Integer Point Sets Minimizing Average Pairwise ℓ_1 Distance: What is the Optimal Shape of a Town?

Erik D. Demaine* Sándor P. Fekete† Günter Rote‡ Nils Schweer† Daria Schymura‡ Mariano Zelke§

Abstract

An *n-town*, $n \in \mathbb{N}$, is a group of n buildings, each occupying a distinct position on a 2-dimensional integer grid. If we measure the distance between two buildings along the axis-parallel street grid, then an n-town has optimal shape if the sum of all pairwise Manhattan distances is minimized. This problem has been studied for *cities*, i.e., the limiting case of very large n. The optimal shape can be described by a differential equation. No closed-form solution is known. We show that optimal n-towns can be computed in time polynomial in n.

1 Introduction

Selecting an optimal set of locations is a fundamental problem not just in real estate, but also in areas of computer science. In grid computing, allocating a task requires selecting n processors from a given grid [2], and the average communication overhead corresponds to the average Manhattan distance between processors [7], so finding an optimal layout for n processors is precisely the problem of determining an optimal n-town. This problem is surprisingly tricky, with no known polynomial algorithm for computing an optimal n-town.

Motivated by the problem of storing records in a 2-dimensional array, Karp et al. [5] studied strategies that minimize average access time between successive queries; among other results, they described an optimal solution for the continuous version of our problem: What shape of area 1 minimizes the average Manhattan distance between two interior points? Independently, Bender et al. [1] solved this problem in the setting of a city, inspiring the subtitle of this paper.

Other related work by Krumke et al. [6] considers the discrete problem of selecting a subset of k points from a set of n points to minimize their average pairwise distance. They prove a 2-approximation for metric distances and prove hardness of approximation for arbitrary distances. Bender et al. [2] solve the geometric version of this problem, giving an efficient processor allocator for Cplant, a Sandia National Labs supercomputer, and a PTAS for minimizing the average Manhattan distance. For the reverse problem of *maximizing* the average Manhattan distance, see [3]. Fekete et al. [4] consider the city-center problem: For a given city, find a point that minimizes the average Manhattan distance.

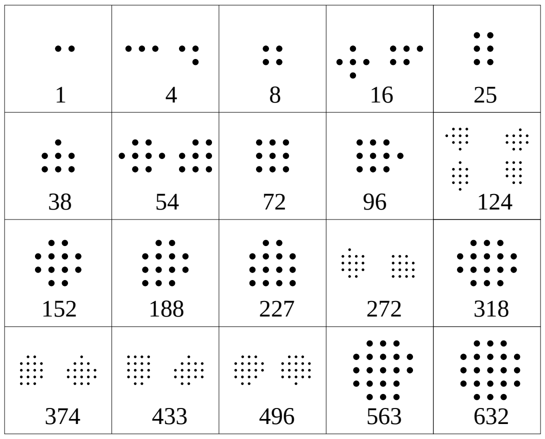

Figure 1: Optimal towns for $n = 2, \dots, 21$. All optimal solutions are shown, up to symmetries; the numbers indicate the total distance between all pairs of points.

We solve the n-town problem with an $O(n^{7.5})$-time dynamic-programming algorithm. We also present computational results and discuss discretization effects.

2 Preliminaries

We want to find a set of n distinct points from the integer grid $\mathbb{Z} \times \mathbb{Z}$ such that the sum of all pairwise Manhattan distances is minimized. A set $S \subset \mathbb{Z} \times \mathbb{Z}$ of cardinality n is an *n-town*. An n-town S is *optimal* if

$$\sum_{\{s,t\} \in S \times S} \|s - t\|_1$$

is minimum. We call this sum the *cost* of S and denote it by $c(S)$. We define the *x-cost* $c_x(S)$ as $\sum_{\{s,t\} \in S \times S} |s_x - t_x|$, where s_x is the x coordinate of s; *y-cost* $c_y(S)$ is the sum of all y distances, and $c(S) = c_x(S) + c_y(S)$. For two disjoint sets S and S', we define $c(S, S') = \sum_{\{s,s'\} \in S \times S'} \|s - s'\|_1$.

