 = o_3 = 0$. However the origin is the median of OPT so if $b < k/2$, the minimum value for $|OPT|$ occurs when $o_0 = k/2$ and $o_1 = k/2 - b$. Therefore if $b \geq k/2$ we have $\frac{|MM|}{|OPT|} \leq \frac{\frac{12(k-b)^2}{16} + b(k-b)}{b(k-b)}$ which is maximal when $b = k/2$ in which case $|MM|/|OPT| = 7/4$.

If $b < k/2$ we have $\frac{|MM|}{|OPT|} \leq \frac{\frac{12(k-b)^2}{16} + b(k-b)}{k(\frac{k}{2}-b) + b(k-b)}$ from which it follows that $\frac{|MM|}{|OPT|} \leq \frac{3k^2 - 2kb - b^2}{2k^2 - 4b^2}$. This is an increasing function of b in the interval $0 \leq b < k/2$ and approaches the value $7/4$. Therefore the lemma holds. \square

We summarize:

Theorem 1 MM is a $7/4$ -approximation algorithm for minimizing the sum of pairwise Manhattan distances in a $2D$ mesh.

3 A PTAS for Two Dimensions

Let $w(S, T)$ be the sum of all the distances between points in S and points in T . Let $w_x(S, T)$ and $w_y(S, T)$ be the sum of all the x - and y - distances between points in S and points in T respectively. So $w(S, T) = w_x(S, T) + w_y(S, T)$. Let $w(S) = w(S, S)$, $w_x(S) = w_x(S, S)$ and $w_y(S) = w_y(S, S)$.

Let $S = \{s_0, s_1, \dots, s_{k-1}\}$ be a minimal weight subset of P , where k is an integer greater than 1. We will label the x - and y -coordinates of a point $s \in S$ by some (x_a, y_b) with $0 \leq a < k$ and $0 \leq b < k$ such that

$x_0 \leq x_1 \leq \dots \leq x_{k-1}$ and $y_0 \leq y_1 \leq \dots \leq y_{k-1}$. (Note that in general, $a \neq b$ for a point $s = (x_a, y_b)$.) We can derive the following equation: $w_x(S) = (k-1)(x_{k-1} - x_0) + (k-3)(x_{k-2} - x_1) + \dots$
 $w_y(S) = (k-1)(y_{k-1} - y_0) + (k-3)(y_{k-2} - y_1) + \dots$ We show that there is a polynomial time approximation scheme (PTAS), i.e., for any fixed positive $m = 1/\varepsilon$, there is a polynomial approximation algorithm that finds a solution that is within $(1 + \varepsilon)$ of the optimum.

The basic idea is similar to the one used in [12] for the problem of selecting a set of points that maximizes the overall distance: We find (by enumeration) a subdivision of an optimal solution into $m \times m$ rectangular cells C_{ij} , each of which must contain a specific number k_{ij} of selected points. From each cell C_{ij} , the points are selected in a way that guarantees that the total distance to all other cells except for the $m-1$ cells in the same ‘‘horizontal’’ strip or the $m-1$ cells in the same ‘‘vertical’’ strip is minimized. As it turns out, this can be done in a way that the total neglected distance within the strips is bounded by a small fraction of the weight of an optimal solution, yielding the desired approximation property. See Figure 6 for the setup.

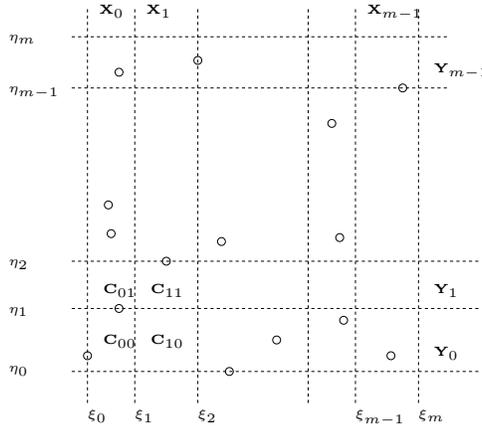


Figure 6: Dividing the point set in horizontal and vertical strips.

