The Kissing Problem: How to End a Gathering When Everyone Kisses Everyone Else Goodbye

Michael A. Bender[1 2 ⋆], Ritwik Bose[1], Rezaul Chowdhury[1], and Samuel McCauley[1]

[1] Department of Computer Science, Stony Brook University, NY 11794-4400, USA
[2] Tokutek, Inc.

Abstract. This paper introduces the kissing problem: given a rectangular room with n people in it, what is the most efficient way for each pair of people to kiss each other goodbye? The room is viewed as a set of pixels that form a subset of the integer grid. At most one person can stand on a pixel at once, and people move horizontally or vertically. In order to move into a pixel in time step t, the pixel must be empty in time step $t - 1$.
The paper gives one algorithm for kissing everyone goodbye.
(1) This algorithm is a 4 + o(1)-approximation algorithm in a crowded room (e.g., only one unoccupied pixel).
(2) It is a 10 + o(1)-approximation algorithm for kissing in a comfortable room (e.g., at most half the pixels are empty).
(3) It is a 25+o(1)-approximation for kissing in a sparse room.

1 Introduction

Leaving a meeting (or party or other gathering) involves different rituals in different cultures. In the U.S., one often takes one's leave via a multicast protocol ("Goodbye everyone. I had a great time tonight. Happy Haiku Day."[3]). In many other parts of the world (in our experience, Latin America and France) it is polite to takes one's leave via a linear number of unicast protocols—kisses on the cheek or other handshake protocols (e.g., handshakes). When a large number of people quit a gathering simultaneously, it may be difficult for all to say goodbye efficiently, because of the complicated routing so that each pair of people can meet. This paper gives algorithms for scheduling and routing the individual goodbyes.

The goodbyes take place on a set of pixels that comprise an $m \times n$ grid, the room in which the shindig took place. Each pixel may be *unoccupied* or may be occupied by exactly one person. (This model does not allow for parties in which people may stand on each other's heads). We have a set $P = \{1 \ldots p\}$ of people. At each unit of time, any subset $S \subseteq P$ of people can move to adjacent unoccupied pixels. A *kiss* is transacted between i and j when they occupy adjacent squares. Note that multiple kisses may occur simultaneously in this model, although we do not suggest that you try this in practice, no matter how quickly you wish to leave a party.

⋆ This research was supported in part by NSF Grants CCF 0937822, CCF 1114809, CCF 0634793, and DOE Grant DE-FG02-08ER25853.

[3] April seventeenth. Lip service to Haiku Day. Just an FYI.

This kissing problem is reminiscent of several other problems in swarm or multi-agent robotics, optimization, and box-moving.

For example, the kissing problem has similarities to the traveling salesman problem (TSP) on a rectilinear grid [15, 33]: to leave the gathering efficiently, you find a short tour among all $p - 1$ others (the "cities"). However, there are differences: (1) In the kissing problem, unlike TSP, cities can move to you. (2) In the kissing problem, people serve as salesman for themselves and as cities for each other. (3) People (unlike salesman) take up space—only one person can stand on the same pixel at any time. (4) In the kissing problem there is a notion of neighborhoods (reminiscent of TSP with neighborhoods [3, 17]) because to say goodbye to someone, you move to a neighboring pixel and kiss. You rarely say goodbye to someone by stepping on him. To summarize, the problem has a whiff of TSP flavor, but remains otherwise distinct.

The kissing problem is also related to the 15-puzzle [32, 41] and other sliding block problems [22, 27]. Sliding-block puzzles generalize the 15-puzzle by allowing unmovable blocks, and blocks that are larger than 1×1. Generally the goal of a sliding-block puzzle is to move a block to a single location (the "warehouseman's problem" [21]), to find out if a single block is movable [18, 19], or somehow reorder all blocks [20]. In contrast, in the kissing problem, the objective is for all blocks to touch each other. In this paper, we only consider gatherings that take place in rectangular rooms without obstacles (e.g., it's ok to stand on the coffee table).

