Theory Comput Syst (2014) 54:715–730
DOI 10.1007/s00224-013-9484-x

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

Michael A. Bender · Ritwik Bose · Rezaul Chowdhury · Samuel McCauley

Published online: 25 May 2013

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 $45+o(1)$-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 (more than half the pixels are empty) with two people abutting the far walls of the room.

This research was supported in part by NSF Grants CCF 0937822, CCF 1114809, CCF 0634793, IIS 1247726, and DOE Grant DE-FG02-08ER25853.

M.A. Bender · R. Chowdhury · S. McCauley (✉)
Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400, USA
e-mail: smccauley@cs.stonybrook.edu

M.A. Bender
e-mail: bender@cs.stonybrook.edu

R. Chowdhury
e-mail: rezaul@cs.stonybrook.edu

M.A. Bender
Tokutek, Inc., Lexington, MA 02421, USA

R. Bose
Department of Computer Science, University of Rochester, Rochester, NY 14627, USA
e-mail: rbose@rochester.edu

This paper gives optimal solutions for small cases, which were found using a heuristic state space search (IDA*).

Keywords Algorithms · Kissing · Goodbye · Block sliding · Approximation algorithms · Routing · Communication

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."[1]). 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 $P = \{1 \dots p\}$ of people. At each unit of time, any subset $S \subseteq P$ of people can move to adjacent unoccupied pixels.[2]

Our goal is to minimize the *makespan*, that is, the number of time steps until people have completed all pairwise kisses. A *kiss* is transacted between i and j when they occupy adjacent pixels. 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 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, 34]: 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 [33, 43] 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

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

[2] Two pixels are said to be *adjacent* if they share an edge.

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, 39], dispersion [25, 41], exploration and mapping [7, 24, 29, 35, 37, 38, 44], rendez-vous [1, 2, 9, 12, 14, 28], and motion planning [4–6, 8, 13, 16, 23, 26]. Reference [42], 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.

This problem is also similar to round-robin tournaments, in which each player must play every other. The pairwise meetings of round-robin tournaments are the same, but the kissing problem allows for simultaneous kisses and restricts the movements of the players, and thus requires different methodology. However, the "cycling without wallflowers" algorithm presented in Sect. 2 is very similar to the Circle Method of round-robin tournaments described in [31, 40].

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 $45 + o(1)$-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 there are more unoccupied pixels than people, as long are there are people abutting the furthest pair of walls. Without this butts-abutting restriction, the algorithm still works, but the approximation ratio can be arbitrarily large.
- 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 Sects. 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 crowded $2 \times n$ 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.

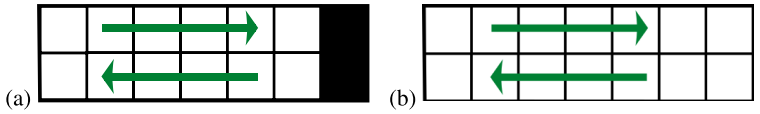

Fig. 1 Two methods for solving the kissing problem on a 2×7 grid. *Arrows* indicate the direction of the two routes. (**a**) The two rightmost people (*shown in black*) remain stationary. (**b**) Everyone participates in the cycle

Our algorithm is based on a circuit that each person follows around the grid. A *wallflower* is a person who stands still and does not participate in this circuit. The algorithms we present do not always have wallflowers. 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 Fig. 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.

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* $2n - 3$ *cycles with wallflowers.*

Proof For the case with no wallflowers (Fig. 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 left to right across the top row. If there are wallflowers (Fig. 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 Fig. 1.

Consider only the kisses that happen when two people are in different routes. If a kiss happens between a person at pixel i and a person at pixel 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 $2n - 1$ time steps. 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, $p_i(t) \equiv i + t - 1 \pmod{2n}$.

