Optimal Deterministic Protocols for Mobile Robots on a Grid[*]

Roberto Grossi
Dipartimento di Informatica
Università di Pisa
Corso Italia 40
I-56125 Pisa, Italy
grossi@di.unipi.it

Andrea Pietracaprina, Geppino Pucci
Dipartimento di Elettronica e Informatica
Università di Padova
Via Gradenigo 6/a
I-35131 Padova, Italy
{andrea,geppo}@artemide.dei.unipd.it

[*]Part of this work was done while the authors were attending the *Research Retreat on Sense of Direction and Compact Routing* held in June 1997 at the Certosa di Pontignano, Siena, Italy. A preliminary version of this paper appeared in the *Proceedings of the Sixth Scandinavian Workshop on Algorithm Theory, 1998.* This work was supported, in part, by the Italian CNR, by UNESCO under contract UVO-ROSTE 875.631.9, and by the Italian MURST under projects *Algorithms for Large Data Sets: Science and Engineering* and *Methodologies and Tools of High Performance Systems for Multimedia Applications.*

Running Head

Deterministic Protocols for Mobile Robots

Corresponding Author

Roberto Grossi

Dipartimento di Informatica
Università di Pisa
Corso Italia 40
I-56125 Pisa, Italy

Phone: +39 050 887293
Fax: +39 050 887226
E-mail: grossi@di.unipi.it

Abstract

This paper studies a system of m robots operating in a set of n work locations connected by aisles in a $\sqrt{n} \times \sqrt{n}$ grid, where $m \leq n$. From time to time the robots need to move along the aisles, in order to visit disjoint sets of locations. The movement of the robots must comply with the following constraints: (1) no two robots can collide at a grid node or traverse a grid edge at the same time; (2) a robot's sensory capability is limited to detecting the presence of another robot at a neighboring node. We present a deterministic protocol that, for any small constant $\epsilon > 0$, allows $m \leq (1-\epsilon)\, n$ robots to visit their target locations in $O\left(\sqrt{dn}\right)$ time, where each robot visits no more than $d \leq n$ targets and no target is visited by more than one robot. We also prove a lower bound showing that our protocol is optimal. Prior to this paper, no optimal protocols were known for $d > 1$. For $d = 1$, optimal protocols were known only for $m \leq \sqrt{n}$, while for general $m \leq n$ only a suboptimal randomized protocol was known.

Keywords

Multi-Robot Grid System, Routing Protocol, Mesh, Social Laws, Computational Agents.

List of Symbols Used

1	one
l	lower-case ell
à	accented lower-case a
@	at (for e-mail)
•	bullet (itemization)
0	zero
O	upper-case letter oh
$O(n)$	italic upper-case oh (big-oh notation)
Ω	upper-case Greek omega
Θ	upper-case Greek Theta
$\sqrt{n}$	square root
$a \times b$	a times b
$<, \leq, >, \geq$	standard inequalities
$\oplus$	circled plus (indicating exclusive-or)
$\log n$	logarithm
$\max\{X, Y\}$	maximum
x'	x prime
x_i	x sub i (indexing)
a^b	a to the b-th power
Σ	upper-case Greek sigma (indicating summation)
δ	lower-case Greek delta
ϵ	lower-case Greek epsilon
I, II, III, IV	capital roman numbers
$a \cdot b$	a dot b (indicating a multiplied by b)
$\lfloor X \rfloor$	floor of X
$\lceil X \rceil$	ceiling of X
$a, b, \ldots, z$	ellipsis
$\{a, b, c\}$	curly brackets (indicating set notation)

1 Introduction

A *Multi Robot Grid* system (shortly, MRG) consists of m *robots* that operate in a set of $n \geq m$ work locations connected by aisles in a $\sqrt{n} \times \sqrt{n}$ grid [ST95]. At any time, the robots are located at distinct grid nodes, and from time to time each robot is given a set of work locations (*targets*) to be visited. The target sets are disjoint and no particular order is prescribed for visiting the targets in each set. Moreover, robots may end up at arbitrary locations, once their visits are completed. We may regard the system as representing a warehouse or a tertiary storage (e.g., tape) system, where robots are employed to gather or redistribute items. We assume that the system is synchronous, that is, all robots are provided with identical clocks. Control is distributed in the sense that each robot's moves are scheduled locally by a processor embedded in the robot. Our goal is to design an efficient distributed on-line protocol that every robot must follow in order to visit the assigned targets while avoiding deadlocks and conflicts with other robots. More specifically, the protocol must comply with the following rules:

- At any time all the robots reside in the grid, i.e., no robot can leave the grid or enter from outside.
- No two robots can occupy a grid node or traverse a grid edge at the same time.
- A robot cannot exchange information with other robots directly. However, each robot is equipped with a short-range sensor that is able to detect the presence of other robots occupying nodes at (Manhattan) distance one from its current location.
- In one time unit, a robot can perform a constant amount of internal computation, read the output of its short-range sensor, and decide whether to remain at the current grid node, or to move to a neighboring node.