*MIT Computer Science and Artificial Intelligence Lab, edemaine@mit.edu

†Algorithms Group, Braunschweig University of Technology, {s.fekete,n.schweer}@tu-bs.de

‡Institut für Informatik, Freie Universität Berlin, Germany, {rote,schymura}@inf.fu-berlin.de

§Institut für Informatik, Goethe-Universität, Frankfurt/M. zelke@em.uni-frankfurt.de

Without loss of generality, we consider optimal towns around the origin and use the following notation: For an n-town S, the set $C_i = \{(i, x) \in S : x \in \mathbb{Z}\}$ is the ith *column* of S and the set $R_i = \{(x, i) \in S : x \in \mathbb{Z}\}$ is the ith *row* of S.

A town S is *convex* if the set of grid points in the convex hull of S equals S. The proofs of the following two lemmas are omitted due to lack of space.

Lemma 1 *An optimal n-town is convex.*

Lemma 2 *In every optimal n-town S, the centers of all rows of odd length lie on a common vertical line V_o. The centers of all rows of even length lie on a common line V_e that has distance $\frac{1}{2}$ from V_o. A corresponding statement holds for the centers of odd and even columns that lie on horizontal lines H_o and H_e of distance $\frac{1}{2}$. Moreover, without changing its cost, we can place S such that H_o and V_o are mapped onto the x axis and y axis, respectively, and H_e and V_e lie in the negative halfplanes.*

Fig. 2 shows the lines from the previous lemma. In the following we assume the preceding symmetry property. For an n-town S, let the *width* of S be $w(S) = \max_{i\in\mathbb{Z}} |R_i|$ and the *height* of S be $h(S) = \max_{i\in\mathbb{Z}} |C_i|$.

Lemma 3 *For every optimal n-town S,*

$$w(S) > h(S)/2 - 2 \text{ and } h(S) > w(S)/2 - 2.$$

Proof. Let S be an n-town, $w = w(S)$, and $h = h(S)$. Let $t = (0, l)$ be the topmost and $(k, 0)$ be the rightmost point of S. Let $r = (k+1, 0)$. We show that $c((S \setminus t) \cup r) < c(S)$ if $w \le h/2 - 2$ and, thus, S is not optimal. We estimate the change in cost $c(r, S) - c(t, S) - |k + l + 1|$ column by column. Comparing the costs in S and $(S \setminus t) \cup r$ shows that $c(r, C_i) - c(t, C_i) \le c(r, C_0) - c(t, C_0)$ for all i. We reduce the total cost by reducing it for C_0:

$$\begin{aligned} & c(r, C_0) - c(t, C_0) \\ = & \sum_{i=1}^{\lfloor \frac{h-1}{2} \rfloor} i + \sum_{i=1}^{\lceil \frac{h-1}{2} \rceil} i + h(k+1) - \sum_{i=1}^{h-1} i \\ \le & \frac{h}{2}(1 - \frac{h}{2}) + h(k+1) \end{aligned}$$

We know that $2k \le w - 1$ because of Lemma 2. The above term is nonpositive if $w \le h/2 - 2$. Hence, by replacing t with r we gain at least $|l + k + 1| \ge 1$. $\square$

Lemma 4 *For every optimal n-town we have*

$$\max\{\, w(S), h(S) \,\} = O(\sqrt{n}).$$

Proof. Let $w = w(S)$ and $h = h(S)$. We know from Lemma 3 that $w > h/2 - 2$. Let H be the set of all grid points in the convex hull of the topmost, rightmost, bottommost, and leftmost point of S. Because of Lemma 1, all points in H belong to S. Because H consists of at most n points, we have

$$n \ge |H| \ge wh/2 - h > (h/2 - 2)h/2 - h \; . \quad \square$$

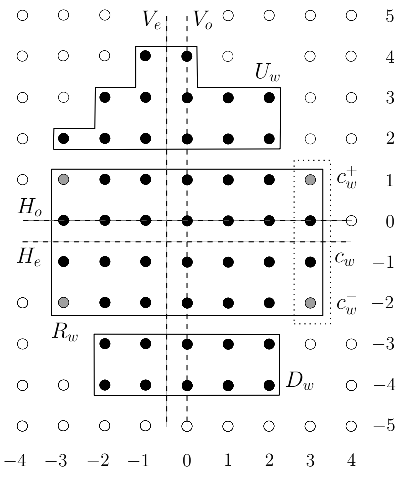

Figure 2: The lines V_o, V_e, H_o, and H_e from Lemma 2. The rectangle R_w and the set of points above and below it with cardinality U_w and D_w, respectively. The gray points are the corner points of R_w. In this example, the height c_w of column w is set to $c = 4$.