For ease of presentation we assume that k is a multiple of m and $m > 2$. Approximation algorithms for other values of k can be constructed in a similar fashion. Consider an optimal solution of k points, denoted by OPT . Furthermore consider a division of the plane by a set of $m+1$ x -coordinates $\xi_0 \leq \dots \leq \xi_1 \leq \dots \leq \xi_m$. Let $X_i := \{p = (x, y) \mid \xi_i \leq x \leq \xi_{i+1}, 0 \leq i < m\}$ be the vertical strip between coordinates ξ_i and ξ_{i+1} . By enumeration of possible choices of ξ_0, \dots, ξ_m we may assume that the ξ_i have the property that, for an optimal solution, from each of the m strips X_i precisely k/m points of P are chosen. (A small perturbation does not change optimality or approximation properties of solutions. This shows that in case of several points sharing the same coordinates, ties may be broken arbitrarily; in that case, points on the boundary between two strips may be considered belonging to one or the other of those strips, whatever is convenient to reach the appropriate number of points in a strip.)

In a similar manner, suppose we know $m+1$ y -coordinates $\eta_0 \leq \eta_1 \leq \dots \leq \eta_m$ such that from each horizontal strip $Y_i := \{p = (x, y) \mid \eta_i \leq y \leq \eta_{i+1}, 0 \leq i < m\}$ a subset of k/m points are chosen for an optimal solution.

Let $C_{ij} := X_i \cap Y_j$, and let k_{ij} be the number of points in OPT that are chosen from C_{ij} . Since

$$\sum_{0 \leq i < m} k_{ij} = \sum_{0 \leq j < m} k_{ij} = k/m,$$

we may assume by enumeration over the $O(k^m)$ possible partitions of k/m into m pieces that we know all the numbers k_{ij} .

Finally, define the vector $\nabla_{ij} := ((2i+1-m)k/m, (2j+1-m)k/m)$. Now our approximation algorithm is as follows: from each cell C_{ij} , choose some k_{ij} points that are minimal in direction ∇_{ij} , i.e.,

select points $p = (x, y)$ for which $(x(2i + 1 - m)k/m, y(2j + 1 - m)k/m)$ is minimal. For an illustration, see Figure 7.

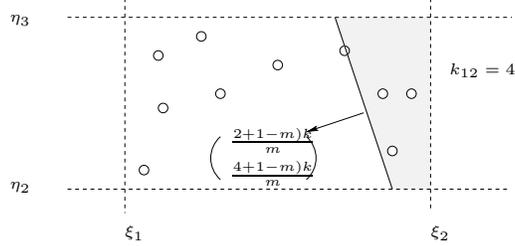


Figure 7: Select points in cell C_{12} .

It can be shown that if we select points in this way from each cell, we minimize the sum of the x -distances from each point in C_{ij} to points not in X_i and the y -distances to points not in Y_j . (Overlap between the selections from different cells is avoided by proceeding in lexicographic order of cells, and choosing the k_{ij} points among the candidates that are still unselected.)

Details are somewhat technical and described in Appendix A. We summarize:

Theorem 2 *The problem of selecting a subset of minimum total Manhattan distance for a set of points in \mathbb{R}^2 allows a PTAS.*

4 Higher-Dimensional Spaces

Using our techniques from the previous sections, it is not too hard to get generalizations to higher dimensions. We start by describing the performance of *MM*.

4.1 A $(2 - 1/2d)$ -Approximation

As in two-dimensional space, we enumerate over all possible medians, using the $O(n^d)$ combinations of point coordinates. From each median, we pick the k points that are closest under L_1 distances.

Lemma 5 *MM is not better than a $2 - 1/2d$ approximation.*

Proof: Consider the following class of examples, based on the cross-polytope in d dimensions, i.e., the d -dimensional L_1 unit ball. Let $\varepsilon > 0$. The example consists of a cluster of $k/2$ points at $(0, \dots, 0)$; in addition, we have $2d$ clusters of $k/4d$ points each at $(\pm e_i)$, where e_i is the i th unit vector. Moreover, we have a cluster of $(\frac{k}{2} - \frac{4}{4d})$ points at $(-1 - \varepsilon, 0, \dots, 0)$, and clusters of $k/4d$ points at $(-2 - \varepsilon, 0, \dots, 0)$, and $(-1 - \varepsilon, 0, \dots, 0) \pm e_i$. Choosing the origin as median and performing *MM* yields a total distance of $\frac{k^2}{4} (2 - \frac{1}{2d})$; all other choices yield a worse sum. On the other hand, picking the $k/2$ points at the origin, and the $k/2$ points near $-e_1$ yields a total distance of $\frac{k^2}{4} (1 + \Theta(\varepsilon))$. \square