In an MRG(n, m, d) *problem*, each of $m \leq n$ robots in an MRG system is required to visit (at most) $d \leq n$ targets, with no grid node being target for more than one robot. For the sake of simplicity, we assume that n is a power of 4 so that the grid can be recursively decomposed into subgrids whose size is still a power of 4. The general case of n being any even power can be handled with minor modifications.

1.1 Related Work

Multi robot grid systems were originally introduced in [ST95, Sect. 2.1] as a practical case study within the general quest for *social laws* to coordinate agents sharing a common environment in a distributed rather than centralized fashion. While central control relies on a single arbiter that regulates all possible interactions among the agents, distributed control is based on a set local rules which must be complied with in order to avoid conflicts. The need for distributed control stems from a number of shortcomings which may limit the applicability of a central protocol, such as the need of reprogramming the system when the set of agents changes over time, or the overhead in computation and communication introduced by the arbiter. In fact, a distributed protocol may exploit better the intrinsic parallelism of the problem, since each agent can be programmed independently of the others to follow the common set of rules. In order to be efficient, a distributed protocol must require a minimal amount of communication to regulate the interaction between the agents. Hence, the protocol must be based on simple rules that can be applied locally and quickly.

Although the MRG(n, m, d) problem entails routing robots on a two-dimensional grid, it exhibits, however, some fundamental differences from classical message routing problems. The nodes of a network used to exchange messages are typically processing units able to compute, maintain a local status and, in many cases, temporarily buffer messages in transit. In contrast, in an MRG system the grid nodes are passive entities with no status or computing power, and robots, which are the active agents in the system, must orchestrate their movements solely based on their processing and sensory capabilities. Moreover, in message routing, packets traveling through the network can be destroyed and replicated as long as each message is eventually delivered to its destination(s), while this is clearly not admissible when dealing with robots.

For the above reasons, as was also observed in [PU96], none of the many message routing protocols known in the literature appears to be directly applicable to the MRG(n, m, d) problem (see [Lei92] and [Sib95] for comprehensive surveys of grid protocols). Even *hot-potato* protocols, which involve very simple operations at the nodes and do not need internal buffers, are not immediately portable, since they require that in one time unit a node is able to receive a packet from each neighbor, compare and manipulate the information carried by the packet headers, and finally redistribute the packets, suitably permuted, among the neighbors (e.g., see [NS95]). Another difference with the typical message routing scenario is that the m robots reside in the grid *all the time*, hence relocation becomes critical as m grows large. The main objective of the present paper is to show how the techniques employed in message routing can be suitably, yet not trivially, adapted to be applicable to the $\mathrm{MRG}(n, m, d)$ problem. For this purpose, we will develop efficient protocols for a set of basic primitives, such as balancing and sorting, which will then be used to orchestrate the robots' movements. We will also aim at solving the $\mathrm{MRG}(n, m, d)$ problem for values of the parameter m as close as possible to the maximum value n.

Preminger and Upfal [PU96] pointed out that any instance of the $\mathrm{MRG}(n, m, d)$ problem can be trivially completed in $n - 1$ steps by letting the robots circulate along a directed Hamiltonian cycle traversing all the grid nodes. In fact, they proved that any deterministic protocol in which robots are completely blind (i.e., a robot cannot detect the presence of other robots at any distance) requires $\Omega(n)$ time, thus implying the optimality of the trivial strategy in this case.

If the robots are not blind, $\Omega(\sqrt{n})$ time is necessary in the worst case due to the grid diameter. Clearly, a single robot with a single destination ($\mathrm{MRG}(n, 1, 1)$ problem) can achieve this bound by simply traversing a shortest path from its source to its target. For a larger number m of robots and a single destination per robot ($\mathrm{MRG}(n, m, 1)$ problem) two optimal $\Theta(\sqrt{n})$-time protocols are presented in [ST92] and [ST95] for two special cases. The first protocol is designed for $m \leq n^{1/4}$ robots, while the second one works for $m \leq \sqrt{n}$ robots, as long as they initially reside in distinct columns.

The only known protocol that deals with an arbitrary number of $m \leq n$ robots and a single destination is given in [PU96]. The algorithm is randomized and solves any $\mathrm{MRG}(n, m, 1)$ problem in suboptimal $O(\sqrt{n} \log n)$ time, with high probability. However, the algorithm assumes that a robot's short-range sensor is able to detect the presence of other robots at distance at most two, that a robot may initially stay outside the grid for an arbitrary amount of time, and that robots disappear from the grid as soon as they have visited their target. No deterministic protocol that takes $o(n)$ time and works under the stricter rules described in this paper is known for the $\mathrm{MRG}(n, m, 1)$ problem.

For the case of $d > 1$ targets, one could repeat the single-target protocol d times. However, as we will show in the paper, this strategy does not always guarantee optimal performance. To the best of our knowledge, no specific algorithm for the case of $d > 1$ targets has been developed so far.

