Communication-Aware Processor Allocation for Supercomputers: Finding Point Sets of Small Average Distance

Michael A. Bender[1], David P. Bunde[2], Erik D. Demaine[3], Sándor P. Fekete[4], Vitus J. Leung[5], Henk Meijer[6], and Cynthia A. Phillips[5]

[1] Department of Computer Science,
SUNY Stony Brook, Stony Brook, NY 11794-4400, USA.
bender@cs.sunysb.edu.
[2] Department of Computer Science,
Knox College, Galesburg, IL 61401, USA.
dbunde@knox.edu.
[3] MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA 02139, USA.
edemaine@mit.edu.
[4] Algorithms Group,
Department of Computer Science,
Braunschweig University of Technology,
D–38106 Braunschweig, Germany.
s.fekete@tu-bs.de.
[5] Discrete Algorithms & Math Department,
Sandia National Laboratories, Albuquerque, NM 87185-1318, USA.
{vjleung, caphill}@sandia.gov.
[6] Department of Science,
Roosevelt Academy,
Middelburg (ZL), The Netherlands.
h.meijer@roac.nl.

Abstract. We give processor-allocation algorithms for grid architectures, where the objective is to select processors from a set of available processors to minimize the average number of communication hops.
The associated clustering problem is as follows: Given n points in $\Re^d$, find a size-k subset with minimum average pairwise L_1 distance. We present a natural approximation algorithm and show that it is a $\frac{7}{4}$-approximation for two-dimensional grids; in d dimensions, 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 we report on experimental results.

1 Introduction

We give processor-allocation algorithms for grid architectures. Our objective is to select processors to run a job from a set of available processors so that the

average number of communication hops between processors assigned to the job is minimized. Our problem is restated as follows: given a set P of n points in $\Re^d$, find a subset S of k points with minimum average pairwise L_1 distance.

Motivation: Processor Allocation in Supercomputers. Our algorithmic work is motivated by a problem in the operation of supercomputers. Specifically, our algorithms and simulations targeted Cplant [6, 25], a commodity-based supercomputer developed at Sandia National Laboratories, and Red Storm, a custom supercomputer developed at Cray. Other supercomputers at Sandia have similar features. Red-Storm-like machines exist and are available as Cray XT3/4s. In these systems, a scheduler selects the next job to run based on priority. The allocator then independently places the job on a set of processors which exclusively run that job to completion. Security and memory constraints forbid migration, preemption, or multitasking. These constraints make the allocation decision more important since it cannot be changed once made.

To obtain maximum throughput in a network-limited computing system, the processors allocated to a single job should be physically near each other. This placement reduces communication costs and avoids bandwidth contention caused by overlapping jobs. Experiments have shown that allocating nearby processors to each job can improve throughput on a range of architectures [3, 16, 20, 21, 23, 28]. Several papers suggest that minimizing the *average number of communication hops* is an appropriate metric for job placement [15, 20, 21]. Experiments with a communication test suite demonstrate that this metric correlates with a job's completion time [16].

Early processor-allocation algorithms allocate only convex sets of processors to each job [5, 8, 18, 30]. For such allocations, each job's communication can be routed entirely within processors assigned to that job, so jobs contend only with themselves. But requiring convex allocations reduces the achievable system utilization to levels unacceptable for a high-performance system that has to serve a wide community of users [14, 26].

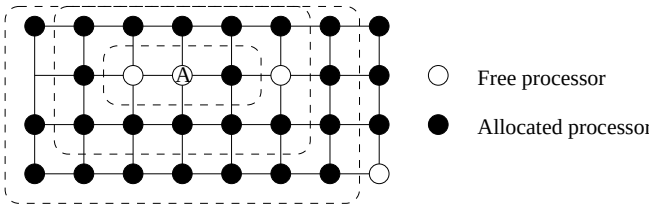

Fig. 1. Illustration of MC: Shells around processor A for a 3×1 request.

Recent work [7, 16, 19, 22, 26, 28] allows discontiguous allocation of processors but tries to cluster them and minimize contention with previously allocated jobs. Mache, Lo, and Windisch [22] propose the MC algorithm for grid architectures:

For each free processor, algorithm MC evaluates the quality of an allocation centered on that processor. It counts the number of free processors within a submesh of the requested size centered on the given processor and within "shells" of processors around this submesh; see Figure 1 reproduced from [22]. The cost of an allocation is the sum of the shell numbers of the allocated processors. MC chooses the allocation with lowest cost. Since users at Sandia do not request processors in a particular shape, in this paper, we consider MC1x1, a variant in which shell 0 is 1×1 and subsequent shells grow in the same way as in MC.

Originally, processor allocation on the Cplant system was *not* based on the locations of the free processors. The allocator simply verified that enough processors were free before dispatching a job. The last allocator used space-filling curves and 1D bin-packing techniques based upon work of Leung et al. [16]. We also had Cplant implementations of a 3D version of MC1x1 and the greedy heuristic (called MM) analyzed in this paper. Cplant has since been retired from active service at Sandia.

Related Algorithmic Work. Krumke et al. [15] consider a generalization of our problem on arbitrary topologies for several measures of locality, motivated by allocation on the CM5. They prove it is NP-hard to approximate average pairwise distance in general, but give a 2-approximation for distances obeying the triangle inequality.