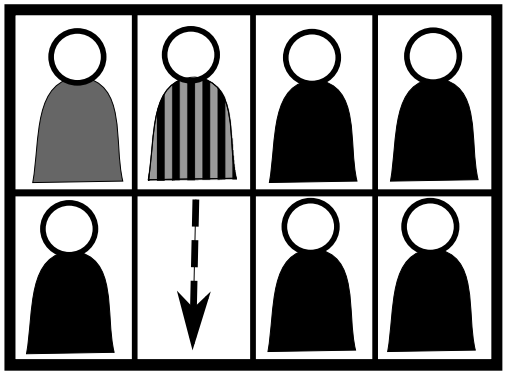

Fig. 2 An illustration of the lower bound for the hallway case. Once the *striped person* moves (shown with an *arrow*), whoever moves next (the *dark grey person*, for example) must have already kissed him

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 all 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 he has passed through pixels 1 and $2(n - 1)$. Therefore, everyone has kissed the wallflowers after $2n - 3$ 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 pixel, 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 Person 1 moves into an empty pixel, that pixel must have been vacated by someone else (Person 2). But Person 2 must have already kissed Person 1 before vacating, so they have already kissed when Person 1 moves into the pixel. An example of this is shown in Fig. 2; if the striped person

moves as shown by the arrow, whoever moves on the next timestep (like the grey person that was to his left) will have already kissed the striped person. 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. We know that p_i is adjacent to s at turn t, 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

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

Solving for t,

$$t \geq 2n^2 - 6n + 3.$$

□

Theorem 1 *For a $2 \times n$ grid in the crowded room, cycling without wallflowers takes $2n(n-1)$ time. Cycling with wallflowers takes $(2n-4)(2n-3) + \lfloor n/2 \rfloor - 1$ 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, let p be the person who begins next to a wallflower, and moves away from it during a cycle. There must be $(2n - 3)$ cycles before p kisses the other wallflower. However, if the wallflowers are chosen to be as close to the initial unoccupied square as possible, the first cycle need not be completed for p to move. In the worst case, the unoccupied square is $\lfloor n/2 \rfloor - 1$ away from p. The remaining $(2n-4)$ cycles must be completed, each of which moves $(2n-3)$ people, leading to a total running time of $(2n-4)(2n-3) + \lfloor n/2 \rfloor - 1$.

We divide the running time without wallflowers by the lower bound to get the approximation

$$\frac{2n^2 - n}{2n^2 - 6n + 3} = 1 + o(1).$$

□

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 the optimal solutions; see Fig. 3. It is unknown whether the cycling without wallflowers is optimal for $n \geq 5$.

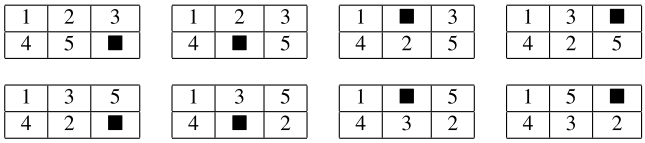

Fig. 3 One of four optimal solutions for the 3×2 case with an unoccupied corner. The unoccupied pixel is shown in *black*. The people are numbered simply for distinguishability