3 Computing Optimal Solutions

We denote by $c_i = |C_i|$ the number of selected points in column i and by c_i^+ and c_i^- the row index of the topmost and bottommost selected point in C_i, respectively. Note that $c_i = c_i^+ - c_i^- + 1$; see Fig. 2.

Lemma 5 *Let S be an optimal n-town containing the points (i, c_i^+) and (i, c_i^-). Then all points inside the rectangle $[-i, i] \times [c_i^-, c_i^+]$ belong to S. The same holds for the points $(-i, c_{-i}^+)$ and $(-i, c_{-i}^-)$ and the rectangle $[-i, i-1] \times [c_{-i}^-, c_{-i}^+]$.*

Proof. If (i, c_i^+) and (i, c_i^-) are contained in S then, by Lemma 2, $(-i, c_i^+)$ and $(-i, c_i^-)$ belong to S as well. By Lemma 1 all points inside the convex hull of these four points are contained in S. The same arguments hold for the second rectangle. $\square$

Now we describe the dynamic program. It starts with the initial empty grid and chooses new columns alternating from the set of columns with nonnegative and with negative column index, i.e., it chooses the columns according to the index sequence $0, -1, 1, -2, 2, \ldots$. Let $w \ge 0$ be the index of the currently chosen column and fix c_w to a value c. We describe the dynamic program for columns with nonnegative index; columns with negative index are handled similarly.

We know from Lemma 5 that, in every optimal solution, every point inside the rectangle $R_w = [-w, w] \times [c_w^-, c_w^+]$ is selected. We define $\text{cost}(w, c, \Delta_w^{\text{UR}}, \Delta_w^{\text{DR}}, \Delta_w^{\text{UL}}, \Delta_w^{\text{DL}}, U_w, D_w)$ as the minimum cost of a town with columns $-w, \ldots, w$ of height $c_i \ge c$ for $-w \le i \le w$ and $c_w = c$ where U_w points lie above the rectangle R_w, having a total distance

Δ_w^{UL} and Δ_w^{UR} to the upper-left and upper-right corner of R_w, respectively, and D_w points lie below R_w, having a total distance Δ_w^{DL} and Δ_w^{DR} to the lower-left and lower-right corner of R_w. For a given n, we are looking for the n-town with minimum cost where $(2w+1)c + U_w + D_w = n$. Next we show that $\mathrm{cost}(w, c, \Delta_w^{\mathrm{UR}}, \Delta_w^{\mathrm{DR}}, \Delta_w^{\mathrm{UL}}, \Delta_w^{\mathrm{DL}}, U_w, D_w)$ can be computed recursively.

Consider the current column w with $c_w = c$. The cost from all points in this column to all points above R_w, in R_w, and below R_w can be expressed as

$$\begin{array}{rl} & \sum_{k=c^-}^{c^+} (\Delta_w^{\mathrm{UR}} + (c^+ - k) \cdot |U_w|) \\ + & \sum_{i=-w}^{w} \sum_{j=c^-}^{c^+} \sum_{k=c^-}^{c^+} [(w-i) + |k-j|] \\ + & \sum_{k=c^-}^{c^+} (\Delta_w^{\mathrm{DR}} + (k - c^-) \cdot |D_w|)\ . \end{array}$$

We can transform this into

$$\begin{array}{rl} & c \cdot (\Delta_w^{\mathrm{UR}} + \Delta_w^{\mathrm{DR}} + U_w \cdot c^+ - D_w \cdot c^-) \\ + & c^- \cdot (D_w - U_w) \cdot ((c+1) \mod 2) \\ + & \left(c^2 w + \frac{c^3-c}{3}\right) \cdot (2w+1) - \frac{c^3-c}{6} \end{array} \qquad (1)$$

which, obviously, depends only on the parameters w, c, Δ_w^{UR}, Δ_w^{DR}, U_w, and D_w (the two parameters Δ_w^{UL}, Δ_w^{DL} are needed if we consider a column with negative index). We denote (1) by $\mathrm{dist}(w, c, \Delta_w^{\mathrm{UR}}, \Delta_w^{\mathrm{DR}}, \Delta_w^{\mathrm{UL}}, \Delta_w^{\mathrm{DL}}, U_w, D_w)$ and state the recursion for the cost function:

$$\begin{array}{l} \mathrm{cost}(w, c, \Delta_w^{\mathrm{UR}}, \ldots, \Delta_w^{\mathrm{DL}}, U_w, D_w) \\ = \mathrm{dist}(w, c, \Delta_w^{\mathrm{UR}}, \ldots, \Delta_w^{\mathrm{DL}}, U_w, D_w) \qquad (2) \\ \quad + \min_{c_{-w} \geq c} \{\mathrm{cost}(-w, c_{-w}, \Delta_{-w}^{\mathrm{UR}}, \ldots, \Delta_{-w}^{\mathrm{DL}}, U_{-w}, D_{-w})\} \end{array}$$

Because of Lemma 5 it suffices to consider only previous solutions with $c_{-w} \geq c$. In the step before, we considered the rectangle $R_{-w} = [-w, w-1] \times [c_{-w}^+, c_{-w}^-]$. Hence, the parameters with index $-w$ can be computed from the parameters with index w as follows:

$$\begin{array}{rcl} U_{-w} & = & U_w - 2w \cdot (c_{-w}^+ - c^+), \\ D_{-w} & = & D_w - 2w \cdot (c^- - c_{-w}^-), \\ \Delta_{-w}^{\mathrm{UR}} & = & \Delta_w^{\mathrm{UR}} - \displaystyle\sum_{i=-w}^{w} \sum_{j=c^+ +1}^{c_{-w}^+} \left[(w-i) + (j - c^+)\right] \\ & & - [U_w - U_{-w}] \cdot (c_{-w}^+ - c^+ + 1), \\ \Delta_{-w}^{\mathrm{DR}} & = & \Delta_w^{\mathrm{DR}} - \displaystyle\sum_{i=-w}^{w} \sum_{j=c^- -1}^{c_{-w}^-} \left[(w-i) + (c^- - j)\right] \\ & & - [D_w - D_{-w}] \cdot (c^- - c_{-w}^- + 1). \end{array}$$

The parameters $\Delta_{-w}^{\mathrm{UL}}$ and $\Delta_{-w}^{\mathrm{DL}}$ can be computed analogously.

The cost function is initialized as follows:

$$\mathrm{cost}(0, c, 0, 0, 0, 0, 0, 0) = \begin{cases} \frac{c^3-c}{6} & 0 \leq c \leq 2\sqrt{n+4} + 4, \\ \infty & \text{otherwise.} \end{cases}$$

The bound on c is shown in the proof of Lemma 4.

Theorem 6 *An optimal n-town can be computed by dynamic programming in $O(n^{15/2})$ time.*

Proof. We have to fill an eight-dimensional array $\mathrm{cost}(w, c, \Delta^{\mathrm{UR}}, \Delta^{\mathrm{DR}}, \Delta^{\mathrm{UL}}, \Delta^{\mathrm{DL}}, U, D)$. Let $C_{\max}$ denote the maximum number of occupied rows and columns in an optimum solution. By Lemma 4, we know that $C_{\max} = O(\sqrt{n})$.

The indices w and c range over an interval of size $C_{\max} = O(\sqrt{n})$. Let us consider a solution for some fixed w and c. The parameters U and D range between 0 and n. However, we can restrict the difference between U and D that we have to consider: If we reflect the rectangle $R = [-w, w] \times [c^-, c^+]$ about its horizontal symmetry axis, the U points above R and the D points below R will not match exactly, but in each column, they differ by at most one point, by Lemma 2. It follows that $|U - D| \leq C_{\max} = O(\sqrt{n})$. (If the difference is larger, such a solution can never lead to an optimal n-town, and hence we need not explore those choices.) In total, we have to consider only $O(n \cdot \sqrt{n}) = O(n^{3/2})$ pairs (U, D).

