Communication-Aware Processor Allocation for Supercomputers*

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, University of Illinois,
Urbana, IL 61801, USA
`bunde@uiuc.edu`

[3] MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA 02139, USA
`edemaine@mit.edu`

[4] Dept. of Mathematical Optimization,
Braunschweig University of Technology,
38106 Braunschweig, Germany
`s.fekete@tu-bs.de`

[5] Discrete Algorithms & Math Department, Sandia National Laboratories,
Albuquerque, NM 87185-1110, USA
`{vjleung, caphill}@sandia.gov`

[6] Dept. of Computing and Information Science, Queen's University,
Kingston, Ontario, K7L 3N6, Canada
`henk@cs.queensu.ca`

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 2D 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 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.

[*] Extended Abstract. A full version is available as [5].

F. Dehne, A. López-Ortiz, and J.-R. Sack (Eds.): WADS 2005, LNCS 3608, pp. 169–181, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Motivation: Processor Allocation in Supercomputers. Our algorithmic work is motivated by a problem in the operation of supercomputers. The supercomputer for which we targeted our simulations and experiments is called Computational Plant or Cplant [7, 25], a commodity-based supercomputer developed at Sandia National Laboratories. In Cplant, 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 constraints forbid migration, preemption, or multitasking. 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 processor allocation affects throughput on a range of architectures [3,17,20,21,23]. Several papers suggest that minimizing the *average number of communication hops* is an appropriate metric for job placement [20, 21, 16]. Experiments with a communication test suite demonstrate that this metric correlates with a job's completion time [17].

Early processor-allocation algorithms allocate only convex sets of processors to each job [18, 9, 29, 6]. 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 government-audited system [15, 26].

Recent work [19,22,8,17,26] 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. The cost of the allocation is the sum of the shell numbers in which free processors occur; see Figure 1 reproduced from [22]. MC chooses the allocation with lowest cost. Since users of Cplant 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.

Until recently, 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 current allocator uses space-filling curves and 1D bin-packing techniques based upon work of Leung et al. [17].

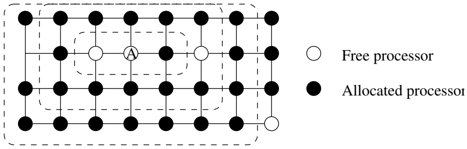

Fig. 1. Illustration of MC: Shells around processor A for a 3×1 request

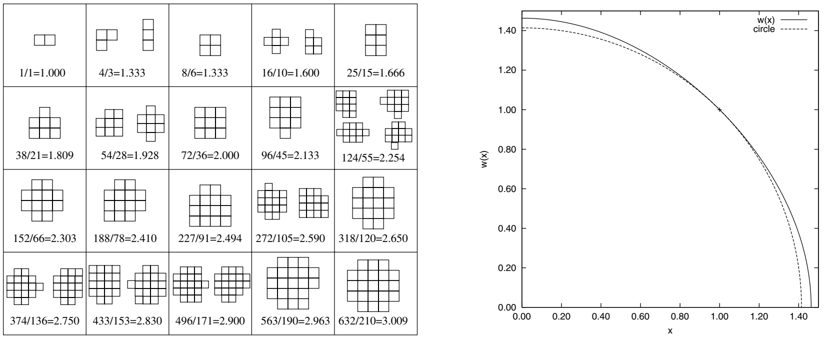

Fig. 2. (Left) Optimal unconstrained clusters for small values of k; numbers shown are the average L_1 distances, with truncated decimal values. (Right) Plot from [4] of a quarter of the optimal limiting boundary curve; the dotted line is a circle

Related Algorithmic Work. Krumke et al. [16] 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. [14] and Bender et al. [4] study the exact nature of this shape. Surprisingly, the resulting convex curve can only be described by a differential equation; the closed-form solution is unknown. The complexity of this special case remains open, but its mathematical difficulty emphasizes 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 [12] give a 2-approximation, Indyk [13] gives a PTAS for $k = 2$, and Bartel et al. [2] give an $O((1/\epsilon) \log^{1+\epsilon} n)$-approximation for general k.