Other examples of multi-agent problems in robotics include pattern formation [6, 11, 14, 38], dispersion [25, 39], exploration and mapping [7, 24, 29, 34, 36, 37, 42], rendezvous [1, 2, 9, 12, 14, 28], and motion planning [4–6, 8, 13, 16, 23, 26], Ref. [40], in particular, considers what happens when an individual robot can speak only to its neighbors and there is no secure communication so that each robot must tell each other robot its message individually. Thus, if the message needs to be conveyed pairwise among all robots, then this is an instance of the kissing problem.

Results. This paper presents an approximation algorithm for the kissing problem with the following guarantees:

- Our kissing algorithm gives a $4 + o(1)$-approximation to the kissing problem in a *crowded room*, in which all pixels in the room are occupied except for one. In particular, it gives a $1+o(1)$-approximation for a $2 \times n$ grid, and achieves optimality for 2×3 and 2×4 grids.
- The kissing algorithm gives a $10 + o(1)$-approximation in a *comfortable room*, in which the number of unoccupied pixels is no more than the number of people.
- The kissing algorithm gives a $25 + o(1)$-approximation in a *sparse room*, in which people may be arbitrarily spread out. The approximation ratio applies when there are people abutting the furthest pair of walls, although the algorithm works in any case.
- We ran experiments to determine optimal solutions for some small cases of the kissing problem using IDA* state space search [10]. These results demonstrate that our algorithm is optimal for 2×3 and 2×4 grids in the crowded room case.

Map. In Sections 2, 3, and 4 we analyze the kissing problem in the crowded-room case, the comfortable-room case, and the sparse case respectively.

2 The Kissing Problem in a Crowded Room

A *crowded room* has only one unoccupied pixel, so only one person can move at a time. In this section we present an algorithm for $2 \times n$ grids that performs within a $1 + o(1)$ factor of optimal. Then we generalize the algorithm to become a $4+o(1)$-approximation algorithm for arbitrary $n \times m$ grids.

Our algorithm is based on a circuit that each person follows around the grid. Formally, a *cycle* is a set of moves where each non-wallflower person moves forward once along the circuit. For the $2 \times n$ grid, we can construct the circuit by keeping the two rightmost people still and cycling everyone else ("cycling with wallflowers") or by cycling all the people; see Figure 1. When there are no wallflowers, people only need to cycle through half the grid to kiss everyone, whereas with wallflowers, some people must cycle through the entire grid, yielding the additional factor of 2. However, when there are wallflowers, the lower-order terms are better because the cycle is shorter, so wallflowers lead to better solutions for small n.

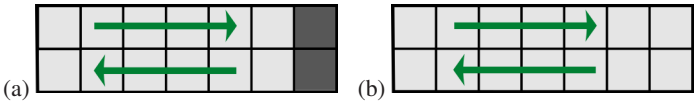

Fig. 1. Two methods for solving the kissing problem on a 2×7 grid. Arrows indicate the direction of the two routes. (a) The two rightmost people remain stationary. (b) Everyone participates in the cycle.

Lemma 1. *On a crowded $2 \times n$ grid, cycling both with and without wallflowers enables all people to kiss each other. This requires n cycles without wallflowers and $2n - 2$ cycles with wallflowers.*

Proof. For the case with no wallflowers (Figure 1b), number the pixels clockwise from 1 to $2n$ starting with the bottom-right pixel, continuing to the left across the bottom row, and then right to left across the top row. If there are wallflowers (Figure 1a), they are excluded from the number, and we only number the remaining pixels—in other words, the numbering proceeds as it would in the $2 \times (n - 1)$ case.

We define the *outgoing route* as the pixels in order from 1 to n, and the *incoming route* as the pixels in order from $n + 1$ to $2n$. The route is used to keep track of the two halves of the cycle that a person can travel. The routes are shown as arrows in Figure 1.

Consider only the kisses that happen when two squares are in different routes. If a kiss happens between a pixel at i and a pixel at j, one is above the other, so we have $i + j = 2n + 1$.