1.2 Our Results

We begin by devising a simple and general protocol for the MRG$(n, m, 1)$ problem, with $m \leq n/4$, which attains optimal $\Theta\left(\sqrt{n}\right)$ time. The algorithm implements a routing strategy in a way that fully complies with the constraints imposed by an MRG system. Our protocol improves upon the work of [PU96] in several directions. First, the protocol is deterministic, hence it provides a worst-case guarantee on performance. Second, it achieves optimality for a wide range of m values, thus reducing the running time of [PU96] by an $O(\log n)$ factor. Third, it works in a weaker model in which the robots reside in the grid all the time and their sensors can detect other robots at distance one. It must be remarked, however, that our protocol requires a common clock governing all robots' movements, while the results in [PU96] and [ST95] can be adapted to hold under a slightly weaker notion of synchronization.

Next, we consider the case of $d > 1$ targets. If we put a constraint on the order of the visits that fixes *a priori* the sequence of targets to reach for each robot, a simple argument based on diameter considerations suffices to prove that any protocol for the problem requires $\Omega\left(d\sqrt{n}\right)$ time, in the worst case. Consequently, applying our optimal MRG$(n, m, 1)$ protocol d times yields an optimal $\Theta\left(d\sqrt{n}\right)$-time general solution in this case. However, if the robots can arbitrarily *rearrange* the order of their targets, the latter approach becomes suboptimal. Indeed, we prove an $\Omega\left(\sqrt{dn}\right)$ lower bound to the MRG(n, m, d) problem and provide an optimal $\Theta\left(\sqrt{dn}\right)$-time protocol that matches the lower bound for any $d \leq n$ and $m \leq (1 - \epsilon)\, n$, where $\epsilon > 0$ is an arbitrary small constant. Ours is the first nontrivial solution to the most general case of the MRG problem.

The paper is organized as follows. Section 2 describes an optimal deterministic protocol for the MRG$(n, m, 1)$ problem under the assumption $m \leq n/4$. In Section 3 the protocol is extended to handle the more general MRG(n, m, d) problem with $d \leq n$ and $m < (1 - \epsilon)\, n$. The section also proves the lower bound showing the optimality of the extended protocol. Some final conclusions and open problems are drawn in Section 4.

2 The Case of Single Targets

Consider an arbitrary instance of the MRG$(n, m, 1)$ problem, for $m \leq n/4$. The basic idea behind our protocol is to perform the routing through sorting, which is a typical strategy employed in the context of packet routing. However, we have to deal with the restrictive rules of an MRG system. In the following, we assume that at any time each robot knows the coordinates of its current location.

Let us consider the grid as partitioned into $n/4$ subgrids, called *tiles*, of size 2×2 each. The protocol has a simple high-level structure consisting of the four phases outlined below:

- *Phase I — Balancing:* The robots relocate in the grid so that each robot ends up in the top-left node of a distinct tile.
- *Phase II — Sorting-by-Row:* The robots sort themselves by their target row. The sorted sequence of robots is arranged on the grid (one robot per tile) according to the *Peano indexing* [Mor66] shown pictorially in Figure 1 and described mathematically later. In other words, at the end of the sorting, the i-th robot in the sorted order occupies the top-left corner of the tile of Peano index i.
- *Phase III — Permuting:* The sorted sequence of robots permutes from the Peano indexing to the row-major indexing.
- *Phase IV — Routing:* The robots first circulate within columns of tiles to reach the rows containing their targets. Then, they circulate around the rows to visit the targets.

Before describing the four phases in detail, we show how to perform some basic primitives in an MRG system which will be needed to implement the above phases.

`Pack` *Given $q \leq t$ robots on a t-node linear array, pack them into q consecutive nodes at one end of the array.*

Solution: Each robot repeatedly crosses an edge towards the designated end whenever its short-range sensor detects that the node across the edge is empty. No collisions arise in this way. Moreover, a simple argument shows that after t time steps all the robots have completed the packing. □

`Count` *Given $q \leq t$ robots on a t-node linear array, make q known to each robot.*

Solution: The robots first pack at one end of the array and then at the other. A robot that ends up at the i-th location from one end and at the j-th location from the other, sets $q = i + j - 1$. This primitive requires no more than $2t$ steps. □

`Compare-Swap` *Given a tile with two robots in it, sort the two robots so that the one associated with the smaller target row goes to the top left corner, while the other goes to the bottom left corner.*

Solution: Suppose that the two robots start at the top and bottom left corners of the tile. The robots execute a number of rounds until they "learn" their relative order in the sorted sequence. Specifically, in the i-th round, the robots "implicitly compare" the i-th most significant bit of the binary representation of their respective target row as follows. A robot positions itself at the left corner (in the same row) of the tile if its bit is 0, while it positions itself at the right corner if its bit is 1. Then each robot can infer the other robot's bit by simply checking for its presence in the same column. The first time that the robots find different bits, the robot whose bit is 0 moves to the top left corner, while the other moves to the bottom left corner, and the algorithm ends. If the robots have the same target row (i.e., all bits are equal) they stay in their starting positions. Overall, the computation takes no more than $\log n$ steps. $\square$

In the following subsections, we describe the four phases of our protocol in more detail.