A natural special case of the allocation problem is the *unconstrained* problem, in the absence of occupied processors: For any number k, find k grid points minimizing average pairwise L_1 distance. For moderate values of k, these sets can be found by exhaustive search; see Figure 2. The resulting shapes appear to approximate some "ideal" rounded shape, with better and better approximation for growing k. Karp et al. [13] and Bender et al. [4] study the exact nature of this shape, shown in Figure 3. Surprisingly, there is no known closed-form solution for the resulting convex curve, but Bender et al. [4] have expressed it as a differential equation. The complexity of this special case remains open, but its mathematical difficulty suggests the hardness of obtaining good solutions for the general constrained problem.

In reconfigurable computing on field-programmable gate arrays (FPGAs), varying processor sizes give rise to a generalization of our problem: place a set of rectangular modules on a grid to minimize the overall weighted sum of L_1 distances between modules. Ahmadinia et al. [1] give an optimal $\Theta(n \log n)$ algorithm for finding an optimal feasible location for a module given a set of n existing modules. At this point, no results are known for the general off-line problem (place n modules simultaneously) or for on-line versions.

Another related problem is *min-sum k-clustering*: separate a graph into k clusters to minimize the sum of distances between nodes in the same cluster. For general graphs, Sahni and Gonzalez [24] show it is NP-hard to approximate this problem to within any constant factor for $k \geq 3$. In a metric space, Guttmann-Beck and Hassin [11] give a 2-approximation, Indyk [12] gives a PTAS for $k = 2$, and Bartel et al. [2] give an $O((1/\epsilon) \log^{1+\epsilon} n)$-approximation for general k.

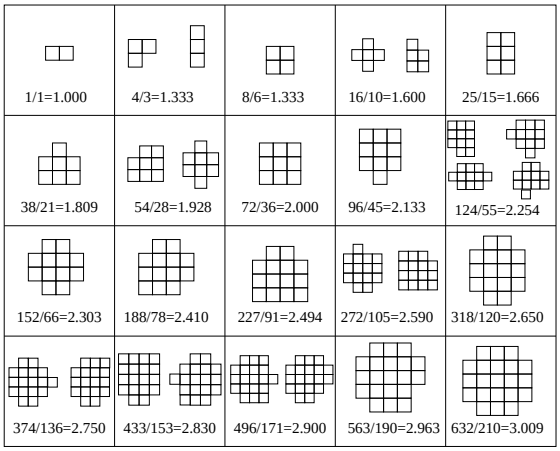

Fig. 2. Optimal unconstrained clusters for small values of k; numbers shown are the average L_1 distances, with truncated decimal values.

Fekete and Meijer [10] consider the problem of *maximizing* the average L_1 distance. They give a PTAS for this *dispersion* problem in $\Re^d$ for constant d, and show that an optimal set of any fixed size can be found in $O(n)$ time.

Our Results. We consider algorithms for minimizing the average L_1 distance between allocated processors in a mesh supercomputer. In particular, we give the following results:

- We prove that a greedy algorithm we call MM is a $\frac{7}{4}$-approximation algorithm for two-dimensional grids. This reduces the previous best factor of 2 [15]. We show that this analysis is tight.
- We present a simple generalization of MM to d-dimensional grids and prove that it gives a $2 - \frac{1}{2d}$ approximation, which is tight.
- We give a polynomial-time approximation scheme (PTAS) for points in $\Re^d$ for constant d.
- We prove that the d-dimensional version of MC1x1 has approximation factor at most d times that of MM.
- Using simulations, we compare the allocation performance of MM to that of other algorithms. As a byproduct, we get insight on how to place a stream of jobs in an online setting.

Our work also led to a linear-time dynamic programming algorithm for the 1-dimensional problem of points on a line or ring; see Leung et al. [17] for details.

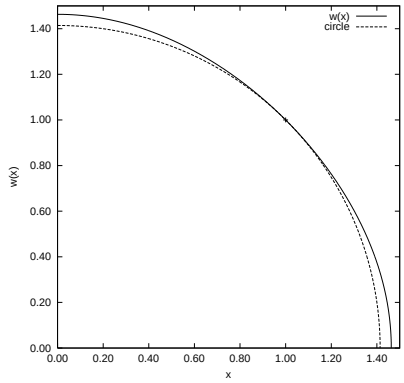

Fig. 3. Plot from Bender et al. [4] of a quarter of the optimal limiting boundary curve for the unconstrained problem; the dotted line is a circle.

2 The Manhattan Median Algorithm

2.1 Description of the Algorithm

Given a set S of k points in the plane, a point that minimizes the total L_1 distance to these points is called an (L_1) median. Given the nature of L_1 distances, this is a point whose x-coordinate (resp. y-coordinate) is the median of the x (resp. y) values of the given point set. We can always pick a median whose coordinates are from the coordinates in S. There is a unique median if k is odd; if k is even, possible median coordinates may form intervals.

The natural greedy algorithm for our clustering problem is as follows:

> Consider the set I containing the $O(n^d)$ intersection points of the horizontal and vertical lines through the points of input P. For each point $p \in I$ do:
>
> 1. Take the k points closest to p (using the L_1 metric), breaking ties arbitrarily.
> 2. Compute the total pairwise distance between all k points.
>
> Return the set of k points with smallest total pairwise distance.