Since there is only one unoccupied pixel, a cycle requires one time step per person, for a total of $2n - 1$. Let p_i denote the person who stands at pixel i when the algorithm begins. After t cycles, p_i stands on pixel $p_i(t) \equiv i + t \pmod{2n}$. Note that during a cycle, there will be intermediate positions where some people have moved forward, but others have not yet. During a cycle, if p_i has not yet moved, $p_i(t) \equiv i + t - 1 \pmod{2n}$.

People p_i and p_j kiss when they are in the same column. A kiss at the end of cycle t occurs if $p_i(t) + p_j(t) \equiv 1 \pmod{2n}$ which means that $i + j + 2t \equiv 1 \pmod{2n}$. There may also be kisses during the cycle. Assume without loss of generality that $i < j$. Consider an intermediate point in the cycle when p_j has moved but p_i has not. People p_i and p_j kiss when $i + j + 2t - 1 \equiv 1 \pmod{2n}$.

Thus, two people kiss after cycle t if $2t \equiv 1 - i - j \pmod{2n}$ and at some point during cycle t if $2t - 1 \equiv 1 - i - j \pmod{2n}$. Once t has reached n, every pair of people has kissed.

The analysis is similar for the wallflower case. The cycles take place on a $2 \times n - 1$ subset of the grid, meaning that after $n - 1$ cycles non-wallflowers have kissed. The wallflowers have already kissed each other, so now we need to ensure that they have kissed everyone else. Person p_i has kissed both wallflowers, once it has passed through pixels 1 and $2(n - 1)$. Therefore, everyone has kissed the wallflowers after $2(n - 1)$ cycles. □

Lemma 2. *A lower bound on the kissing problem on a crowded* $2 \times n$ *grid is* $2n^2 - 6n + 4$.

Proof. We determine the number of kisses that need to be completed over the course of the algorithm, then show an upper bound on the number of kisses attainable per move, leading to a lower bound on the number of moves necessary for all to kiss.

Kisses that are made in the initial state do not need to be made during the algorithm. Initially, there are $3n - 4$ or $3n - 5$ kisses when the unoccupied pixel is in a corner or non-corner, respectively.

We next show that after the initial kisses, at most two kisses are made per turn; that is, # kisses ≤ 2(# moves). When p_i moves to an adjacent empty square, he has at most three new neighbors (because this is the $2 \times n$ case). But one neighbor must be empty, the pixel vacated by p_i, leaving only two people for p_i to kiss.

This bound can be improved to show that only one kiss can be made per turn after the first, when two kisses can be made. If I move into an empty pixel, that pixel must have been vacated by someone else. But this person is my neighbor again after I move into the pixel, and we have already kissed. More formally, consider the turn $t > 1$, where s is the unoccupied pixel. Let p_i move into s at time step $t + 1$. Pixel s must have been occupied by some person p_j at time $t - 1$. We know that p_i is adjacent to s at turn $t - 1$, so p_i and p_j have already kissed. Furthermore, p_j is a neighbor of s as it only moved once. Therefore, when p_i moves into s, one of its neighbors must be unoccupied, and one must be a person he has already kissed. Since each pixel has at most three neighbors, only one new kiss can be made per time step after the first, when two kisses can be made. Therefore,

$$(\text{\# kisses}) \leq (\text{\# moves made}) + 1.$$

We can take the number of kisses necessary, subtract the number of initial kisses, and combine with the bound on the number of moves t to get

$$\binom{2n-1}{2} - (3n - 4) \leq t + 1.$$

Solving for t,

$$t \geq 2n^2 - 6n + 4.$$

□

Theorem 1. *For a $2 \times n$ grid in the crowded room, cycling without wallflowers takes $2n(n-1)$ time. Cycling with wallflowers takes $(2n-3)(2n-4)$ time. Cycling without wallflowers yields a $1 + o(1)$ approximation to optimal.*

Proof. Without wallflowers, by Lemma 1, the algorithm moves $2n-1$ people per cycle, and continues for n cycles, for a running time of $2n^2 - n$. Similarly, with wallflowers, the algorithm continues for $(2n-4)$ cycles, each of which moves $(2n-3)$ people exactly once, for a total running time of $(2n-3)(2n-4)$.