2.1 Phase I: Balancing

In this phase, the $m \leq n/4$ robots start at arbitrary positions in the grid and must distribute themselves among the $n/4$ tiles so that each tile contains at most one robot in its top-left node. This is accomplished in $\log n - 2$ *balancing steps*, numbered from 0 to $\log n - 3$, according to the following inductive scheme. At the beginning of Step i, with i even, the robots are already distributed evenly among square subgrids of size $\sqrt{n/2^i} \times \sqrt{n/2^i}$ by induction. (This clearly holds for $i = 0$). During the step, the robots work independently within each square subgrid, and partition themselves evenly among rectangular subgrids of size $\sqrt{n/2^i} \times \sqrt{n/2^{i+2}}$. Analogously, in Step i with i odd, the robots work independently within each rectangular subgrid of size $\sqrt{n/2^{i-1}} \times \sqrt{n/2^{i+1}}$, and partition themselves evenly among square subgrids of size $\sqrt{n/2^{i+1}} \times \sqrt{n/2^{i+1}}$. Clearly, at the end of Step $\log n - 3$ the robots are evenly partitioned among the subgrids of size 2×2 (the tiles), with at most one robot per tile. At this point, each robot moves to the top-left corner of its tile.

We now describe the implementation of Step i, with i odd (the implementation of a balancing step of even index requires only minor modifications). Consider an arbitrary $t \times t/2$ rectangular subgrid, with $t = \sqrt{n/2^{i-1}}$, and suppose that there are p robots in the subgrid. Let the rows (resp., columns) of the subgrid be numbered from 1 to t (resp., $t/2$). At the end of the step we want to have $\lfloor p/2 \rfloor$ robots in the upper half (top $t/2$ rows) and the remaining $\lceil p/2 \rceil$ in the lower half (bottom $t/2$ rows) of the subgrid. This is done through the following substeps:

(1) The robots in each row pack towards the left.

(2) The robots in each column pack towards the bottom.

Comment: After this step, the robots form a "staircase" descending from northwest to southeast in the subgrid.

(3) In each column $k < t/2$, each robot determines the number of robots in the column. If this number is odd, the topmost robot (referred to as *leftover*) moves to the top of the column.

(4) All leftovers pack towards the right of the topmost row. Then they move down along column $t/2$ towards the bottom. Then, in column $t/2$, each robot determines the number of robots in the column.

Comment: If $p \leq t^2/4$ (which is always the case) then there is enough room in column $t/2$ to hold all leftovers.

(5) For every column k, let x be number of robots in the column after Step 4. (Note that x may be odd only for $k = t/2$.) If $k < t/2$, the robots pack around the column center, i.e., on rows $(t-x)/2+1, (t-x)/2+2, \ldots, (t+x)/2$. If $k = t/2$, the robots pack so that $\lfloor x/2 \rfloor$ of them end up in the upper half and the remaining $\lceil x/2 \rceil$ end up in the lower half.

Lemma 1 *Phase I takes $O(\sqrt{n})$ time.*

Proof: The correctness of the above strategy is immediate. The resulting time bound is a geometrically decreasing sum, whose i-th term is the cost $O\left(\sqrt{n/2^i}\right)$ of balancing step i, which is implemented in terms of the `Pack` and `Count` primitives presented before. $\square$

2.2 Phase II: Sorting-by-Row

At the end of the balancing phase, the robots are spread among the grid nodes in such a way that there is at most one robot in each tile, parked in the tile's top-left corner. The robots will now sort themselves according to their target row, with ties broken arbitrarily. The sorting algorithm relies upon a grid implementation of Batcher's bitonic sorting algorithm [Bat68] for sequences of size $n/4$ or smaller. We recall that Batcher's algorithm is structured as a cascade of $\log n - 2$ *merging stages*. At the beginning of the i-th merging stage, $1 \leq i \leq \log n - 2$, the robots are partitioned into $(n/4)/2^{i-1}$ sorted subsequences each of size 2^{i-1}. Then, pairs of subsequences are merged independently so that, at the end of the stage, there are $(n/4)/2^i$ sorted subsequences each of size 2^i. In turn, the i-th merging stage is made of a sequence of i steps, called (i,j)-*compare-swap* for $j = i-1, i-2, \ldots 0$. More specifically, an (i,j)-compare-swap step compares and swaps pairs of elements at distance 2^j in each subsequence (the direction of the compare/swap operator is fixed *a priori* and depends on the values of i and j).

In order to efficiently implement Batcher's algorithm on the grid, we number the $n/4$ tiles according to the *Peano indexing*, which can be defined as follows (see Figure 1). Split the set of indices $I = \{0, \ldots, n/4 - 1\}$ into four equally sized subsets of consecutive indices $I_0 = \{0, \ldots, n/16 - 1\}$, $I_1 = \{n/16, \ldots, n/8 - 1\}$, $I_2 = \{n/8, \ldots, 3n/16 - 1\}$, $I_3 = \{3n/16, \ldots, n/4 - 1\}$. Similarly, split the grid into four quadrants of $n/16$ tiles each, namely, $H_{t\ell}$, $H_{b\ell}$, H_{tr}, and H_{br}, where t stands for "top," b for "bottom," ℓ for "left," and r for "right." Assign the set of indices I_0 to $H_{t\ell}$, I_1 to $H_{b\ell}$, I_2 to H_{tr} and I_3 to H_{br}. Then proceed recursively within the quadrants until quadrants of one tile each are reached. An easy argument shows that two tiles with indices h and $h \oplus 2^j$ in the Peano indexing, where $\oplus$ denotes bitwise exclusive-or, lie on the same row or column of tiles (depending on the parity of j) at distance $O\left(\sqrt{2^j}\right)$ from each other.