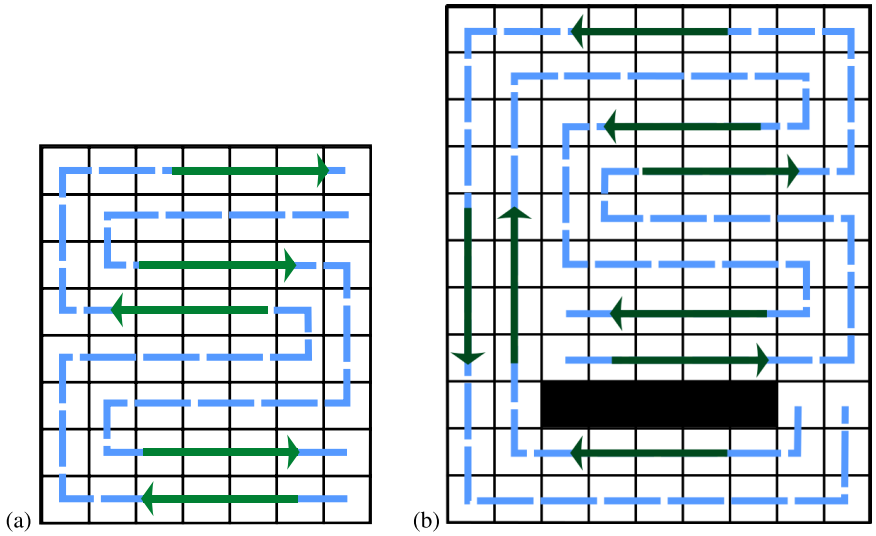

Fig. 4 (**a**) The circuit for an 7×8 grid. The path *curves* boustrophedonically and the *furrows* run parallel to the odd dimension. (**b**) The circuit for a 9×11 grid. Each person moves along the *dotted lines* in the direction of the *arrows*, except the wallflowers, who are colored *black*. Note that the wallflowers abut both ends of the path

The cycle method can be extended to larger grids in what we call the *boustrophedon* algorithm.[3] This algorithm uses the same idea of a continuous cycle of people from the hallway case. 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 Fig. 4a.

If both m and n are odd, the algorithm uses a different setup; see Fig. 4b. 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

[3]Look it up t'nod uoy fi know what -ehportsoub onic means.

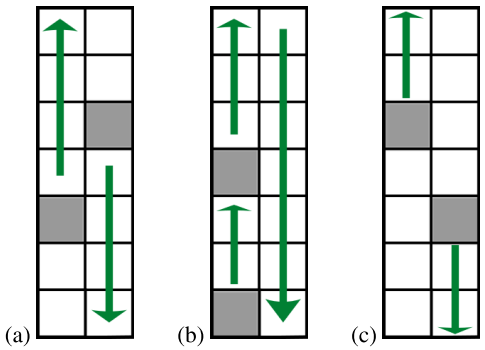

Fig. 5 The three cases used in the proof. p_i and p_j are colored *grey*. (**a**) In Case 1, p_i and p_j are moving towards each other. (**b**) In Case 2, p_i and p_j are on the same route. (**c**) In Case 3, p_i and p_j are moving away from each other

snakes around the upper right $(m-2) \times (n-3)$ grid as in the even case. Wallflowers are 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. We must reexamine how long each person walks due to the corners in the routes; while it is intuitively clear that two people will still pass even as the circuit bends, the proof of Lemma 1 does not apply immediately. 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 one that make up the path of width two filling the room. These are shown in Fig. 4 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 Consider two people p_i and p_j. Let the number of pixels remaining on their routes be ℓ_i and ℓ_j, respectively, and assume without loss of generality that $\ell_i \leq \ell_j$.

We consider three cases for the initial placement of p_i and p_j relative to one another, assuming that they do not kiss in the initial placement. In Case 1, p_j lies on the opposite route from p_i, but they are moving towards each other; that is, p_j is adjacent to a pixel between p_i and the end of p_i's route (a pixel *after* p_i). Case 2 has p_j on the same route as p_i. Case 3 has p_j on the opposite route from p_i, but p_j is adjacent to a pixel that is between p_i and the beginning of p_i's route (a pixel *before* p_i). Note that if p_i and p_j are in a corner with p_j on the inside and p_i on the outside of the corner, p_j has neighbors both before and after p_i, whereas p_i has no neighbors on the opposite route at all. In this situation, we define p_i and p_j to be in Case 3. These cases are diagrammed in Fig. 5.

We define two people as having *passed* if they were in Case 1 in the past and are now in Case 3. In other words, they were heading towards each other, and are now heading away from each other. We show that p_i and p_j pass or kiss in no more than $2\ell - 1$ cycles, regardless of their starting state. We then show that if p_i and p_j pass, they kiss.

Case 1: After $\ell_i - 1$ cycles, p_i is in the last pixel on his route. Person p_j must still be on his route since $\ell_i \leq \ell_j$, so p_i and p_j are in Case 3, or p_j is in the first pixel of his route, in which case they have kissed. Therefore, p_i and p_j have passed or kissed in at most $\ell_i - 1$ cycles.