Fekete and Meijer [11] 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 develop algorithms for minimizing the average L_1 distance between allocated processors in a mesh supercomputer. A greedy heuristic we analyze called MM and a 3D version of MC1x1 have been implemented on Cplant. In particular, we give the following results:

- We prove that MM is a $\frac{7}{4}$-approximation algorithm for $2D$ grids, reducing the previous best factor of 2 [16], and we show that this analysis is tight.
- We present a simple generalization to general d-dimensional space with fixed d and prove that the algorithm gives a $2-\frac{1}{2d}$-approximation algorithm, which is tight.
- We give an efficient polynomial-time approximation scheme (PTAS) for points in $\Re^d$ for constant d.
- Using simulations, we compare the allocation performance of our algorithm to that of other algorithms. As a byproduct, we get insight on how to place a stream of jobs in an online setting.

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 d-dimensional version of MC1x1 has approximation factor at most d times that of MM. We have an algorithm to solve the 2-dimensional case for $k = 3$ in time $O(n \log n)$.

2 Manhattan Median Algorithm for Two-Dimensional Point Sets

2.1 Median-Based Algorithms

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 $O(n^2)$ intersection points of the horizontal and vertical lines through the points in P. For each of these points p 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 2D meshes. (Note that Krumke et al. [16] call this algorithm Gen-Alg and show it is a 2-approximation in arbitrary metric spaces.)

2.2 Analysis of the Algorithm

For $S \subseteq P$, let $|S|$ 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.

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

Proof. For a class of examples establishing the lower bound, consider the situation shown in Figure 3. For any $\epsilon > 0$, it has clusters of $k/2$ points at $(0, 0)$ and $(1, 0)$. In addition, it has clusters of $k/8$ points at $(0, \pm(1-\epsilon))$, $(1, \pm(1-\epsilon))$, $(2-\epsilon, 0)$, and $(-1+\epsilon, 0)$. The best choices of median are $(0, 0)$ and $(1, 0)$, which yield a total distance of $7k^2(1 - \Theta(\epsilon))/16$. The optimal solution is the points at $(0, 0)$ and $(1, 0)$, which yield a total distance of $k^2/4$. □

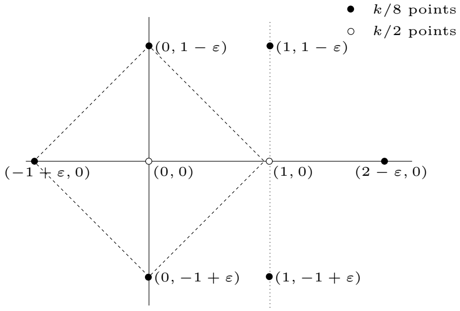

Fig. 3. A class of examples where MM yields a ratio of 7/4

Now we show that 7/4 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 $|OPT|$ is minimum. 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 distances 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 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 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 $|MM|$.

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

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

Proof. The set of arrangements of n points in the unit circle 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. □

For $k \leq 8n/11$ we show $|MM|$ is at most $7/4$ times larger than $|OPT|$.

Lemma 3. *For $k \leq 8n/11$ we have $\rho_k \leq 7/4$.*

Sketch of Proof. We assume that we have a point set P for which ρ_k is equal to $7/4$. We can assume without loss of generality that $P = MM \cup OPT$. If there is a point $p \in P$ that does not lie in a corner of C or on the origin, we look at all points that lie on the axis-parallel rectangle through p with corners on C. We move these points simultaneously, in such a way that they stay on an axis-parallel rectangle with corners on C. This move changes $|MM|$ by some small amount δ_a and $|OPT|$ by some amount δ_o. However if we move all points in the opposite direction $|MM|$ and $|OPT|$ change by $-\delta_a$ and $-\delta_o$ 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 in P. We can then repeat this argument until all points of P lie on a corner of C or on the origin.

It is now not too hard to show that the ratio MM/OPT is maximal if there are $k/2$ points at the origin, $k/2$ points in one corner of C and $k/8$ points at each of the other three corners. So we have $|MM|/|OPT| = 7/4$. Notice that n has to be at least $11k/8$ for this value to be obtained. □

For larger values of k it can be shown that ρ_k decreases, so we summarize:

Theorem 1. *MM is a $7/4$-approximation algorithm for minimizing the sum of pairwise L_1 distances in a 2D mesh.*

3 PTAS for Two Dimensions

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 [11] 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

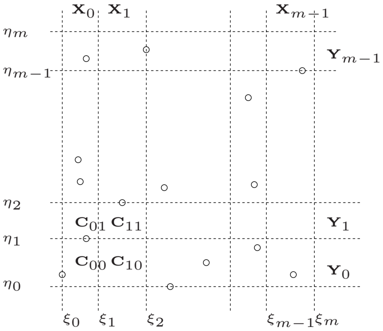

Fig. 4. Dividing the point set in horizontal and vertical strips

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 4 for the setup.