We divide the running time by the lower bound to get the approximation

$$\frac{2n^2 - n}{2n^2 - 6n + 4} = 1 + o(1).$$

□

Corollary 1. *The cycle algorithm on the crowded $2 \times n$ room without wallflowers is faster if $n \geq 5$ and the cycle algorithm with wallflowers is faster if $n < 5$.*

For the 2×3 and 2×4 grids, we used a heuristic search to show that this gives one of optimal solutions; see Figure 2. It is unknown whether the cycling without wallflowers is optimal for $n \geq 5$.

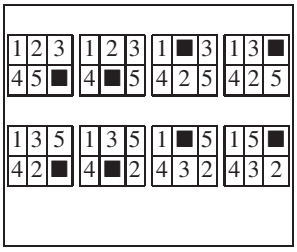

Fig. 2. One of four optimal solutions for the 3×2 case with an unoccupied corner.

The cycle method can be extended to larger grids in what we call the *boustrophedon* algorithm. When m or n is even, bend the cycle snakelike throughout the room, alternating right to left and left to right. If one of the dimensions of the room is odd, then the furrows run parallel to this dimension; see Figure 3a.

If both m and n are odd, the algorithm uses a different setup; see Figure 3b. Mark the people in the third row from the bottom, except those in the leftmost or rightmost two columns, as wallflowers. The cycle starts immediately above the wallflowers and snakes around the upper right $(m-2) \times (n-3)$ grid as in the even case. Wallflowers are

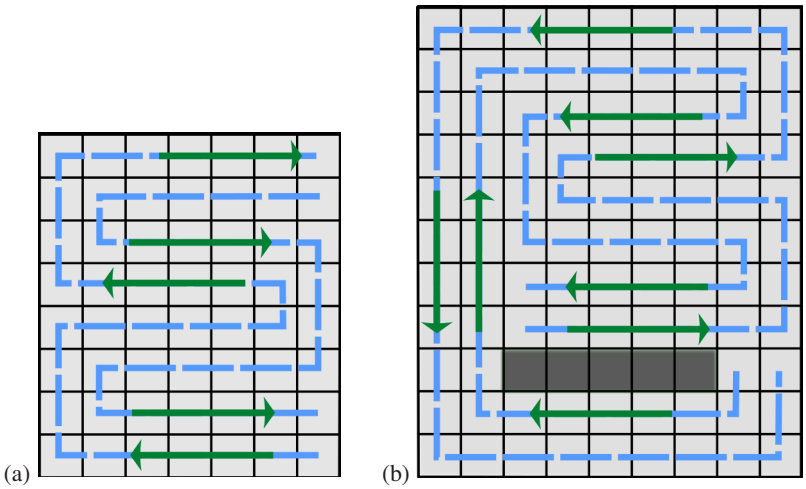

Fig. 3. (a) The circuit for an 7×8 grid. The path curves boustrophedonically and the furrows run parallel to the odd dimension. (b) The circuit for a 9×11 grid. Each person moves along the dotted lines in the direction of the arrows, except the wallflowers, who are darker. Note that the wallflowers abut both ends of the path.

excluded (as they are in many gatherings). The cycle then goes around the remainder of the grid, under the wallflowers, and up to fill the rest of the pixels. Note that the path starts and ends adjacent to the wallflowers. In this configuration, we will show that people only need to walk a limited distance around the circuit to guarantee that every moving person has kissed everyone else. Note that wallflowers have not yet kissed each other, so at the end, the $2 \times (m-4)$ solution is used with the wallflowers and the people immediately above, to ensure that all wallflowers kiss.

We divide the grid into two parts, an *outgoing route* and *incoming route*. These are defined similarly to those in the $2 \times n$ case, each route representing one of the two paths of width 1 that make up the path of width 2 filling the room. These are shown in Figure 3 as two separate dotted lines, with arrows to show the direction of travel.