The same argument helps to reduce the number of quadruples $(\Delta^{\mathrm{UL}}, \Delta^{\mathrm{UR}}, \Delta^{\mathrm{DL}}, \Delta^{\mathrm{DR}})$. Each Δ-variable can range between 0 and $n \cdot 2C_{\max} = O(n^{3/2})$. However, when reflecting around the horizontal symmetry axis of R, each of the at most $D_{\max}$ differing points contributes at most $2C_{\max} = O(\sqrt{n})$ to the difference between the distance sums Δ^{UL} and Δ^{DL}. Thus we have $|\Delta^{\mathrm{UL}} - \Delta^{\mathrm{DL}}| \leq C_{\max} \cdot 2C_{\max} = O(n)$, and similarly, $|\Delta^{\mathrm{UR}} - \Delta^{\mathrm{DR}}| = O(n)$

By a similar argument, reflecting about the vertical symmetry axis of R, we conclude that $|\Delta^{\mathrm{UL}} - \Delta^{\mathrm{UR}}| = O(n)$ and $|\Delta^{\mathrm{DL}} - \Delta^{\mathrm{DR}}| = O(n)$. In summary, the total number of quadruples $(\Delta^{\mathrm{UL}}, \Delta^{\mathrm{UR}}, \Delta^{\mathrm{DL}}, \Delta^{\mathrm{DR}})$ that the algorithm has to consider is $O(n^{3/2}) \cdot O(n) \cdot O(n) \cdot O(n) = O(n^{9/2})$. In total, the algorithm processes $O(\sqrt{n}) \cdot O(\sqrt{n}) \cdot O(n^{3/2}) \cdot O(n^{9/2}) = O(n^7)$ octuples. For each octuple, the recursion (2) has to consider at most $C_{\max} = O(\sqrt{n})$ values c_{-w}, for a total running time of $O(n^{15/2})$. □

4 n-Towns, Cities, and n-Block Cities

When considering n-towns for large values of n, we converge towards the continuous weight distributions of *cities.* However, the arrangements of buildings in many cities are discretized in a different sense: An *n-block city* is the connected union of n axis-aligned unit

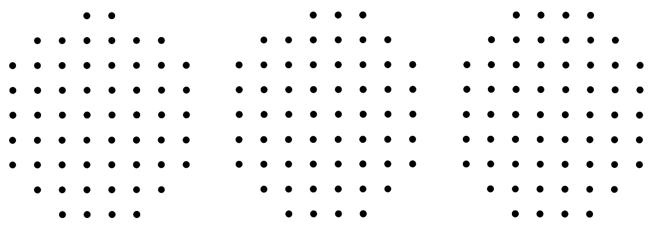

Figure 3: Optimal n-towns for $n = 58, 59, 60$; the values Φ and Ψ refer to n-town and corresponding n-block city.

squares ("city blocks"), i.e., a polyomino with area n; see Fig. 4. In the following, we discuss the exact relationship between n-towns and n-block cities.

Scaling a shape S by a factor of d increases the total cost by a factor of d^5, i.e., by a factor of $A^{2.5}$ for an area of A. This motivated Bender et al. [1] to use the expression $D(S) := \frac{c(S)}{A(S)^{2.5}}$ as a scale-independent measure for the quality of the shape of a city, e.g., any square Q gets the same value

$$D(Q) \quad = \quad \frac{2}{a^5}\int_0^a\int_0^a |x_1 - x_2|\, dx_2 dx_1 = \frac{2}{3}\ .$$

A circle C yields $D(C) = \frac{512}{45\pi^{2.5}} \approx 0.6504$ and the optimal shape achieves a value of $\psi = 0.650\,245\,952\,951\ldots$

When comparing n-towns and n-block cities, we need to account for the discretization effect. E.g., a 1-town has an average distance of 0, as all the weight is concentrated in a single point, while a 1-block city has a value of 2/3, just like any other square. We can compute the resulting cost for n-block cities by adding an appropriate adjustment term $\Delta(S)$ to the corresponding $c(S)$ for n-towns. Using the notation of Section 3, a straightforward computation shows that this term is $\Delta(S) = \frac{1}{3}(\sum_i c_i^2 + \sum_j r_j^2)$. As we noted above, any n-town is enclosed in a bounding box of size $\Theta(\sqrt{n}) \times \Theta(\sqrt{n})$; hence, $\Delta(S) \in \Theta(n^{1.5})$. The dynamic programming formulation of Section 3 can be modified to compute optimal n-block cities, but we have not done this.

When considering the continuous weight distributions of n-block cities, we have to account twice for each pair of discrete block centers; hence, the appropriate measure for the quality of an n-town is $\Phi(S) = \frac{2c(S)}{n^{2.5}}$. The measure for the corresponding n-block city is $\Psi(S) = \frac{2c(S)+\Delta(S)}{n^{2.5}}$. Thus, the relative difference is $\Theta(\frac{1}{n})$.