We call this strategy MM, for **M**anhattan **M**edian. We prove that MM is a $\frac{7}{4}$-approximation on two-dimensional meshes; more generally, we show that MM is a $2 - \frac{1}{2d}$-approximation algorithm. (Note that Krumke et al. [15] call a minor variation of this algorithm Gen-Alg and show it is a 2-approximation in arbitrary metric spaces, making simple use of triangle inequality.)

2.2 Lower Bound

For $S \subseteq P$, let $w(S)$ denote the sum of L_1 distances between points in S. For a point p in the plane, we use p_x and p_y to denote its x- and y-coordinates, respectively.

Theorem 1. *MM is not better than a* $2 - \frac{1}{2d}$*-approximation.*

Proof. We construct an example based on the cross-polytope in d dimensions, i.e., the d-dimensional L_1 unit ball. Let $\varepsilon > 0$ be small. Denote the origin with O and the ith unit vector with e_i. The example has $k/2$ points at O and $O + e_1$. In addition, there are $k/(4d)$ points at $O - (1-\varepsilon)e_1$, $O + (2-\varepsilon)e_1$, $O \pm (1-\varepsilon)e_i$ for $i = 2, \ldots, d$, and $O + e_1 \pm (1-\varepsilon)e_i$ for $i = 2, \ldots, d$.

For $d = 2$, this corresponds to the situation shown in Figure 4: there are two "central" clusters of $k/2$ points at $(0,0)$ and $(1,0)$, and six "decoy" clusters of $k/8$ points at $(0, \pm(1-\epsilon))$, $(1, \pm(1-\epsilon))$, $(2-\epsilon, 0)$, and $(-1+\epsilon, 0)$. The optimal solution consists of the points at $(0,0)$ and $(1,0)$, which yield a total distance of $k^2/4$. The ε-perturbation prevents *MM* from finding this solution; instead, it is lured into picking the decoy clusters of size $k/8$, and any choice of median yields a set of points spanning a diamond: $k/2$ points in the center, and $k/8$ points in or near each of the surrounding corners of the L_1 unit ball, as indicated around $(0,0)$ in the figure. The resulting total distance is $7k^2(1 - \Theta(\epsilon))/16$, for a ratio of $2 - \frac{1}{4} = 7/4$.

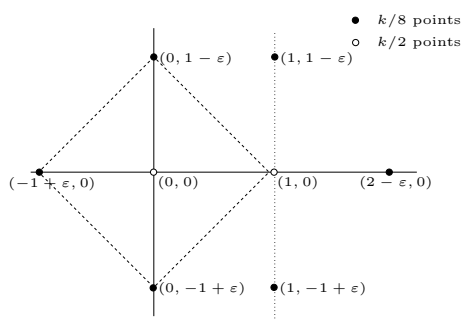

Fig. 4. A class of examples where MM yields a ratio of 7/4.

In general d, *MM* always picks a set of clusters in the shape of a cross-polytope, with $k/2$ points in the center, and $k/4d$ in each of the corners. The resulting total distance is $(k^2/4)\,(2 - 1/(2d))\,(1 + \Theta(\varepsilon))$. A better (in fact, optimal) solution consists of the points at O and $O + e_1$, giving a total distance of $k^2/4$. □

In the above class of examples, we chose an ε-perturbation to guarantee a bad choice, without having to resort to tie-breaking assumptions for a lower

bound. For the upper bound on the worst-case ratio established in the next section, it suffices to consider any kind of tie-breaking, without the need to employ perturbation.

2.3 Upper Bound

Now we show that $2 - \frac{1}{2d}$ for points in d-dimensional space (and thus, $\frac{7}{4}$ in the plane) is indeed the worst-case bound. We 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 $w(OPT)$ is minimum. Without loss of generality assume that the origin is a median point of OPT. This means that at most $k/2$ points of OPT have positive x_i-coordinates, and at most $k/2$ points have negative x_i-coordinates. 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 distances is at least as high as that of the solution returned by the algorithm.

Without loss of generality, assume that the largest L_1 distance of a point in MM to the origin is 1, so MM lies in the L_1 unit circle C. (Note that C is a cross-polytope; for the special case $d = 2$, it is diamond-shaped.) We say that points are either inside C, on C, or outside C. All points of P 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 maximizing $w(MM)$; this yields an upper bound on the value of $w(MM)$.

Clearly $1 \leq w(MM)/w(OPT)$. Let ρ_k be the supremum of $w(MM)/w(OPT)$ over all inputs P. By assuming that ties are broken badly, we can assume that there is an input for which $w(MM)/w(OPT) = \rho_k$:

Lemma 1. *For any n and k, there are point sets P with $w(P) = n$ for which $w(MM)/w(OPT)$ attains the value ρ_k.*

Proof. The set of arrangements of n points in the unit sphere C is a compact set in dn-dimensional space. By our assumption on breaking ties, $w(MM)/w(OPT)$ is upper semicontinuous, so it attains a maximum. □

We show that $w(MM)$ is at most $2 - \frac{1}{2d}$ times larger than $w(OPT)$.

Theorem 2. *MM is a $2 - \frac{1}{2d}$-approximation algorithm for minimizing the sum of pairwise L_1 distances in a d-dimensional mesh.*

Proof. For ease of presentation, we assume without loss of generality that $P = MM \cup OPT$. Let $B = OPT \cap MM$, $O = OPT - B$ and $A = MM - B$.