Lemma 3. *On a $n \times m$ crowded room, the boustrophedon algorithm enables all people to kiss each other. If ℓ is the length of the longer of the two routes, at most $2\ell - 1$ cycles are required.*

Proof omitted due to lack of space.

Lemma 4. *A lower bound for the kissing problem on a crowded $n \times m$ grid is* $(m^2n^2 - 7mn + 12 - 2m - 2n)/4$.

Proof. Each non-corner border pixel is adjacent to three other pixels, each corner is adjacent to two, and the remaining pixels are adjacent to four others, for a total of $(8 + 6(n-2) + 6(m-2) + 4(n-1)(m-1))/2 = 2nm + n + m - 6$ kisses. This formula overcounts kisses we attributed to the unoccupied pixel. Therefore, there are no more than $2nm + n + m - 10$ kisses initially.

As in Lemma 2, when a person moves after the first time step, one of his neighbors must be unoccupied and one he has already kissed. Since each pixel has at most four neighbors, only two new kisses can be made per turn, except for the first time step, when three kisses can be made. Therefore, (# kisses) ≤ 2(# moves made) $+ 1$.

We take the number of kisses necessary, subtract the number of initial kisses, and combine with the bound on the number of moves t to get

$$t \geq n^2m^2/4 - 7mn/4 - n/2 - m/2 + 5.$$

□

Theorem 2. *The boustrophedon algorithm on an $n \times m$ crowded room is a $4 + o(1)$-approximation algorithm.*

Proof. By Lemma 3, the algorithm must run for $2\ell - 1$ cycles. If one of the sides is even the algorithm must run for $2\ell - 1$ cycles, where $\ell \leq mn/2 + 4$. This bound comes about because each time the path bends the longer route increases by at most 4, but since it bends back and forth the routes increase alternately. In total, therefore, the algorithm takes $(nm - 1)(nm + 7) = m^2n^2 + 7mn - 7$ time. We thus obtain an approximation ratio of

$$\frac{(nm - 1)(nm + 7)}{n^2m^2/4 - 7mn/4 - n/2 - m/2 + 3} = 4 + o(1).$$

The value of ℓ is more complicated in the odd case because the circuit is less regular. There is first a maximum route length of $2 + m + n$ over the irregular L-shape, then the $(m - 3)(n - 2)/2 + 4$ more to fill the remaining $(m - 3) \times (n - 2)$ grid. So in total, $\ell \leq (m - 3)(n - 2)/2 + m + n + 6$. Each cycle takes $nm - m - 3$ time, as all pixels except the one unoccupied and the $m - 4$ wallflowers must move. After this, we must do the $2 \times m - 4$ algorithm at a cost of $2(m - 4)(m - 5)$. We thus obtain an approximation of

$$\frac{((m - 3)(n - 2) + 2m + 2n + 11)(nm - m - 3) + 2(m - 4)(m - 5)}{n^2m^2/4 - 7mn/4 - n/2 - m/2 + 3} = 4 + o(1).$$

□

3 The Kissing Problem in a Comfortable Room

This section considers kissing in a *comfortable room*, in which k pixels are unoccupied for $1 < k < mn/2$. Because there are k unoccupied pixels, up to k moves and $\Theta(k)$ kisses can be made per time step. This section generalizes the boustrophedon algorithm from the previous section. The same circuit is used, so the series of positions after each cycle is the same, but more gaps means that people travel faster around the circuit. The boustrophedon algorithm now delivers a $10 + o(1)$-approximation to optimal.

Lemma 5. *In a comfortable room, after less than k time steps, we can guarantee that k people will be able to move forward along the cycle at each time step.*

Proof. Intuitively, each person with more than one empty space in front of him moves forward at each time step. Then any set of consecutive empty pixels must either stay the same in size (if the person in front and behind the set both move forward), or decrease in size (if only the person behind it moves forward). Since less than half of the pixels are empty, there must be a person with another person in front of him, and he cannot move, so the set behind him decreases in size. Therefore, the total number of consecutive empty pixels decreases each step, and since that total is no more than k, the people are appropriately spaced after k time steps. □

Since we have k unoccupied pixels, after the people are dispersed it is possible for k movements to be made simultaneously. However, this method may lead to errors: the movement of multiple people can result in misses when two people move past each other simultaneously on opposite routes. When this happens, the movements are split into two time steps such that any person in the outgoing route moves in one time step, and the people in the incoming route move in the next.