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 5.

It can be shown 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.

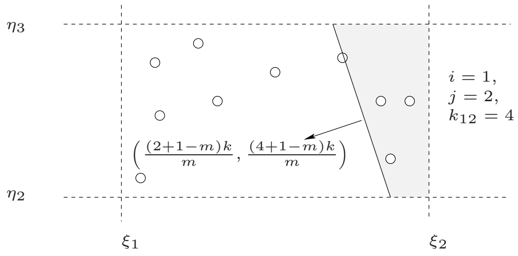

Fig. 5. Selecting points in cell C_{12}

The details are somewhat technical and are described in the full version of the paper [5]. We summarize:

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

4 Higher-Dimensional Spaces

Using the same techniques, we also generalize our results to higher dimensions. We start by describing the performance of *MM.*

4.1 $\left(2 - \frac{1}{2d}\right)$ -Approximation

As in two-dimensional space, *MM* enumerates over the $O(n^d)$ possible medians. For each median, it constructs a candidate solution of the k closest points.

Lemma 4. *MM is not better than a* $2 - 1/(2d)$ *approximation.*

Proof. We construct an example based on the cross-polytope in d dimensions, i.e., the d-dimensional L_1 unit ball. Let $\varepsilon > 0$. Denote the origin with O and the i^{th} 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 $i = 2, \ldots, d$. *MM* does best with O or $O + e_1$ as median, giving a total distance of $(k^2/4)\,(2 - 1/(2d))\,(1 + \Theta(\varepsilon))$. Optimal is the points at O and $O + e_1$, giving a total distance of $k^2/4$. □

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

Lemma 5. *Worst-case arrangements for MM can be assumed to have all points at positions* $(0, \ldots, 0)$ *and* $\pm e_i$*, where* e_i *is the ith 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 number of coordinate values is kept. 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, analogous to the proof of Lemma 3.

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

The restricted set of arrangements can be evaluated with symmetry to yield

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

4.2 PTAS for General Dimensions

Theorem 4. *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 analogous to the details described in the full version of the paper [5] 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 allocate streams of jobs; now the set of free processors available for each job depends on previous allocations.

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

Situation	Decision Algorithm			
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

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*, places each job. This determines the free processors available for the next job. Each allocation decision serves as an input to the other algorithm, the *decision algorithm*. 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. [17] used on Cplant. The simulation used the LLNL Cray T3D trace[1] from the Parallel Workloads Archive [10]. 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 7/2-approximation, and HilbertBF has an unbounded ratio[2].

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. 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 and 2 help explain why. When starting on an empty grid, MC

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

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

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, 28], which simulates messages moving through the network. We ran the NASA Ames iPSC/860 trace[1] from the Parallel Workloads Archive [10], 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.

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. Michael Bender was partially supported by Sandia and NSF Grants EIA-0112849 and CCR-0208670. 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. der Veen. Optimal routing-conscious dynamic placement for reconfigurable computing. In *International Conference on Field-Programmable Logic and its applications*, volume 3203 of *LNCS*, pages 847–851. Springer, 2004.
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.
4. C. M. Bender, M. A. Bender, E. D. Demaine, and S. P. Fekete. What is the optimal shape of a city? *J. Physics A: Mathematical and General*, 37:147–159, 2004.

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

5. M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, and C. A. Phillips. Communication-aware processor allocation for supercomputers. Technical Report cs.DS/0407058, Computing Research Repository, `http://arxiv.org/abs/cs.DS/0407058`, 2004.
6. 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.
7. 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.
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. Int. Conf. Dist. Comp. Systems*, pages 256–263, 1991.
10. D. Feitelson. The parallel workloads archive. `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`.
11. S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2004.
12. N. Guttmann-Beck and R. Hassin. Approximation algorithms for minimum sum p-clustering. *Disc. Appl. Math.*, 89:125–142, 1998.
13. 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.
14. 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.
15. 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.
16. S. Krumke, M. Marathe, H. Noltemeier, V. Radhakrishnan, S. Ravi, and D. Rosenkrantz. Compact location problems. *Th. Comp. Sci.*, 181:379–404, 1997.
17. 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.
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. 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.
29. Y. Zhu. Efficient processor allocation strategies for mesh-connected parallel computers. *J. Parallel and Distributed Computing*, 16:328–337, 1992.