An (i,j)-compare-swap step can be performed as follows. Let k denote any integer in $\{0, \ldots, n/4 - 1\}$ whose binary representation has 0 in position j. The following substeps are executed in parallel for all such values of k:

(1) The robot residing in tile $k + 2^j$ in the Peano indexing (if any) moves to tile k.

(2) The robots in tile k execute the `Compare-Swap` primitive according to their target row, with ties being broken arbitrarily. When only one robot is in the tile, it moves directly to the tile's top left corner.

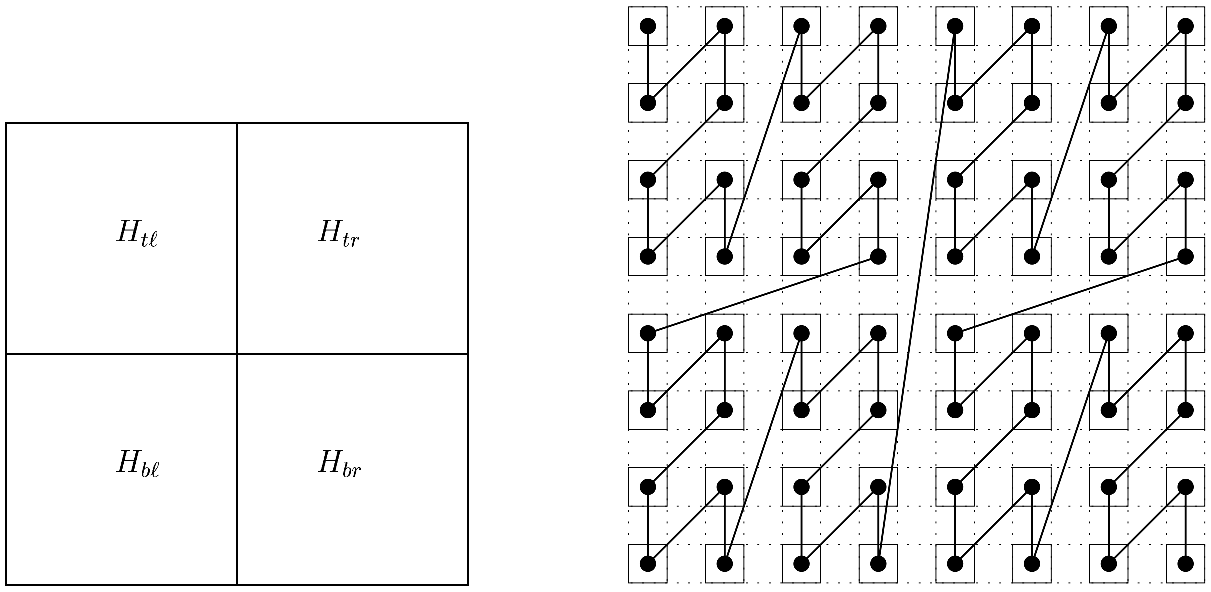

Figure 1: (Left) Logical partition of the grid into four quadrants. The Peano indexing for the whole grid is recursively defined as the concatenation of the Peano indexings for $H_{t\ell}$, $H_{b\ell}$, H_{tr}, and H_{br}, in that order. (Right) The 64 tiles of a 16×16 grid ordered according to the Peano indexing. Each square represents a tile with 2×2 grid nodes and contains one robot at most, after the balancing phase.

(3) The robot with the larger or smaller target (if any) moves to tile $k + 2^j$, depending on the direction of the (i, j)-compare-swap operator.

The routing implied by Step 1 above is easily performed by the robots without collisions. In particular, when j is odd, the robot in tile $k + 2^j$ first moves to the bottom-left corner of its tile, and then moves left until it reaches the bottom-left corner of tile k (which is on the same row as tile $k + 2^j$ by our numbering). When j is even, the robot in tile $k + 2^j$ first moves to the top-right corner of the tile and then moves upwards along the column, until it reaches the bottom-right corner of tile k. From there, it then positions itself at the bottom-left of the tile. Step 3 can be accomplished analogously. Thus, Steps 1 and 3 require $O\left(\sqrt{2^j}\right)$ time overall. By using the `Compare-Swap` primitive discussed before, Step 2 requires $O(\log n)$ time.