Lemma 6. *On a $n \times m$ grid with k blanks, the boustrophedon algorithm enables all people to kiss each other.*

Proof omitted due to lack of space.

Lemma 7. *In a comfortable room, the maximum number of kisses resulting from a given number of moves is (# kisses) $\leq \lceil \frac{5k}{2} \rceil$ (# moves). A lower bound for the kissing problem on a comfortable $n \times m$ grid is $(mn-1)(mn-2)/(5k+1)$.*

Proof. Similarly to the crowded-room case, if a person p_i moves into a pixel, one of the neighbors of the pixel was just vacated by p_i and now must be empty. Thus the number of kisses gained per move is no more than three.

However, this bound can be improved using a similar idea to that for the crowded-room case. Assume that p_i is at pixel s, which has only one adjacent unoccupied pixel at time t. Therefore, p_i has already kissed all people adjacent to s at time $t+1$ (no new people can be adjacent to s as the only unoccupied pixel next to s is now occupied by p_i). But these neighbors are the only people who can move into s, so no matter who moves into s, they have already kissed p_i and do not get a kiss from the pixel they vacated, for a total of two new kisses at most.

However, if s has more than one adjacent unoccupied pixel at time t, it is possible that some new person p_j is adjacent to s at time $t+1$. But then, p_j is adjacent to two blank squares at $t+1$ (the pixel it vacated and s, which must also be unoccupied as it previously contained p_i), so p_j can only gain two kisses. However, if it moves into s at time $t+2$, it may kiss all three people. Each blank square can produce no more than five kisses for every two moves, so (# kisses) $\leq \lceil \frac{5k}{2} \rceil$ (# moves).

There are $\binom{mn-1}{2}$ kisses that need to be made. There is no lower bound on the number of kisses that are made in the initial state, as the people could be in a checkerboard pattern with no two neighboring people and no kisses. Solving, we get

$$t \geq (mn-1)(mn-2)/(5k+1).$$

□

Theorem 3. *This modification of the boustrophedon algorithm on a grid with* $1 < k < mn/2$ *blanks takes* $\frac{(mn+7)(nm-m-3)}{k} + 2(m-4)(m-6) + k$ *time, and gives a* $10 + o(1)$*-approximation algorithm.*

Proof. The first step in the algorithm is to spread out the unoccupied pixels as mentioned in Lemma 5, at a cost of $\leq k - 1$ time. As mentioned in the proof of Lemma 6, this algorithm has the same number of cycles as the crowded room case, which was, in the worst case, $2\ell - 1 = mn + 7$. During each cycle, a total of $nm - m - 3$ people must move forward, and by Lemma 5, exactly k can move forward per time step, so each cycle takes $\lceil (nm - m - 3)/k \rceil$ time steps. After these cycles have been completed, as with the crowded room case, we must make sure the wallflowers kiss if both m and n are odd. There may be only one unoccupied pixel in the $2 \times (m-4)$ grid, so we cannot take advantage of parallelization and this step takes $2(m-4)(m-6)$ time in the worst case. Therefore, the total running time is

$$\frac{(mn+7)(nm-m-3)}{k} + 2(m-4)(m-6) + k.$$

Dividing by the lower bound, we get $10 + o(1)$, the approximation for this algorithm. □

4 The Kissing Problem in the Sparse Room

This section considers kissing in a *sparse room* in which the number of empty pixels $k \geq mn/2$.

Our strategy is to conglomerate all the people into the bottom rows using a sorting algorithm for a two-dimensional grid, and then to use the algorithm for comfortable rooms. We assume without loss of generality that $m \geq n$. Furthermore, we assume that one or more people occupy the first and last columns. If not, the algorithm still works, but the approximation ratio may be worse.