Case 2: After ℓ_i steps, p_i is on the first pixel of the opposite route, so p_i and p_j are in Case 1 or have kissed. After another $\ell_j - \ell_i - 1$ steps they are in Case 3 or have kissed (similarly to the proof for Case 1), for a total of $\ell_j - 1$ steps at most until j and i pass.

Case 3: After ℓ_i cycles, p_i is on the same route as p_j, and p_j has $\ell_j - \ell_i$ pixels after him on his route. We examine two later points: (1) After $\ell_j - \ell_i$ more cycles, p_j will be the first pixel on its opposite route, and p_i will remain on the same route, so they will be in Case 1. (2) After $\ell - 1$ cycles, p_i will be in the last pixel of his route, so p_i and p_j must be in Case 3 again. They will be in Case 1 at point 1, and will be in Case 3 at point 2, so they will have passed after a total of $\ell + \ell_i - 1 \leq 2\ell - 1$ cycles.

Now we show that if two people p_i and p_j pass, they must kiss. Note that the transition from Case 1 to Case 3 must be direct; by the case definitions it is impossible to enter Case 2 from Case 1. People p_i and p_j may kiss between Cases 1 and 3, in which case our assumption is proven. If not, the points are in Case 3 immediately after Case 1. Let t be the number of cycles after which p_i and p_j first enter Case 3 from Case 1. At time $t - 1$, p_i and p_j are in Case 1. Therefore, they are in different routes at $t - 1$; assume p_i moves first. Let $R(p(t))$ be the pixel adjacent to $p(t)$ in the opposite route (if there is more than one, choose the one with the longest distance to the end of the route. If p has no neighbor in the opposite route, take the neighbor of the closest pixel to p that does have a neighbor in the opposite route, again breaking ties by longest distance to the end of the route). Since each person moves forward one step during a cycle, $p_i(t-1)$ is exactly one step away from $p_i(t)$, and the same holds for p_j. We know that at t, p_i and p_j are in Case 3, so by definition $R(p_j(t))$ is behind $p_i(t)$, likewise $R(p_i(t))$ is behind $p_j(t)$. Then both $R(p_i(t))$ and $p_j(t-1)$ are behind $p_j(t)$, so since $p_j(t-1)$ is adjacent to $p_j(t)$, $R(p_i(t))$ is behind or equal to $p_j(t-1)$. If $R(p_i(t))$ is behind $p_j(t-1)$, $R(p_j(t-1))$ is behind $p_i(t)$. But $R(p_j(t-1))$ is in front of $p_i(t-1)$ since they are in Case 1, which contradicts that $p_i(t)$ and $p_i(t-1)$ are adjacent. But then the other case is true, $R(p_i(t))$ is equal to $p_j(t-1)$, so $p_i(t)$ is adjacent to $p_j(t-1)$. Since p_i was assumed to move first, these two positions occur at the same time, and p_i and p_j kiss. □

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, for a total

of $(8 + 6(n-2) + 6(m-2) + 4(n-2)(m-2))/2 = 2nm - n - m$ kisses. This formula overcounts at least two kisses we attributed to the unoccupied pixel. Therefore, there are no more than $2nm - n - m - 2$ 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

$$t \geq n^2m^2/4 - 7mn/4 + n/2 + m/2 + 1. \qquad \square$$

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, $\ell \leq mn/2 + 4$. 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 $(nm - 1)(nm + 7) = m^2n^2 + 6mn - 7$ time. We thus obtain an approximation ratio of

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

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

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

$\square$

There are several other algorithms that require $O(m^2n^2)$ time and thus achieve the same approximation. For example, it is possible to use the crowded hallway algorithm as a black box, making each pair of columns kiss.

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 $3k$ 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 $45 + o(1)$-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 along the cycle (into the empty space) 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, moving everyone simultaneously may lead to missed kisses 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 We follow a similar structure to the proof of Lemma 3, and show that any two people p_i and p_j must kiss eventually.