Establishing a matching upper bound can be done analogously to Section 2. Lemmas 2 and 3 hold for general dimensions. The rest is based on the following general lemma:

Lemma 6 *Worst-case arrangements for MM can be assumed to have all points at positions $(0, \dots, 0)$ and $\pm e_i$, where e_i is the i th unit vector.*

Sketch of Proof. Consider a worst-case arrangement within the cross-polytope centered at the origin, with radius 1. Local moves consist of continuous changes in point coordinates, performed in such a way that the existing number of coordinate identities is kept. This means that if there is a point to be moved at a coordinate different from 0, 1, -1, then all other points sharing that coordinate are moved in a way that changes keeps the identical coordinates the same, analogous to Figure 5 (b).

Note that under these moves, the functions OPT and MM are locally linear, so the ratio of MM and OPT is locally constant, strictly monotonically decreasing, or strictly monotonically increasing. If the ratio is decreasing with respect to a move, it must be increasing with respect to the opposite move; this means the arrangement was not worst-case optimal to start with.

If the ratio stays locally constant during a move, it will continue to be extremal until an event occurs, i.e., when the number of coordinate identities between points increases, or the number of point coordinates at 0, 1, -1 increase. While there are points with coordinates different from 0, 1, -1 , there is always a move that decreases the total degrees of freedom, until all dn degrees of freedom have been eliminated. This means we can always reach an arrangement that fixes the dn point coordinates to be from the set $\{0, 1, -1\}$. These leaves as only positions within the cross-polytope the origin and the $2d$ positions $\pm e_i$. \square

Using symmetry, the remaining restricted set of arrangements can be evaluated quite easily. This yields

Theorem 3 *For points in d -dimensional space, MM is a $2 - 1/2d$ -approximation algorithm, which is tight.*

4.2 A PTAS for General Dimensions

Theorem 4 *For any fixed d , the problem of selecting a subset of minimum total Manhattan distance for a set of points in \mathbb{R}^d allows a PTAS.*

Sketch of Proof. For any chosen $m = \Theta(1/\varepsilon)$, we subdivide the set of n points by $d(m + 1)$ axis-aligned hyperplanes, such that $(m + 1)$ are normal for each coordinate direction. Moreover, any set of $(m + 1)$ hyperplanes normal to the same coordinate axis is assumed to subdivide the optimal solution into k/m equal subsets, called *slices*. Enumeration of all possible structures of this type yields a total of n^m choices of hyperplanes in each coordinate, for a total of n^{md} possible choices. For each choice, we have a total of m^d cells, each containing between 0 and k points; thus, there are $O(m^{kd})$ different distributions of cardinalities to the different cells.

Just like in the two-dimensional case, each cell has a corresponding gradient direction. This allows it to pick for each cell the assigned number of points that are extremal in this gradient direction.

It is easily seen that for each coordinate x_i , the above choice minimizes the total sum of x_i -distances between points not in the same x_i -slice. The remaining technical part (showing that the sum of distances within slices are small compared to the distances between different slices) is analogous to the details described in Appendix A and omitted. \square

5 Experiments

Although this paper has focused on allocating a single job, the real allocator makes a decision for each job that is scheduled and which processors are available for each job depends on the previous allocations. In order to understand the interaction between the quality of an individual allocation and the quality of future allocations, we ran a simulation involving pairs of algorithms. One algorithm, the *situation algorithm*, is run normally. Each allocation decision is treated as an input for the other algorithm, called the *decision algorithm*. The sum of pairwise distances for the decision algorithm are recorded as the result for that pair.