Lemma 2 *Phase II takes $O(\sqrt{n})$ time.*

Proof: The i-th merging stage of the sorting algorithm, $1 \leq i \leq \log n - 2$, consists of a sequence of (i, j)-compare-swap steps, for $j = i-1, i-2, \ldots, 1, 0$. As an (i, j)-compare-swap step takes $O\left(\sqrt{2^j} + \log n\right)$ time, the total running time of the algorithm is

$$T_{\mathrm{sort}}(n) = \sum_{i=1}^{\log n - 2} \sum_{j=0}^{i-1} O\left(\sqrt{2^j} + \log n\right) = O\left(\sqrt{n}\right).$$

□

2.3 Phase III: Permuting

After the sorting phase, the robots reside in distinct tiles, sorted by target row according to the Peano indexing. In Phase III, the robots permute in such a way that the sorted sequence is rearranged according to the row-major indexing. Let us call *t-column* (resp., *t-row*) a column (resp., row) of tiles. The permutation is executed according to the following recursive protocol. If $n = 4$, the permutation is trivial. Consider $n > 4$.

(1) Each robot in H_{tr} swaps positions with the one occupying the corresponding position in $H_{b\ell}$.

(2) Within each quadrant, the sorted subsequence of robots recursively permutes from Peano to row-major indexing.

(3) Within each quadrant, the robots permute so that those in odd t-rows pack to the top, while those in even t-rows pack to the bottom of the quadrant.

(4) Each robot in the lower half of $H_{t\ell}$ (resp., $H_{b\ell}$) swaps positions with the one occupying the corresponding position in the top half of H_{tr}, (resp., H_{br}).

The correctness of the permutation protocol is easily established by induction. Below, we give a pictorial illustration of the steps for $m = 64$ robots:

Initial configuration

1	3	9	11	33	35	41	43
2	4	10	12	34	36	42	44
5	7	13	15	37	39	45	47
6	8	14	16	38	40	46	48
17	19	25	27	49	51	57	59
18	20	26	28	50	52	58	60
21	23	29	31	53	55	61	63
22	24	30	32	54	56	62	64

After Step 1

1	3	9	11	17	19	25	27
2	4	10	12	18	20	26	28
5	7	13	15	21	23	29	31
6	8	14	16	22	24	30	32
33	35	41	43	49	51	57	59
34	36	42	44	50	52	58	60
37	39	45	47	53	55	61	63
38	40	46	48	54	56	62	64

After Step 2

1	2	3	4	17	18	19	20
5	6	7	8	21	22	23	24
9	10	11	12	25	26	27	28
13	14	15	16	29	30	31	32
33	34	35	36	49	50	51	52
37	38	39	40	53	54	55	56
41	42	43	44	57	58	59	60
45	46	47	48	61	62	63	64

After Step 3

1	2	3	4	17	18	19	20
9	10	11	12	25	26	27	28
5	6	7	8	21	22	23	24
13	14	15	16	29	30	31	32
33	34	35	36	49	50	51	52
41	42	43	44	57	58	59	60
37	38	39	40	53	54	55	56
45	46	47	48	61	62	63	64

After Step 4

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Lemma 3 *Phase III takes $O(\sqrt{n})$ time.*

Proof: The movements of robots implied by Step 1, Step 3 and Step 4 can be executed in a conflict-free fashion in $O(\sqrt{n})$ time as one robot at most is in each tile. Since the recursive Step 2 is executed in parallel and independently within subgrids of geometrically decreasing side, we conclude that the overall permutation time is also $O(\sqrt{n})$. □

2.4 Phase IV: Routing

The routing phase starts with the m robots sorted by target row and occupying the first m tiles in the row-major indexing, with at most one robot per tile (in the tile's top-left corner). We number the t-columns (resp., the t-rows) from 1 to $\sqrt{n}/2$. Note that, due to sorting, each t-column holds no more than two robots with targets in the same row. The routing is performed by first moving the robots to their target row and then to their final target. This is accomplished in parallel as follows:

(1) For $1 \le i \le \sqrt{n}/4$ and $1 \le j \le \sqrt{n}/2$, the robot residing in t-column $2i$ and t-row j moves to the top-right corner of the tile in t-column $2i-1$ and t-row j.

Comment: After this step, in any odd-numbered t-column there can be up to four robots destined for the same row, while the even-numbered t-columns are empty.

(2) The robots in each odd-numbered t-column circulate along a directed Hamiltonian cycle traversing all of the nodes in the t-column. When a robot traveling on the right side of the t-column reaches its target row, it attempts to shift right to the adjacent tile and then moves to the rightmost unoccupied node in such tile.

    Comment: Within an odd-numbered t-column, no more than two robots with the same target row are able to move to the adjacent t-column.

(3) The robots in each t-row circulate along a directed Hamiltonian cycle traversing all the nodes in the t-row, therefore visiting their target locations.

(4) All the robots go back to the t-columns they occupied at the end of Step 1.

(5) Steps 2–3 are repeated to deliver the robots that have not visited their targets yet. To this end, the robots that have already completed their task will not attempt to shift right during Step 2.

    Comment: All robots that have not visited their targets at the beginning of Step 5 are now able to do so.