As in the proof of Lemma 3, p_i and p_j must either pass or kiss. The proof of this extends immediately to the case with multiple blanks because by definition of a cycle, adding extra blanks does not change the position of p_i and p_j after a cycle—each has moved forward once. We proved that after $2\ell - 1$ cycles p_i and p_j must pass or kiss; because the positions of all people after each cycle is the same, this remains true even if there are multiple blanks.

Then we only need to show that if two people pass, they kiss, even if there are multiple blanks (allowing both people to move simultaneously). If two people pass, they must be on opposite routes by definition. But in our algorithm, two people on different routes cannot move at the same time (each route moves individually). Therefore, if two people pass, they must move one at a time, and the argument from Lemma 3 still applies. □

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 $(mn - k)(mn - k - 1)/(5k + 1)$.*

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

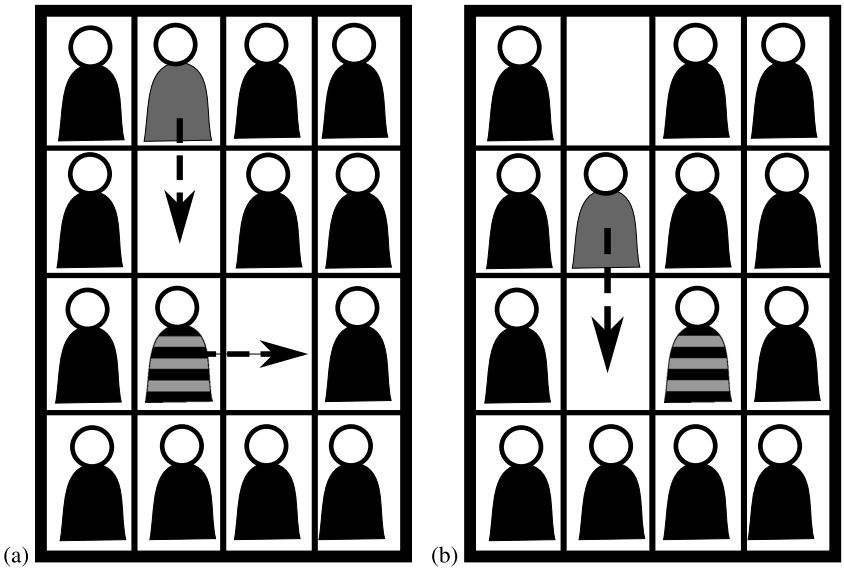

Fig. 6 An example for the lower bound for the comfortable room. (**a**) The *striped person* moves *right*, and the *grey person* moves into a space next to the one just vacated. (**b**) If the *grey person* moves *down* during the next time step he kisses three new people

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 anyone in the pixel they vacated (as the pixel is unoccupied), 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 unoccupied pixels 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 unoccupied pixel can produce no more than five kisses for every two moves, so (# kisses) $\leq \lceil \frac{5k}{2} \rceil$(# moves). Figure 6 shows an example where a person can kiss two people during one time step, and kiss three during the next time step.

There are $\binom{mn-k}{2}$ 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

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

We formalize the *comfortable boustrophedon algorithm* as follows: First, move any person with more than two unoccupied pixels ahead of him along his cycle forward, as in Lemma 5. Continue moving such people until there are no two consecutive unoccupied pixels along the cycle. Second, move the k people with unoccupied pixels in ahead of them forward along the cycle. Continue this movement until all pairs

of non-wallflowers have kissed. Third, if both m and n are odd, there are wallflowers which have not kissed any people. Move a blank square adjacent to one of the wallflowers, and then perform the crowded hallway case from Sect. 2 on the row of wallflowers and the row above them, causing all pairs of wallflowers to kiss.