Claim 0: *No point $p \in O$ 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 $w(OPT)$ is reduced, $w(MM)$ remains the same and the ratio $w(MM)/w(OPT)$ 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: *Without loss of generality, we may assume that the origin is also a median of MM.*

Suppose that the origin is not a median of MM. We consider the case when more than $k/2$ points of MM have positive x_1-coordinate; the other cases are handled analogously. We set the x_1-coordinate of the point in MM with smallest positive x_1-coordinate to zero. By assumption, this causes the point to move away from at least as many points of MM as it moves toward. Thus, $w(MM)$ does not decrease. The origin is a median of OPT so $w(OPT)$ does not increase. Therefore, the ratio $w(MM)/w(OPT)$ cannot decrease. Since the ratio cannot increase by assumption, it must remain the same. Thus, we have constructed a point set achieving $w(MM)/w(OPT) = \rho_k$ with one fewer point having positive x_1-coordinate. Repeating this process will make some point on the line $x_1 = 0$ a median.

Claim 3: *No point $p \in A$ 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 $w(MM)/w(OPT)$ 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 L_1 distances from p to all points in $MM - p$. Consider the set Q of all points q for which the sum of the L_1 distances from q to all points in $MM - p$ is at most D. The sum of distances is the sum of convex functions so it is also a convex function and the set Q is a convex polygon through p. Therefore, we can move p along the edge of C on which it lies so that it either moves outside of Q or remains on the boundary of Q. In the former case, $w(MM)$ increases. In the latter, $w(MM)$ remains the same. In either case, $w(OPT)$ stays the same or decreases. If $w(MM)$ increases and/or $w(OPT)$ decreases, $w(MM)/w(OPT)$ 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. Let $S(p)$ be the points that lie on the axis-parallel rectangle through p with corners on C. The set $S(p)$ is illustrated in Figure 5(b). We move the points in $S(p)$ simultaneously in such a way that they stay on an axis-parallel box with corners on C.

For an intuitive description of this motion, consider $d = 2$, as shown in Figure 5(b). We move all points in $S(p)$ with maximal x_2-coordinates but not on C

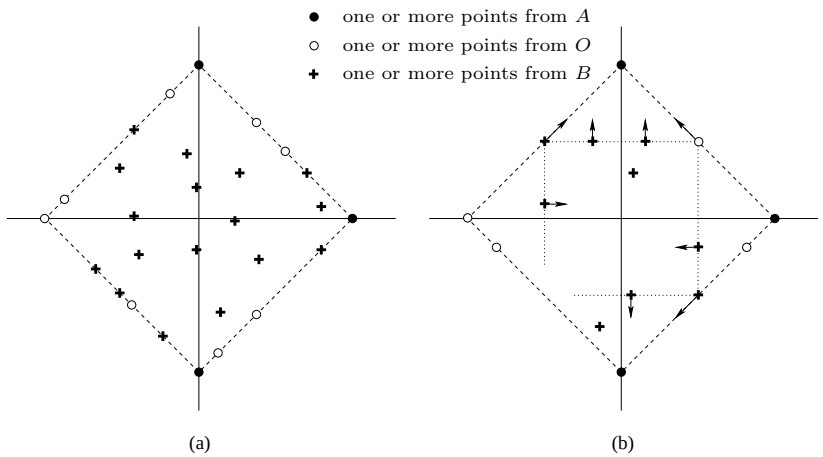

Fig. 5. Points of A, O and B (a) after claim 4 and (b) during motion used in claim 5.

upwards by ϵ. We move all points in $S(p)$ with maximal x_2-coordinates and on C upwards while remaining on C. Similarly the other points of $S(p)$ move either left, right or down. We choose ϵ small enough such that no point from $S \setminus S(p)$ enters the rectangle on which $S(p)$ lies. This move changes $w(MM)$ by some amount δ_a and $w(OPT)$ by some amount δ_o. However if we move all points in the opposite direction (i.e. points with maximal y-coordinates downwards, etc.) $w(MM)$ and $w(OPT)$ change by $-\delta_a$ and $-\delta_o$ respectively. So if $\delta_a/\delta_o \neq \rho_k$, one of these two moves increases $w(MM)/w(OPT)$, 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 from $S \setminus S(p)$ enters the rectangle on which $S(p)$ lies, a point in $S(p)$ reaches C, or the rectangle collapses into a line.

More generally, local moves consist of continuous changes in point coordinates, performed in a way that preserves the number of coordinate values. This means that to move a point having a coordinate value different from $0, 1, -1$, then all other points sharing that coordinate value are moved to keep the identical coordinates the same.

Note that under these moves, the functions OPT and MM are locally linear, so the ratio of MM and OPT is locally constant, strictly increasing, or strictly decreasing. If a move decreases the ratio, the opposite move increases it, contradicting the assumption that the arrangement is worst-case.

If the ratio is 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. Thus, we can always reach an arrangement with point coordi-

nates values from the set $\{0, 1, -1\}$. This leaves the origin and the $2d$ positions $\pm e_i$ as only positions within the cross-polytope.