Lemma 4 *Phase IV takes $O(\sqrt{n})$ time.*

Proof: Steps 1–3 require $O(\sqrt{n})$ time altogether and are executed at most twice each (due to Step 5). Step 4 can be executed as follows. In each odd-numbered t-column, the robots in each row pack to the left. Then, robots in each even-numbered t-column circulate along a directed Hamiltonian cycle traversing all the nodes in the t-column, and when a robot sees an empty spot in the adjacent t-column (to the left) it moves into such a spot packing to the left. Thus, Step 4 requires $O(\sqrt{n})$ time. This implies that the whole routing phase also takes $O(\sqrt{n})$ time. □

The following theorem is an immediate consequence of Lemmas 1, 2, 3 and 4.

Theorem 1 *Any instance of the MRG$(n, m, 1)$ problem, with $m \leq n/4$, can be solved in time $O(\sqrt{n})$ in the worst case.*

The simple diameter-based argument shows that the running time stated in the above theorem is optimal. Moreover, the result is easily extended to the case in which $m' \leq m$ robots have one target to reach, while the $m - m'$ remaining ones do not have any visit to perform. It is sufficient to associate the latter robots with a fictitious destination whose row is $\sqrt{n}+1$ and let them participate to the various phases of the protocol. Clearly, there is no increase in the running time.

3 The Case of Multiple Targets

In this section we devise a protocol for the more general MRG(n, m, d) problem where each of m robots needs to visit up to d grid nodes, with each grid node being visited by at most one robot. The protocol is first presented for the case $m \leq n/4$, and then extended to handle up to $m \leq (1-\epsilon)\,n$ robots, for any small constant $\epsilon > 0$. Before describing the protocols, we prove a lower bound on the running time of any protocol for the MRG(n, m, d) problem. The lower bound will be employed later to show the optimality of the proposed protocols.

Lemma 5 *For every choice of integers n, m, d, with $1 \leq m, d \leq n$, there exists an instance of the MRG(n, m, d) problem whose solution requires $\Omega\left(\sqrt{dn}\right)$ time.*

Proof: If $n < 4$ or $d < 4$, the bound follows from the diameter argument. Therefore, let us examine the case $n, d \geq 4$. Let n', d' be the largest powers of 4 such that $n' \leq n$ and $d' \leq d$. Note that $n' \geq n/4$, $d' \geq d/4$ and $d' \leq n'$. Consider a square subgrid of n' nodes, partitioned into d' square tiles of size $\sqrt{n'/d'} \times \sqrt{n'/d'}$, and suppose that one of the m robots has the d' centers of the tiles among its targets, where the center of a tile is the node in the $(\sqrt{n'/d'}/2)$-th row and in the $(\sqrt{n'/d'}/2)$-th column of the tile. In order to visit its targets the robot must traverse at least $\sqrt{n'/d'}/2$ nodes in each of $d'-1$ subgrids or more, for an overall time requirement of $\Omega\left(d'\sqrt{n'/d'}\right) = \Omega\left(\sqrt{dn}\right)$. □

3.1 An Optimal Protocol for $m \leq n/4$

Consider an instance of the MRG(n, m, d) problem with $d \leq n$ and $m \leq n/4$, and let $k = \lfloor \log_4 d \rfloor$. The protocol is structured as a sequence of $k+1$ *stages* of geometrically increasing running time. For $0 \leq i \leq k$, in stage i all robots having to visit at least 4^i and less than 4^{i+1} destinations, accomplish their task. We call such robots *active* in stage i, whereas the remaining robots are called *inert* in stage i. Note that since all robots reside on the grid at all times, the protocol must orchestrate the movement for both active and inert robots in every stage.

Stage 0 and stage 1 are executed by simply running the single-target protocol fifteen times, one for every possible target of each robot active in these stages. Clearly, inert robots will participate in the protocol by associating themselves to "fake" destinations, as described at the end of Section 2.

In stage i, $2 \leq i \leq k$, at most $n/4^i$ robots are active. Let $\delta_i = n/4^{i-1}$ and regard the grid as being conceptually partitioned into 4^{i-1} square subgrids of δ_i nodes each, which we refer to as δ_i-*tiles*. Observe that all robots active in this stage fit in one quadrant of a δ_i-tile. Stage i is executed in two *rounds*. In the first round, the inert robots pack in the lower half of the grid, while the active robots tour all δ_i-tiles in the upper half of the grid, stopping in each δ_i-tile for a time sufficient to visit all of their destinations in the tile, and progressively accumulating in the first tile of the lower half as they complete their tour. Different robots may stop in a δ_i-tile for different amounts of time, depending on the number of their targets in the tile. Similarly, in the second round the inert robots pack in the upper half of the grid while the active robots visit their destinations in the lower half.

We describe in detail the operations performed by the robots in the first round of stage i, omitting the description for the second round, which is virtually identical. For $0 \leq j < 4^{i-1}$, let T_j denote the j-th δ_i-tile based on a snake-like ordering of the δ_i-tiles which proceeds alternatively left-to-right and right-to-left. Note that the δ_i-tiles in the upper half of the grid are those of indices $j < 4^{i-1}/2$.

(1) The robots relocate in the grid so that all active robots end up in the first tile T_1, while the inert robots pack to the bottom tiles T_j with $4^{i-1}/2 + 1 \leq j < 4^{i-1}$.