In Figure 4, we show the corresponding values for the small examples from Figure 1; Figure 3 shows the values for larger n. (Going from n-towns to n-block cities breaks some ties; we show only the Ψ-optimal solutions among the Φ-optimal solutions.) This shows how $\Phi(S)$ and $\Psi(S)$ converge from below and above towards the optimal city value of about $0.650\,245\,952\,951$.

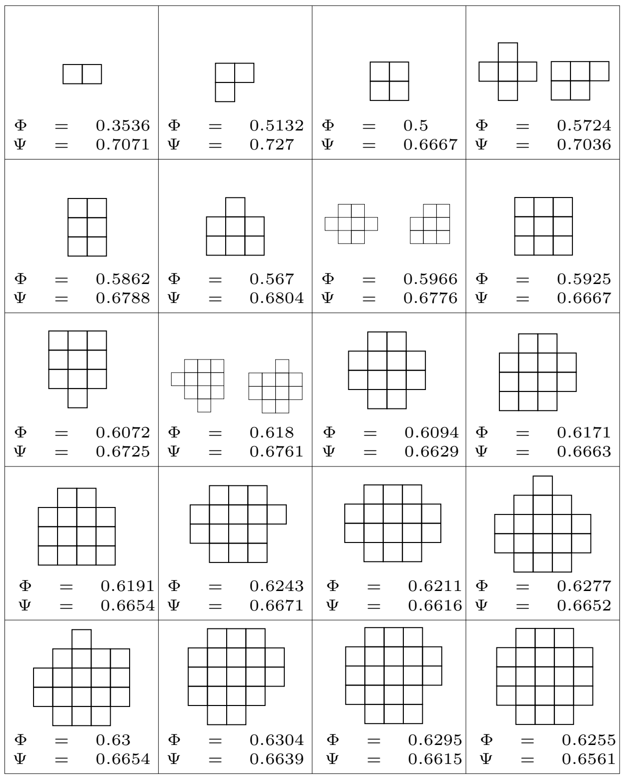
$\Phi = 0.3536$ $\Psi = 0.7071$
$\Phi = 0.5132$ $\Psi = 0.727$
$\Phi = 0.5$ $\Psi = 0.6667$
$\Phi = 0.5724$ $\Psi = 0.7036$
$\Phi = 0.5862$ $\Psi = 0.6788$
$\Phi = 0.567$ $\Psi = 0.6804$
$\Phi = 0.5966$ $\Psi = 0.6776$
$\Phi = 0.5925$ $\Psi = 0.6667$
$\Phi = 0.6072$ $\Psi = 0.6725$
$\Phi = 0.618$ $\Psi = 0.6761$
$\Phi = 0.6094$ $\Psi = 0.6629$
$\Phi = 0.6171$ $\Psi = 0.6663$
$\Phi = 0.6191$ $\Psi = 0.6654$
$\Phi = 0.6243$ $\Psi = 0.6671$
$\Phi = 0.6211$ $\Psi = 0.6616$
$\Phi = 0.6277$ $\Psi = 0.6652$
$\Phi = 0.63$ $\Psi = 0.6654$
$\Phi = 0.6304$ $\Psi = 0.6639$
$\Phi = 0.6295$ $\Psi = 0.6615$
$\Phi = 0.6255$ $\Psi = 0.6561$

Figure 4: Values of Φ and Ψ for small Φ-optimal towns.

References

[1] C. M. Bender, M. A. Bender, E. D. Demaine, and S. P. Fekete. What is the optimal shape of a city? *Journal of Physics A: Mathematical and General*, 37:147–159, 2004.

[2] 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: Finding point sets of small average distance. *Algorithmica*, 50(2):279–298, 2008.

[3] S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2003.

[4] S. P. Fekete, J. S. B. Mitchell, and K. Weinbrecht. On the continuous Weber and k-median problems. In *Proceedings of the 16th Annual Symposium on Computational Geometry (SoCG)*, pages 70–79, 2000.

[5] R. M. Karp, A. C. McKellar, and C. K. Wong. Near-optimal solutions to a 2-dimensional placement problem. *SIAM J. Comput.*, 4(3):271–286, 1975.

[6] S. Krumke, M. Marathe, H. Noltemeier, V. Radhakrishnan, S. Ravi, and D. Rosenkrantz. Compact location problems. *Theoret. Comp. Sci.*, 181(2):379–404, 1997.

[7] 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.