We can now complete the proof of Theorem 2. Let b denote the number of points at the origin. These points are all in $B = OPT \cap MM$, because they were originally inside C. Let $x_1, \ldots, x_{2d}$ be the numbers of points in the $2d$ corners of C chosen by MM (forming the set X), and $z_1, \ldots, z_{2d}$ be the numbers of points in the $2d$ corners of C contained in OPT, forming the set Z. As both $\sum_{i=1}^{2d} x_i = k - b$ and $\sum_{i=1}^{2d} z_i = k - b$, the total sum of distances between points in the origin and points in the corners of C is $b(k-b)$ for both MM and OPT, resulting in $w(MM) = b(k-b) + w(X)$ and $w(OPT) = b(k-b) + w(Z)$. We will discuss bounds for $w(X)$ and $w(Z)$ and combine them to achieve the desired result.

An upper bound for $w(X)$ is obtained by considering continuous variables for the x_i; noting that the Manhattan distance between all distinct corners of C is 2, we see that $w(X)$ is bounded from above by $\max 2 \sum_{i \neq j} x_i x_j$ under the constraint $\sum_{i=1}^{2d} x_i = k - b$. It is an easy exercise to see that this maximum is achieved by making all x_i equal, i.e., $x_i = \frac{k-b}{2d}$. As there are $d(2d-1)$ distinct pairs of corners of C, this yields the upper bound of $\frac{(k-b)^2(2d-1)}{2d}$ for $w(X)$.

For the rest of the proof, we distinguish two cases, depending on the value of b. For $b \geq \frac{k}{2}$, we simply use $w(Z) \geq 0$, yielding the upper bound

$$\frac{w(MM)}{w(OPT)} = \frac{b(k-b) + w(X)}{b(k-b) + w(Z)} \tag{1}$$

$$\leq \frac{2db(k-b) + (k-b)^2(2d-1)}{2db(k-b)} \tag{2}$$

$$= 1 + \frac{(k-b)(2d-1)}{2db}. \tag{3}$$

The latter is maximized for b as small as possible, i.e., $b = \frac{k}{2}$, so

$$\frac{w(MM)}{w(OPT)} = \frac{b(k-b) + w(X)}{b(k-b) + w(Z)} \tag{4}$$

$$\leq 1 + \frac{(k/2)(2d-1)}{dk} \tag{5}$$

$$= 2 - \frac{1}{2d}, \tag{6}$$

as claimed.

For $b < \frac{k}{2}$, we note that $w(Z)$ is bounded from below by $\min 2 \sum_{i \neq j} z_i z_j$ under the constraint $\sum_{i=1}^{2d} z_i = k - b$; as the origin must be a median of OPT, the additional constraints $z_i \leq \frac{k}{2}$ can be imposed. Evaluating this relaxation is equally straightforward: an optimal solution is obtained by $z_0 = \frac{k}{2}$ and $z_1 = \frac{k}{2} - b$,

so $k\left(\frac{k}{2}-b\right)$ is a lower bound on $w(Z)$. Thus, we get

$$\frac{w(MM)}{w(OPT)} \leq \frac{2db(k-b)+(k-b)^2(2d-1)}{2db(k-b)+dk(k-2b)} \tag{7}$$

$$= \frac{(k-b)(2db+(k-b)(2d-1))}{d(2bk-2b^2+k^2-2bk)} \tag{8}$$

$$= \frac{(k-b)((2d-1)k+b)}{d(k^2-2b^2)}. \tag{9}$$

To see that the latter is bounded from above by $2-\frac{1}{2d}=\frac{4d-1}{2d}$, it suffices to show that

$$2(k-b)((2d-1)k+b) < (4d-1)(k^2-2b^2) \text{ for } b < \frac{k}{2},$$

which simplifies to

$$(4d-2)k^2-(4d-4)bk-2b^2 < (4d-1)k^2-(8d-2)b^2$$

or

$$(8d-4)b^2-(4d-4)bk-k^2 < 0.$$

To see the latter, note that the function $f(x)=(8d-4)x^2-(4d-4)kx-k^2$ is negative for $x=0$ and its first derivative $f'(x)=8(2d-1)x-4(d-4)k$ is positive for $0 \leq x \leq \frac{k}{2}$; as $f(\frac{k}{2})=0$, we conclude that $f(b)<0$ for all $b \in [0, \frac{k}{2}[$, and the claim holds.

□

3 Analysis of MC1x1

MC was originally presented as a heuristic algorithm, but we prove that MC1x1 has approximation ratio $(2-2/k)d$ in dimension d. Krumke et al. [15] used the same ideas to prove that a variant of MM is a $(2-2/k)$-approximation algorithm; their argument also applies to MM.

Theorem 3. *MC1x1 is a $(2-2/k)d$-approximation algorithm for minimizing the sum of pairwise L_1 distances in a d-dimensional mesh.*

Proof. Recall that MC1x1 minimizes the sum of the selected points' shell numbers. Let point v be the center of the shells for the selected allocation and let σ be the sum of the shell numbers for points of *MC1x1*. First, we bound $w(MC1x1)$ in terms of σ. The total distance from v to all points of *MC1x1* is at most σd since a point in shell i is at most id steps from v. Thus, $w(MC1x1) \leq (k-1)\sigma d$ since this is the distance if all paths are routed through v.