Our simulation used the algorithms MC1x1, MM, MM+Inc, and HilbertBF. MM+Inc takes the allocation of MM and then tries to improve it by replacing a processor included in the allocation with an excluded processor until a local minimum is reached. HilbertBF is the 1-dimensional strategy developed by Leung et al. [19] currently being run on Cplant. The simulation used the LLNL Cray T3D trace from the Parallel Workloads Archive [11]. This trace consists of 21323 jobs running on a machine with 256 processors, treated as a 16×16 mesh for the simulation. Table 1 shows the results.

Observe that, in each row, the algorithms are ranked best to worst as MM+Inc, MM, MC1x1, and HilbertBF. This is consistent with their worst-case results since MM is a $7/4$ -approximation, MC1x1 is a 4-approximation, and HilbertBF has an unbounded approximation factor.² However, looking just at the

²On an $N \times N$ mesh, the approximation ratio can be $\Omega(N)$.

Situation Algorithm	Decision Algorithm			
	MC1x1	MM	MM+Inc	HilbertBF
MC1x1	5256	5218	5207	5432
MM	5323	5285	5276	5531
MM+Inc	5319	5281	5269	5495
HilbertBF	5090	5059	5046	5207

Table 1: Average sum of pairwise distances when the decision algorithm makes allocations with input provided by the situation algorithm.

diagonal entries, where the free processors depend on the same algorithm’s previous decisions, give the ranking (from best to worst) HilbertBF, MM, MC1x1, and MM+Inc. The locally-better decisions of MM+Inc seem to paint the algorithm into a corner over time.

We confirmed that locally-better decisions are not best for an entire trace using Proximity [31, 32], which simulates messages moving through the network. We ran the NASA Ames iPSC/860 trace from the Parallel Workloads Archive [11], but scaled the number of processors for each job down by a factor of 4. This made the trace run on a machine with 32 processors, allowing us to solve for the optimal at each step. In terms of average job flow time, MC1x1 was best, followed by MM, and then the optimal. (MM+Inc was not considered in this simulation. HilbertBF was much worse than all three of the algorithms mentioned, but this is at least partially because of problems with the curve for machines that are not square meshes whose dimensions are powers of 2.)

6 Conclusions

We have presented a number of new results on a natural generalization of a classical problem. Clearly, there are several interesting extensions. As indicated by our experiments, studying the online version of the problem may be of particular interest and relevance. We hope to present results on this aspect in future work.

Acknowledgments

We would like to thank Moe Jette and Bill Nitzberg for providing the LLNL and NASA Ames iPSC/860 traces, respectively, to the Parallel Workloads Archive.

References

- [1] A. Ahmadinia, C. Bobda, S. Fekete, J. Teich, and J. der Veen. Optimal routing-conscious dynamic placement for reconfigurable computing. In *International Conference on Field-Programmable Logic and its applications*, 2004. To appear. Available at <http://arxiv.org/abs/cs.DS/0406035>.
- [2] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k -clustering in metric spaces. In *Proc. 33rd Symp. on Theory of Computation*, pages 11–20, 2001.
- [3] S. Baylor, C. Benveniste, and Y. Hsu. Performance evaluation of a massively paralel I/O subsystem. In R. Jain, J. Werth, and J. Browne, editors, *Input/Output in parallel and distributed computer systems*, volume 362 of *The Kluwer International Series in Engineering and Computer Science*, chapter 13, pages 293–311. Kluwer Academic Publishers, 1996. http://www.research.ibm.com/people/b/baylor/papers/sjb_io94.ps.
- [4] C. M. Bender, M. A. Bender, E. Demaine, and S. Fekete. What is the optimal shape of a city? *Journal of Physics A: Mathematical and General*, 37:147–159, 2004.
- [5] S. Bhattacharya and W.-T. Tsai. Lookahead processor allocation in mesh-connected massively parallel computers. In *Proc. 8th International Parallel Processing Symposium*, pages 868–875, 1994.
- [6] R. Brightwell, L. A. Fisk, D. S. Greenberg, T. Hudson, M. Levenhagen, A. B. Maccabe, and R. Riesen. Massively parallel computing using commodity components. *Parallel Computing*, 26(2-3):243–266, 2000.