We compact the people into the lower part of the room using a sorting algorithm on a mesh, e.g., [30, 31, 35]. An asymptotically optimal sorting algorithm leads to a constant approximation.

An algorithm for sorting on a mesh arranges the elements in numerical order, boustrophedonically, from bottom to top. To apply the algorithm, label the p people using the odd numbers $1, 3, \ldots 2p-1$, and label the unoccupied pixels with the unused integers from $1, \ldots, mn$, so that after the sort no two people are adjacent either vertically or horizontally. After the sorting is completed, only the bottom n_f rows contain people, where n_f is the smallest integer satisfying $mn_f \geq 2p$.

When two (adjacent) pixels swap labels in the sorting algorithm, the people standing on those pixels may move in the kissing algorithm. Specifically, if exactly one of the pixels is occupied, then the person standing on that pixel moves onto the adjacent pixel. On the other hand, if the pixels are both occupied or both unoccupied, then the pixels switch labels, but there is no movement.

The next step is to use the algorithm for the comfortable room case on the $m \times n_f$ grid. Note that since $mn_f \geq 2p$, the room is still not comfortable, since a comfortable

room has more occupied pixels than vacant ones. Nonetheless, Lemma 6 still holds. Moreover, the people are already spread out, meaning that they do not block each other.

Theorem 4. *Assuming there is a person in the first and last column, the minimum number of moves for the sparse case is* $\max\{m-2,(p-1)/5\}$. *The running time of this algorithm is* $2mn_f + 3m + o(m)$*, which leads to a* $25 + o(1)$ *approximation ratio for the sparse case.*

Proof. The people in the first and last column require at least $m-2$ steps to kiss. Furthermore, analogous to Lemma 7, each move can only give $\lceil 5p/2 \rceil$ kisses since only p people move at a time. Therefore, the lower bound is $\max\{m-2,(p-1)/5\}$.

The sorting algorithm discussed in [30] is used to sort the room in $3m+o(m)$ time, so all people are in the bottom n_f rows. The boustrophedon algorithm requires no more than mn_f cycles, each of which takes two time steps (one for each route). Therefore, this algorithm takes a total of $2mn_f + 3m + o(m)$ time.

We examine the approximation ratio in two cases. First, assume $m-2 \geq (p-1)/5$. Then the lower bound is m and furthermore, $mn_f < 2p + m < 11m$. We can rewrite the running time in terms of m, then divide by the lower bound m, to get

$$\frac{25m + o(m)}{m} = 25 + o(1).$$

Similarly, assume $(p-1)/5 \geq m$, so the lower bound is $(p-1)/5$. By definition of n_f, $mn_f < 2p + m$. Then our running time can be written in terms of p as $4p + 3(p-1)/5 + o(p)$. Diving by the lower bound we get

$$\frac{5(p-1)/5 + 4p + o(p)}{(p-1)/5} = 25 + o(1).$$

Therefore, the algorithm is a 25-approximation of optimal. □

5 Conclusion

We now bid readers adieu. Rather than giving individual kisses, we take our leave with phatic comments on open problems and future work (the scholarly equivalent of the multicast "bye y'all").

This paper considers kissing only in rectangular rooms. How quickly can a gathering break up in a less austere environment than a rectangle? What about rectilinear polygons, possibily with holes (to model those parties where people don't stand on furniture)?

The boustrophedon algorithm presented here is likely to have better approximation ratios than this paper proves. Could some nontrivial version of the algorithm even be optimal? The complexity of kissing problem remains open for any environment.