Now we bound $w(OPT)$ in terms of σ. For this, we use the concept of a star, which is a set of points with one identified as its center. The length of a star is the total distance between the center and its other points. The smallest star

with k points has length at least σ since a point distance i from the star's center is in the ith shell around that center. Thus, the total distance from one point of OPT to the others is at least σ. Since summing the lengths of stars of OPT with each point as the center counts the distance between each pair of points twice, $w(OPT) \geq k\sigma/2$ and the lemma follows by combining our bounds. □

4 Polynomial-Time Approximation Schemes

As noted in the introduction, the complexity of finding a subset of k points of small average Manhattan distance for a set of n points in d-dimensional space that has minimum average distance is still unknown. Given the fact that even the description of optimal subsets for the unconstrained, two-dimensional problem is unknown, we conjecture the following.

Conjecture 1 *Unless P=NP, there is no polynomial-time algorithm for finding a minimum average Manhattan distance subset for points in constant dimension $d \geq 2$.*

In the following, we give a polynomial-time approximation scheme (PTAS) for points in d-dimensional space. For ease of presentation and notation, we start with two-dimensional space, then describe necessary generalizations.

4.1 PTAS for Two-Dimensional Space

Let $w(S,T)$ be the sum of all the distances from points in S to points in T. Let $w_x(S,T)$ and $w_y(S,T)$ be the sum of x- and y- distances from points in S to 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 $w_y(S) = w_y(S,S)$. We call $w(S)$ the *weight* of S.

Let $S = \{s_0, s_1, \ldots, s_{k-1}\}$ be a minimum-weight subset of P, where k is an integer greater than 1. We 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 $x_0 \leq x_1 \leq \ldots \leq x_{k-1}$ and $y_0 \leq y_1 \leq \ldots \leq y_{k-1}$. (Note that in general, $a \neq b$ for a point $s = (x_a, y_b)$.) We can derive the following equations: $w_x(S) = (k-1)(x_{k-1} - x_0) + (k-3)(x_{k-2} - x_1) + \ \ldots$ and $w_y(S) = (k-1)(y_{k-1} - y_0) + (k-3)(y_{k-2} - y_1) + \ \ldots$ 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 within $(1+\varepsilon)$ of the optimum.

The basic idea is similar to the one used by Fekete and Meijer [10] to select a set of points maximizing the overall distance: We find (by enumeration) a subdivision of an optimal solution into $m \times m$ rectangular cells C_{ij}, each containing a specific number k_{ij} of selected points. The points from each cell C_{ij} are selected in a way that minimizes 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. 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.

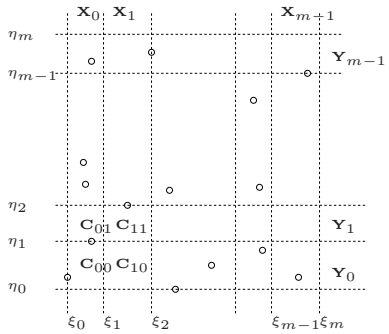

Fig. 6. Dividing the point set into horizontal and vertical strips.

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 a division of the plane by a set of $m+1$ x-coordinates $\xi_0 \leq \xi_1 \leq \ldots \leq \xi_m$. Let $X_i := \{p = (x, y) \mid \xi_i \leq x \leq \xi_{i+1}\}$ be the vertical strip between coordinates ξ_i and ξ_{i+1}. By enumeration of possible values of $\xi_0, \ldots, \xi_m$ we may assume that each of the m strips X_i contains precisely k/m points of an optimal solution. (A small perturbation does not change optimality or approximation properties of solutions. Thus, without loss of generality, we assume that no pair of points share either x-coordinate or y-coordinate.)

In a similar manner, assume we know $m+1$ y-coordinates $\eta_0 \leq \eta_1 \leq \ldots \leq \eta_m$ so that an optimal solution has precisely k/m points in each horizontal strip $Y_i := \{p = (x, y) \mid \eta_i \leq y \leq \eta_{i+1}\}$.

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 for all $i, j \in \{1, 2, \ldots, m\}$,

$$\sum_{0 \leq l < m} k_{lj} = \sum_{0 \leq l < m} k_{il} = k/m,$$

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 $\nabla_{ij} := ((2i + 1 - m)k/m, (2j + 1 - m)k/m)$. Our approximation algorithm is as follows: from each cell C_{ij}, choose k_{ij} points that are minimum 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 minimum. For an illustration, see Figure 7.

In the following we show that selecting points of C_{ij} this way minimizes the sum of x-distances to points not in X_i and the sum of y-distances to points not in Y_j. Before giving full technical details, we summarize:

Theorem 4. *The problem of selecting a subset of minimum total L_1 distance for a set of points in $\Re^2$ allows a PTAS.*

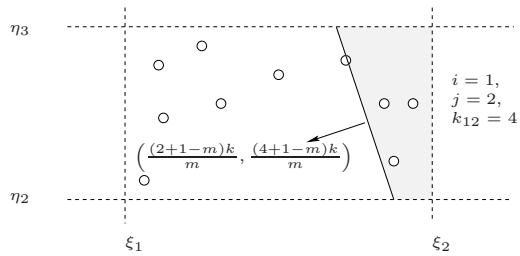

Fig. 7. Selecting points in cell C_{12}.