- [7] D. Bunde, V. Leung, and J. Mache. Communication patterns and allocation strategies. In *Proc. 3rd Int. Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems*, 2004.
- [8] C. Chang and P. Mohapatra. Improving performance of mesh connected multicomputers by reducing fragmentation. *Journal of Parallel and Distributed Computing*, 52(1):40–68, 1998.
- [9] P.-J. Chuang and N.-F. Tzeng. An efficient submesh allocation strategy for mesh computer systems. In *Proc. International Conf. on Distributed Computer Systems*, pages 256–263, 1991.
- [10] Cray Inc. Network queuing environment. <http://www.cray.com/products/software/nqe.html>.
- [11] D. Feitelson. The parallel workloads archive. <http://www.cs.huji.ac.il/labs/parallel/workload/index.html>.
- [12] S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2004.
- [13] N. Guttman-Beck and R. Hassin. Approximation algorithms for minimum sum p -clustering. *Disc. Appl. Math.*, 89:125–142, 1998. <http://www.math.tau.ac.il/~hassin/cluster.ps.gz>.
- [14] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proc. 40th Annual IEEE Symp. Found. Comp. Science (FOCS)*, pages 154–159, 1999.
- [15] R. M. Karp, A. C. McKellar, and C. K. Wong. Near-optimal solutions to a 2-dimensional placement problem. *SIAM Journal on Computing*, 4:271–286, 1975.
- [16] P. Krueger, T.-H. Lai, and V. Dixit-Radiya. Job scheduling is more important than processor allocation for hypercube computers. *IEEE Trans. on Parallel and Distributed Systems*, 5(5):488–497, 1994.
- [17] S. Krumke, M. Marathe, H. Noltemeier, V. Radhakrishnan, S. Ravi, and D. Rosenkrantz. Compact location problems. *Theoretical Computer Science*, 181(2):379–404, 1997.
- [18] Lawrence Livermore National Laboratory. Advanced Simulation and Computing (ASCI). <http://www.llnl.gov/asci/>.
- [19] V. Leung, E. Arkin, M. Bender, D. Bunde, J. Johnston, A. Lal, J. Mitchell, C. Phillips, and S. Seiden. Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies. In *Proc. 4th IEEE International Conference on Cluster Computing*, pages 296–304, 2002.
- [20] K. Li and K.-H. Cheng. A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system. *Journal of Parallel and Distributed Computing*, 12:79–83, 1991.
- [21] V. Lo, K. Windisch, W. Liu, and B. Nitzberg. Non-contiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Computing*, 8(7), 1997.
- [22] J. Mache and V. Lo. Dispersal metrics for non-contiguous processor allocation. Technical Report CIS-TR-96-13, University of Oregon, 1996.
- [23] J. Mache and V. Lo. The effects of dispersal on message-passing contention in processor allocation strategies. In *Proc. Third Joint Conference on Information Sciences, Sessions on Parallel and Distributed Processing*, volume 3, pages 223–226, 1997.
- [24] J. Mache, V. Lo, and K. Windisch. Minimizing message-passing contention in fragmentation-free processor allocation. In *Proc. 10th International Conf. Parallel and Distributed Computing Systems*, pages 120–124, 1997.
- [25] S. Moore and L. Ni. The effects of network contention on processor allocation strategies. In *Proc. 10th International Parallel Processing Symposium*, pages 268–274, 1996.
- [26] NASA. The portable batch system. <http://www.nas.nasa.gov/Software/PBS/>.
- [27] S. Sahni and T. Gonzalez. p -complete approximation problems. *JACM*, 23(3):555–565, 1976.
- [28] Sandia National Laboratories. The Computational Plant Project. <http://www.cs.sandia.gov/cplant>.
- [29] Sandia National Laboratories. Red storm. <http://www.cs.sandia.gov/platforms/RedStorm.html>.
- [30] V. Subramani, R. Kettimuthu, S. Srinivasan, J. Johnson, and P. Sadayappan. Selective buddy allocation for scheduling parallel jobs on clusters. In *Proc. 4th IEEE International Conference on Cluster Computing*, 2002.
- [31] University of Oregon Resource Allocation Group. Procsimity. <http://www.cs.uoregon.edu/research/DistributedComputing>.
- [32] K. Windisch, J. Miller, and V. Lo. Procsimity: An experimental tool for processor allocation and scheduling in highly parallel systems. In *Proc. Fifth Symp. on the Frontiers of Massively Parallel Computation*, pages 414–421, 1995. <ftp://ftp.cs.uoregon.edu/pub/lo/procsimity.ps.gz>.
- [33] Y. Zhu. Efficient processor allocation strategies for mesh-connected parallel computers. *J. Parallel and Distributed Computing*, 16:328–337, 1992.