References

1. S. Alpern. Rendezvous search on labeled networks. *Naval Research Logistics*, 49(3):256–274, 2002.

2. S. Alpern, V. Baston, and S. Essegaier. Rendezvous search on a graph. *Journal of Applied Probability*, 36(1):223–231, 1999.
3. E. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.
4. R. Arkin. Motor schema-based mobile robot navigation. In *Proc. IEEE Conference on Robotics and Automation*, pages 264–271, 1987.
5. T. Balch and R. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.
6. T. Balch and M. Hybinette. Behavior-based coordination of large-scale robot formations. In *Proc. 4th International Conference on MultiAgent Systems*, pages 363–364, 2000.
7. M. Batalin and G. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *Proc. 6th International Symposium on Distributed Autonomous Robotic Systems*, pages 373–382, 2002.
8. W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 476–481, 2000.
9. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *Proc. 30th International Conference on Automata, Languages and Programming (ICALP)*, pages 192–192, 2003.
10. J. Culberson and J. Schaeffer. Efficiently searching the 15-puzzle. Technical report, Department of Computing Science, University of Alberta, 1994.
11. S. Das, P. Flocchini, N. Santoro, and M. Yamashita. On the computational power of oblivious robots: forming a series of geometric patterns. In *Proc. 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 267–276, 2010.
12. A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. *Proc. 11th Annual European Symposium on Algorithms (ESA)*, pages 184–195, 2003.
13. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed coordination of a set of autonomous mobile robots. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 480–485, 2000.
14. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous oblivious robots with limited visibility. *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 247–258, 2001.
15. M. Garey, R. Graham, and D. Johnson. Some NP-complete geometric problems. In *Proc. 8th Annual ACM Symposium on Theory of Computing (STOC)*, pages 10–22, 1976.
16. V. Gervasi and G. Prencipe. Need a fleet? Use the force! In *Proc. 2nd International Conference on Fun With Algorithms (FUN)*, pages 149–164, 2001.
17. J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. *Nordic Journal of Computing*, 6(4):469, 1999.
18. R. Hearn and E. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.
19. R. A. Hearn and E. D. Demaine. The nondeterministic constraint logic model of computation: Reductions and applications. In *Proc. 29th International Conference on Automata, Languages and Programming (ICALP)*, pages 778–778, 2002.
20. R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters, Ltd., 2009.
21. J. Hopcroft, J. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.
22. E. Hordern. *Sliding Piece Puzzles*. Oxford University Press, 1986.
23. A. Howard, M. Matarić, and G. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

24. A. Howard, M. Matarić, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proc. 6th International Symposium on Distributed Autonomous Robotics Systems (DARS)*, pages 299–308, 2002.
25. T. Hsiang, E. Arkin, M. Bender, S. Fekete, and J. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Proc. 5th Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 77–94, 2004.
26. Y. Hwang and N. Ahuja. Gross motion planning a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.
27. F. Karlemo and P. Östergård. On sliding block puzzles. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 34(1):97–107, 2000.
28. D. Kowalski and A. Malinowski. How to meet in anonymous network. *Structural Information and Communication Complexity*, pages 44–58, 2006.
29. R. Kurazume and S. Nagata. Cooperative positioning with multiple robots. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1250–1257, 1994.
30. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann Publishers, 1992.
31. D. Nassimi and S. Sahni. Bitonic sort on a mesh-connected parallel computer. *IEEE Transactions on Computers*, 100(1):2–7, 1979.
32. D. Rater and M. Warmuth. Finding a shortest solution for the nxn extension of the 15-puzzle is intractable. *Journal of Symbolic Computation*, 10:111–137, 1990.
33. H. Ratliff and A. Rosenthal. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.
34. I. M. Rekleitis, G. Dudek, and E. E. Milios. Graph-based exploration using multiple robots. In *Proc. 5th International Symposium on Distributed and Autonomous Robotic Systems (DARS)*, pages 241–250, 2000.
35. I. Scherson and S. Sen. Parallel sorting in two-dimensional VLSI models of computation. *IEEE Transactions on Computers*, 38(2):238–249, 1989.
36. R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proc. National Conference on Artificial Intelligence (AAAI)*, pages 852–858, 2000.
37. K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 254–259, 1993.
38. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.
39. I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.
40. J. Wang. On sign-board based inter-robot communication in distributed robotic systems. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1045–1050, 1994.
41. R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.
42. B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. 2nd International Conference on Autonomous Agents*, pages 47–53, 1998.