Correctness of the PTAS

Let APX be the point set selected by the algorithm described in Section 4. It is clear that APX can be computed in polynomial time. We will proceed by a series of lemmas to determine how well $w(APX)$ approximates $w(OPT)$. In the following, we consider the distances involving points from a particular cell C_{ij}. Let APX_{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 $APX_{i\bullet}$, $OPT_{i\bullet}$, $APX_{\bullet j}$ 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 $\overline{APX}_{i\bullet} := APX \setminus APX_{i\bullet}$, $\overline{APX}_{\bullet j} := APX \setminus APX_{\bullet j}$, $\overline{OPT}_{i\bullet} := OPT \setminus OPT_{i\bullet}$ and $\overline{OPT}_{\bullet j} := OPT \setminus OPT_{\bullet j}$.

For the rest of the notation notice that

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

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

Lemma 2.

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

Proof. Consider a point $p \in OPT_{ij} \setminus APX_{ij}$. We will replace it with an arbitrary point $p' \in APX_{ij} \setminus OPT_{ij}$ that was chosen by the heuristic instead of p. Let $p - p' = h = (h_x, h_y)$. When replacing p in OPT by p' in APX, we decrease the x-distance to the ik/m points left of C_{ij} by h_x, while increasing 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 negative.

Performing these replacements for all points in $OPT \setminus APX$, we can transform OPT to APX, while decreasing the sum of distances $w_x(OPT_{ij}, \overline{OPT}_{i\bullet}) + w_y(OPT_{ij}, \overline{OPT}_{\bullet j})$. to the sum $w_x(APX_{ij}, \overline{APX}_{i\bullet}) + w_y(APX_{ij}, \overline{APX}_{\bullet j})$ □

Corollary 1.

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

In the following two lemmas we show that

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

is a small fraction of $w(APX)$. Analogous proofs can be given for

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

Lemma 3.

$$\sum_{0<i<m-1} w_x(APX_{i\bullet}) \;\leq\; \frac{w_x(APX)}{m-2}.$$

Proof. Let $\delta_i = \xi_{i+1} - \xi_i$. Since $i(m-i-1) \geq m-2$ for $0 < i < m-1$, we have $w_x(APX_{i\bullet}) \leq \frac{k^2}{m^2}\delta_i \leq \frac{ik}{m}\frac{(m-i-1)k}{m}\delta_i \;\frac{1}{m-2}$ for $0 < i < m-1$. Since APX 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

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

so

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

□

Lemma 4. *For $i = 0$ and $i = m-1$ we have $w_x(APX_{i\bullet}) \;\leq\; \frac{w_x(APX)}{m-1}$.*

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

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

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 $\overline{APX}_{0\bullet}$ and since there are $(m-1)k/m$ points in $\overline{APX}_{0\bullet}$, we have $\xi_1 - x_j < \frac{m}{(m-1)k} w_x(p_j, \overline{APX}_{0\bullet})$ so

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

□

Combining the three lemmas, we get the claimed result and the proof of Theorem 2.

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

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

4.2 PTAS for General d

It is straightforward to generalize the PTAS to arbitrary, fixed dimension d.

Theorem 5. *For any fixed d, the problem of selecting a subset of minimum total L_1 distance for a set of points in $\Re^d$ allows a PTAS.*

Sketch of Proof. For $m = \Theta(1/\varepsilon)$, we subdivide the set of n points with $d(m+1)$ 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. As in the two-dimensional case, each cell picks the assigned number of points extremal in its 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 completely analogous to the details described for the two-dimensional case and omitted. □

5 Experiments

The work discussed so far is motivated by the allocation of a single job. In the following, we examine how well our algorithms work in the context of streams of jobs; in those scenarios, the set of free processors available for each job depends on previous allocations.

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; in essence, one algorithm (the *situation algorithm*) creates a sequence of instances on which the other algorithm (the *decision algorithm*) is evaluated. More precisely, the situation algorithm places a sequence of jobs; at each step of this iteration, this determines the free processors available for the next job. Pretending the next job has higher priority, we compute the achievable average sum of pairwise distances, as determined by the decision algorithm. This value is recorded to keep track of the decision algorithm's performance. However, this allocation is not carried out; instead, the situation algorithm is used for the actual placement, in turn creating a decision instance for the next iteration, etc. In this manner, each entry in Table 1 represents the average sum of pairwise distances for the decision algorithm, with processor availability determined by the situation algorithm.

Our simulation used the algorithms MC1x1, MM, MM+Inc, and HilbertBF. MM+Inc uses local improvement on the allocation of MM, replacing an allocated processor with an excluded processor that improves average pairwise distance until it reaches a local minimum. HilbertBF is the 1-dimensional strategy of Leung et al. [16] used on Cplant. The simulation used the LLNL Cray T3D trace from the Parallel Workloads Archive [9]. This trace has 21323 jobs run on a machine with 256 processors, treated as a 16×16 mesh in the simulation.

In each row, the algorithms are ranked in the order MM+Inc, MM, MC1x1, and HilbertBF. This is consistent with the worst-case performance bounds; MM is a 7/4-approximation, MC1x1 is a 4-approximation, and HilbertBF has approximation ratio $\Omega(N)$ on an $N \times N$ mesh.

Situation	Decision Algorithm			
Algorithm	MC1x1	MM	MM+Inc	HilbertBF
MC1x1	5256	5218	5207	5432
MM	5323	5288	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.

6 Conclusions

The algorithmic work described in this paper is one step toward developing algorithms for scheduling mesh-connected network-limited multiprocessors. We have given provably good algorithms to allocate a single job, a problem we conjecture to be NP-hard. The next step is to study the allocation of job sequences, a markedly different algorithmic challenge.