A Correctness of the PTAS

A.1 Notation

Let MM be the point set selected by the algorithm described in Section 3. It is clear that MM can be computed in polynomial time. We will proceed by a series of lemmas to determine how well $w(MM)$ approximates $w(OPT)$. In the following, we consider the distances involving points from a particular cell C_{ij} . Let MM_{ij} be the set of k_{ij} points that are selected from C_{ij} by the heuristic, and let OPT_{ij} be a set of k_{ij} points of an optimal solution that are attributed to C_{ij} . Let $MM_{i\bullet}$, $OPT_{i\bullet}$, $MM_{\bullet j}$ and $OPT_{\bullet j}$ be the set of k/m points selected from X_i and Y_j by the heuristic and an optimal algorithm respectively. Finally $\overline{MM}_{i\bullet} := MM \setminus MM_{i\bullet}$, $\overline{MM}_{\bullet j} := MM \setminus MM_{\bullet j}$, $\overline{OPT}_{i\bullet} := OPT \setminus OPT_{i\bullet}$ and $\overline{OPT}_{\bullet j} := OPT \setminus OPT_{\bullet j}$.

For the rest of the notation notice that

$$w(HEU) = \sum_{i,j} [w_x(MM_{ij}, \overline{MM}_{i\bullet}) + w_y(MM_{ij}, \overline{MM}_{\bullet j})] + \sum_i w_x(MM_{i\bullet}) + \sum_j w_y(MM_{\bullet j}).$$

We first show that the first part is smaller than $w(OPT)$. We then show that the second and third part are small fractions of $w(HEU)$.

A.2 Details

Lemma 7 $w_x(MM_{ij}, \overline{MM}_{i\bullet}) + w_y(MM_{ij}, \overline{MM}_{\bullet j}) \leq w_x(OPT_{ij}, \overline{OPT}_{i\bullet}) + w_y(OPT_{ij}, \overline{OPT}_{\bullet j})$.

Proof: Consider a point $p \in OPT_{ij} \setminus MM_{ij}$. Thus, there is a point $p' \in MM_{ij} \setminus OPT_{ij}$ that was chosen by the heuristic instead of p . Let $p - p' = h = (h_x, h_y)$. When replacing p' in MM by p , we increase the x -distance to the ik/m points left of C_{ij} by h_x , while decreasing the x -distance to $(m - i - 1)k/m$ points right of C_{ij} by h_x . In the balance, this yields a change of $((2i + 1 - m)k/m)h_x$. Similarly, we get a change of $((2j + 1 - m)k/m)h_y$ for the y -coordinates. Since p' was chosen to minimize the inner product $\langle p', \nabla_{ij} \rangle$ we know that the inner product $\langle h, \nabla_{ij} \rangle \geq 0$, so the overall change of distances is positive.

Performing these replacements for all points in $MM \setminus OPT$, we can transform MM to OPT , while increasing the sum of distances $w_x(MM_{ij}, \overline{MM}_{i\bullet}) + w_y(MM_{ij}, \overline{MM}_{\bullet j})$ to the sum

$$w_x(OPT_{ij}, \overline{OPT}_{i\bullet}) + w_y(OPT_{ij}, \overline{OPT}_{\bullet j}).$$

□

Corollary 8

$$\sum_{i,j} w_x(MM_{ij}, \overline{MM}_{i\bullet}) + w_y(MM_{ij}, \overline{MM}_{\bullet j}) \leq w(OPT).$$