Theorem 3 *The comfortable boustrophedon algorithm on a grid with* $1 < k < mn/2$ *blanks takes* $\frac{2((m-3)(n-2)+2m+2n+11)(nm-m-3)}{k} + 2(m-4)(m-6) + m + n + k$ *time, and gives a* $45 + o(1)$*-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 $\leq k$ 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, $2\ell - 1 = (m-3)(n-2) + 2m + 2n + 11$. During each cycle, a total of $nm - m - 3$ people must move forward, and by Lemma 5, exactly k can move forward every two time steps (one per route), so each cycle takes $\lceil 2(nm - m - 3)/k \rceil$ 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. An additional $m + n$ time may be required to bring an unoccupied square adjacent to the wallflowers. Therefore, the total running time is

$$\frac{2((m-3)(n-2) + 2m + 2n + 11)(nm - m - 3)}{k} + 2(m-4)(m-6) + m + n + k.$$

(Note that we did not take the ceiling of the time required for a cycle; this will at worst add two to the total running time.) Dividing by the lower bound, we get $45 + o(1)$, 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 is $k \geq mn/2$ (more than half the room is unoccupied).

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 $m \geq n$ (there are more columns than rows). 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 leave the approximation ratio for instances without people in the first and last column as an open problem.

We compact the people into the lower part of the room using a sorting algorithm on a mesh, e.g., [30, 32, 36]. 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.

The *sparse boustrophedon algorithm*, which solves the kissing problem in a sparse room, proceeds as follows: First, 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 will 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.

Second, use the comfortable boustrophedon algorithm 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, and the first step of the comfortable boustrophedon algorithm is unnecessary.

Theorem 4 *Assuming there is a person in the first and last column, the minimum running time 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 $(p-1)/5 \geq m-2$, 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 $4p + 3(p-1)/5 + o(p)$. Dividing by the lower bound we get

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

Therefore, the algorithm has a performance ratio of $25 + o(1)$. □

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, possibly 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 the kissing problem remains open for any environment.

References

1. Alpern, S.: Rendezvous search on labeled networks. Nav. Res. Logist. **49**(3), 256–274 (2002)
2. Alpern, S., Baston, V., Essegaier, S.: Rendezvous search on a graph. J. Appl. Probab. **36**(1), 223–231 (1999)
3. Arkin, E., Hassin, R.: Approximation algorithms for the geometric covering salesman problem. Discrete Appl. Math. **55**(3), 197–218 (1994)
4. Arkin, R.C.: Motor schema—based mobile robot navigation. Int. J. Robot. Res. **8**(4), 92–112 (1989)
5. Balch, T., Arkin, R.: Behavior-based formation control for multirobot teams. IEEE Trans. Robot. Autom. **14**(6), 926–939 (1998)
6. Balch, T., Hybinette, M.: Behavior-based coordination of large-scale robot formations. In: Proc. 4th International Conference on Multi Agent Systems, pp. 363–364 (2000)
7. Batalin, M., Sukhatme, G.: Spreading out: a local approach to multi-robot coverage. In: Proc. 6th International Symposium on Distributed Autonomous Robotic Systems, pp. 373–382 (2002)
8. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), vol. 1, pp. 476–481 (2000)
9. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Solving the robots gathering problem. In: Automata, Languages and Programming, pp. 1181–1196. Springer, Berlin (2003)
10. Culberson, J., Schaeffer, J.: Efficiently searching the 15-puzzle. Technical report, Department of Computing Science, University of Alberta (1994)
11. Das, S., Flocchini, P., Santoro, N., Yamashita, M.: 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), pp. 267–276 (2010)
12. Dessmark, A., Fraigniaud, P., Kowalski, D.R., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica **46**(1), 69–96 (2006)
13. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Distributed coordination of a set of autonomous mobile robots. In: Proc. IEEE Intelligent Vehicles Symposium (IV), pp. 480–485 (2000)
14. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous robots with limited visibility. Theor. Comput. Sci. **337**(1), 147–168 (2005)
15. Garey, M., Graham, R., Johnson, D.: Some NP-complete geometric problems. In: Proc. 8th Annual ACM Symposium on Theory of Computing (STOC), pp. 10–22 (1976)
16. Gervasi, V., Prencipe, G.: Coordination without communication: the case of the flocking problem. Discrete Appl. Math. **144**(3), 324–344 (2004)
17. Gudmundsson, J., Levcopoulos, C.: A fast approximation algorithm for TSP with neighborhoods. Nord. J. Comput. **6**(4), 469 (1999)
18. Hearn, R., Demaine, E.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theor. Comput. Sci. **343**(1–2), 72–96 (2005)