The difference between making a single allocation and a sequence of allocations is already illustrated by the diagonal entries in Table 1, where the free processors depend on the same algorithm's previous decisions. These give the ranking (from best to worst) HilbertBF, MC1x1, MM+Inc, and MM. The locally better decisions of MM+Inc seem to paint the algorithm into a corner over time. Figures 1, 2, and 3 help explain why. When starting on an empty grid, MC produces connected rectangular shapes. Locally, these shapes are slightly worse than the round shapes produced by MM, but rectangles have better packing properties because they avoid small patches of isolated grid nodes.

We confirmed this behavior over an entire trace using Procsimity [27, 29], which simulates messages moving through the network. We ran the NASA Ames iPSC/860 trace[-1] from the Parallel Workloads Archive [9], scaling down the number of processors for each job by a factor of 4. This made the trace run on a machine with 32 processors, allowing us to find the greedy placement that minimizes average pairwise distance at that step. For average job flow time, MC1x1 was best, followed by MM, and then greedy. We did not run MM+Inc in this simulation. HilbertBF was much worse than all three of the algorithms mentioned in part due to difficulties using it on a nonsquare mesh.

Based on these results and the work of Leung et al. [16], one of the first allocators developed and licensed for the Red Storm (Cray XT3/4) supercomputer uses a machine specific space-filling curve and a 1D bin-packing technique. We expect to have Red Storm implementations of a 3D version of MC1x1 and the greedy heuristic (called MM) analyzed in this paper.

Thus, the online problem in an iterated scenario is the most interesting open problem. We believe that a natural attack may be to consider online packing of rectangular shapes of given area. We plan to pursue this in future work.

Acknowledgments

We thank Jens Mache for informative discussions on processor allocation. We thank Moe Jette and Bill Nitzberg for providing the LLNL and NASA Ames iPSC/860 traces, respectively, to the Parallel Workloads Archive. Michael Bender was partially supported by Sandia and NSF Grants EIA-0112849, CCR-0208670, CCF-0621439/0621425, CCF-0540897/05414009, CCF-0634793/0632838, and CNS-0627645. David Bunde was partially supported by Sandia and NSF grant CCR 0093348. Sándor Fekete was partially supported by DFG grants FE 407/7 and FE 407/8. Henk Meijer was partially supported by NSERC. Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

1. A. Ahmadinia, C. Bobda, S. Fekete, J. Teich, and J. van der Veen. Optimal routing-conscious dynamic placement for reconfigurable computing. In *14th International Conference on Field-Programmable Logic and Application*, volume 3203, pages 847–851, 2004. Journal version to appear in *IEEE Transactions on Computing*.
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 parallel 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.
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. 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.
8. P.-J. Chuang and N.-F. Tzeng. An efficient submesh allocation strategy for mesh computer systems. In *Proc. Int. Conf. Dist. Comp. Systems*, pages 256–263, 1991.
9. D. Feitelson. The parallel workloads archive. `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`.
10. S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2004.
11. N. Guttmann-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`.
12. P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proc. 40th Ann. IEEE Symp. Found. Comp. Sci. (FOCS)*, pages 154–159, 1999.

13. 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.
14. 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.
15. S. Krumke, M. Marathe, H. Noltemeier, V. Radhakrishnan, S. Ravi, and D. Rosenkrantz. Compact location problems. *Th. Comp. Sci.*, 181:379–404, 1997.
16. 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.
17. V. Leung, C. Phillips, M. Bender, and D. Bunde. Algorithmic support for commodity-based parallel computing systems. Technical Report SAND2003-3702, Sandia National Laboratories, 2003.
18. 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.
19. 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.
20. J. Mache and V. Lo. Dispersal metrics for non-contiguous processor allocation. Technical Report CIS-TR-96-13, University of Oregon, 1996.
21. J. Mache and V. Lo. The effects of dispersal on message-passing contention in processor allocation strategies. In *Proc. Third Joint Conf. on Information Sciences, Sessions on Parallel and Distributed Processing*, volume 3, pages 223–226, 1997.
22. J. Mache, V. Lo, and K. Windisch. Minimizing message-passing contention in fragmentation-free processor allocation. In *Proc. 10th Intern. Conf. Parallel and Distributed Computing Systems*, pages 120–124, 1997.
23. S. Moore and L. Ni. The effects of network contention on processor allocation strategies. In *Proc. 10th Int. Par. Proc. Symp.*, pages 268–274, 1996.
24. S. Sahni and T. Gonzalez. p-complete approximation problems. *JACM*, 23(3):555–565, 1976.
25. Sandia National Laboratories. The Computational Plant Project. `http://www.cs.sandia.gov/cplant`.
26. 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.
27. University of Oregon Resource Allocation Group. Procsimity. `http://www.cs.uoregon.edu/research/DistributedComputing/ProcSimity.html`.
28. D. Weisser, N. Nystrom, C. Vizino, S. T. Brown, and J. Urbanic. Optimizing job placement on the Cray XT3. In *CUG 2006 Proceedings*, 2006.
29. 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`.
30. Y. Zhu. Efficient processor allocation strategies for mesh-connected parallel computers. *J. Parallel and Distributed Computing*, 16:328–337, 1992.