In the following two lemmas we show that

$$\sum_i w_x(MM_{i\bullet})$$

is a small fraction of $w(MM)$. Similar proofs can be given for

$$\sum_j w_y(MM_{\bullet j}).$$

Lemma 9

$$\sum_{0 < i < m-1} w_x(MM_{i\bullet}) \leq \frac{w_x(MM)}{2(m-2)}.$$

Proof: Let $\delta_i = \xi_{i+1} - \xi_i$. Since $i(m-i-1) \geq m-2$ for $0 < i < m-1$, we have for $0 < i < m-1$ $w_x(MM_{i\bullet}) \leq \frac{k^2}{2m^2} \delta_i \leq \frac{ik(m-i-1)k}{m} \delta_i \frac{1}{2(m-2)}$. Since MM has ik/m and $(m-i-1)k/m$ points to the left of ξ_i and right of ξ_{i+1} respectively, we have

$$w_x(MM) \geq \sum_{0 < i < m-1} \frac{ik(m-i-1)k}{m} \delta_i$$

so

$$\sum_{0 < i < m-1} w_x(MM_{i\bullet}) \leq \frac{1}{2(m-2)} w_x(MM).$$

□

Lemma 10 For $i = 0$ and $i = m-1$ we have $w_x(MM_{i\bullet}) \leq \frac{w_x(MM)}{m-1}$.

Proof: Without loss of generality assume $i = 0$. Let $x_0, x_1, \dots, x_{(k/m)-1}$ be the x -coordinates of the points $p_0, p_1, \dots, p_{(k/m)-1}$ in $MM_{0\bullet}$. So

$$\begin{aligned} w_x(MM_{0\bullet}) &= \left(\frac{k}{m} - 1\right) \left(x_{\frac{k}{m}-1} - x_0\right) + \left(\frac{k}{m} - 3\right) \left(x_{\frac{k}{m}-2} - x_1\right) + \dots \\ &\leq \left(\frac{k}{m} - 1\right) (\xi_1 - x_0) + \left(\frac{k}{m} - 3\right) (\xi_1 - x_1) + \dots \\ &\leq \frac{k}{m} (\xi_1 - x_0) + \frac{k}{m} (\xi_1 - x_1) + \dots \\ &\leq \frac{k}{m} (\xi_1 - x_0) + \frac{k}{m} (\xi_1 - x_1) + \dots + \frac{k}{m} \left(\xi_1 - x_{\frac{k}{m}-1}\right). \end{aligned}$$

Since $\xi_1 - x_j \leq x - x_j$ where $0 \leq j < k/m$ and x is the x -coordinate of any point in $\overline{MM}_{0\bullet}$ and since there are $(m-1)k/m$ points in $\overline{MM}_{0\bullet}$, we have $\xi_1 - x_j < \frac{m}{(m-1)k} w_x(p_j, \overline{MM}_{0\bullet})$ so

$$\begin{aligned} w_x(MM_{0\bullet}) &\leq \frac{k}{m} \frac{m}{(m-1)k} \sum_{0 \leq i < \frac{k}{m}} w_x(p_i, \overline{MM}_{0\bullet}) \\ &\leq \frac{1}{m-1} \sum_{0 \leq i < \frac{k}{m}} w_x(p_i, \overline{MM}_{0\bullet}) \\ &= \frac{1}{m-1} w_x(MM_{0\bullet}, \overline{MM}_{0\bullet}) \\ &\leq \frac{1}{m-1} w_x(MM). \end{aligned}$$

□

A.3 Result

Combining the three lemmas we get the claimed result.

$$\begin{aligned} w(MM) &= \sum_{i,j} w_x(MM_{ij}, \overline{MM}_{i\bullet}) + w_y(MM_{ij}, \overline{MM}_{\bullet j}) + \sum_i w_x(MM_{i\bullet}) + \sum_j w_y(MM_{\bullet j}) \\ &\leq w(OPT) + \frac{1}{2(m-2)} (w_x(MM) + w_y(MM)) + \frac{2}{m-1} (w_x(MM) + w_y(MM)) \\ &= w(OPT) + \frac{1}{2(m-2)} w(MM) + \frac{2}{m-1} w(MM). \end{aligned}$$

So $w(MM)(1 - \frac{1}{2(m-2)} - \frac{2}{m-1}) \leq w(OPT)$.