19. Hearn, R.A., Demaine, E.D.: The nondeterministic constraint logic model of computation: reductions and applications. In: Proc. 29th International Conference on Automata, Languages and Programming (ICALP), pp. 401–413 (2002)
20. Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. AK Peters, Wellesley (2009)
21. Hopcroft, J., Schwartz, J., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". Int. J. Robot. Res. **3**(4), 76–88 (1984)
22. Hordern, E.: Sliding Piece Puzzles. Oxford University Press, London (1986)
23. Howard, A., Matarić, M., Sukhatme, G.: An incremental self-deployment algorithm for mobile sensor networks. Auton. Robots **13**(2), 113–126 (2002)
24. Howard, A., Matarić, M., Sukhatme, G.: 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), pp. 299–308 (2002)
25. Hsiang, T., Arkin, E., Bender, M., Fekete, S., Mitchell, J.: Algorithms for rapidly dispersing robot swarms in unknown environments. In: Proc. 5th Workshop on Algorithmic Foundations of Robotics (WAFR), pp. 77–94. Springer, Berlin (2004)
26. Hwang, Y., Ahuja, N.: Gross motion planning—a survey. ACM Comput. Surv. **24**(3), 219–291 (1992)
27. Karlemo, F., Östergård, P.: On sliding block puzzles. J. Comb. Math. Comb. Comput. **34**(1), 97–107 (2000)
28. Kowalski, D.R., Malinowski, A.: How to meet in anonymous network. Theor. Comput. Sci. **399**(1), 141–156 (2008)
29. Kurazume, R., Nagata, S.: Cooperative positioning with multiple robots. In: Proc. IEEE International Conference on Robotics and Automation, pp. 1250–1257 (1994)
30. Leighton, F.T.: Introduction to Parallel Algorithms and Architectures. Morgan Kaufmann, San Mateo (1992)
31. Lucas, É.: Récréations Mathématiques vol. 2. Gauthier-Villars, Paris (1883). Sixième récréation: Les jeux de demoiselles
32. Nassimi, D., Sahni, S.: Bitonic sort on a mesh-connected parallel computer. IEEE Trans. Comput. **100**(1), 2–7 (1979)
33. Rater, D., Warmuth, M.: Finding a shortest solution for the nxn extension of the 15-puzzle is intractable. J. Symb. Comput. **10**, 111–137 (1990)
34. Ratliff, H., Rosenthal, A.: Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper. Res. **31**(3), 507–521 (1983)
35. Rekleitis, I.M., Dudek, G., Milios, E.E.: Graph-based exploration using multiple robots. In: Proc. 5th International Symposium on Distributed and Autonomous Robotic Systems (DARS), pp. 241–250 (2000)
36. Scherson, I., Sen, S.: Parallel sorting in two-dimensional VLSI models of computation. IEEE Trans. Comput. **38**(2), 238–249 (1989)
37. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: Proc. National Conference on Artificial Intelligence (AAAI), pp. 852–858 (2000)
38. Singh, K., Fujimura, K.: Map making by cooperating mobile robots. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 254–259 (1993)
39. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999)
40. van't Hof, P., Post, G., Briskorn, D.: Constructing fair round robin tournaments with a minimum number of breaks. Oper. Res. Lett. **38**(6), 592–596 (2010)
41. Wagner, I., Lindenbaum, M., Bruckstein, A.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. **15**(5), 918–933 (1999)
42. Wang, J.: On sign-board based inter-robot communication in distributed robotic systems. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 1045–1050 (1994)
43. Wilson, R.M.: Graph puzzles, homotopy, and the alternating group. J. Comb. Theory, Ser. B **16**(1), 86–96 (1974)
44. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proc. 2nd International Conference on Autonomous Agents, pp. 47–53 (1998)