Comment: Tile $T_{4^{i-1}/2}$ is left empty to collect the active robots having completed their task in the upper tiles.

(2) The following sequence of substeps is repeated $17 \cdot 4^{i-1}$ times, in parallel for each index j, $0 \leq j < 4^{i-1}/2$:

(2.a) Each active robot with unvisited targets in T_j visits one arbitrary such target.

(2.b) Robots that visited all of their targets in T_j move to the top-left quadrant of T_j, while each robot that has still some unvisited targets in T_j moves to the bottom-right quadrant of T_j.

(2.c) Robots in the top-left quadrant of T_j move to the top-left quadrant of T_{j+1}.

(2.d) The robots in tile $T_{4^{i-1}/2}$, including those newly arrived in it, pack to the tile's bottom-right quadrant.

Lemma 6 *For $2 \leq i \leq k$, stage i is correct and is completed in $O\left(\sqrt{4^i n}\right)$ time.*

Proof: Fix some i, $2 \leq i \leq k$, and consider the first round of stage i (the argument for the second round is identical). Note that, since $i \geq 2$, the δ_i-tiles T_j, with $4^{i-1}/2+1 \leq j < 4^{i-1}$, comprise $\delta_i(4^{i-1}/2 - 1) \geq n/4$ grid nodes altogether, hence all inert robots fit in such tiles. Step (1) is easily accomplished in $O(\sqrt{n})$ time through the balancing and sorting techniques described in Section 2. As for Step (2), we discuss its four substeps in detail. Substep (2.a) is executed independently within δ_i-tiles and entails one execution of the single-target protocol of Section 2, which takes $O\left(\sqrt{\delta_i}\right)$ time. Substeps (2.b), (2.c) and (2.d) can be executed through balancing and sorting within δ_i-tiles and simple relocations between adjacent δ_i-tiles, in time $O\left(\sqrt{\delta_i}\right)$. The correctness of these substeps follows from observing that all robots active in stage i fit in one quadrant of a δ_i-tile. Consequently, at any time there cannot be more than $\delta_i/4$ robots in each tile T_j, with $0 \leq j \leq 4^{i-1}/2$, and that at the beginning of Substep (2.d) all robots in $T_{4^{i-1}/2}$ can be packed into the tile's bottom-right quadrant.

It remains to show that $17 \cdot 4^{i-1}$ iterations of Step (2) are sufficient for each active robot to visit its targets in the upper half of the grid. Consider an active robot and let d_j be the number of its targets in T_j, for $0 \leq j < 4^{i-1}/2$. The robot will stay in such a tile for $\max\{1, d_j\}$ iterations, hence the total number of iterations needed to visit all of its targets is

$$\sum_{j=0}^{4^{i-1}/2-1} \max\{1, d_j\} < \frac{4^{i-1}}{2} + 4^{i+1} < 17 \cdot 4^{i-1}.$$

Thus, Step (2) takes $O\left(4^i\sqrt{\delta_i}\right) = O\left(\sqrt{4^i n}\right)$ time overall. Since the complexity of Step (2) dominates that of Step (1), it is also the running time for the entire round. □

The complexity of the protocol follows from Lemma 5 and Lemma 6.

Theorem 2 *Any instance of the MRG(n, m, d) problem with $m \leq n/4$ and $d \leq n$ can be solved in optimal $\Theta\left(\sqrt{dn}\right)$ time, in the worst case.*

Proof: Stage 0 and stage 1 can be correctly performed in $\Theta(\sqrt{n})$ time by Theorem 1. The correctness and complexity of stage i, for $1 \leq i \leq k$, is established in Lemma 6. The total cost is therefore $O\left(\sqrt{n}+\sum_{i=1}^{k}\sqrt{4^i n}\right) = O\left(\sqrt{dn}\right)$, and the optimality follows from Lemma 5. $\square$

3.2 Extension to $m \leq (1-\epsilon)\, n$ Robots

Let ϵ be an arbitrary constant, $0 < \epsilon < 1$, and consider an instance of the MRG(n, m, d) problem with $m \leq (1-\epsilon)\, n$ robots. Let c be the smallest power of 2 larger than or equal to $2/\epsilon$, and let $\delta = n/c^2$. We regard the grid as conceptually partitioned into c^2 δ-tiles, each of size δ. For $0 \leq i < c^2$, we denote by T_i the i-th δ-tile along a predetermined Hamiltonian cycle of the δ-tiles, with T_0 being the δ-tile at the northeast corner of the grid (it is easy to see that such a cycle exists, since c is even). The protocol is organized as follows. Initially, the robots pack southwest on the grid leaving T_0 empty (see Figure 2). Then such empty δ-tile is "slid" around the grid occupying, in turn, every position along the Hamiltonian cycle. When a δ-tile T_{i-1} becomes the empty tile, the robots in T_i move into T_{i-1} in four batches of one quadrant each at a time. Once in T_{i-1}, the (at most) $\delta/4$ robots of a batch visit their targets in the tile by employing the protocol from the previous subsection. By repeatedly shifting all robots along the Hamiltonian cycle and repeating the above operations, all robots initially in T_i are able to visit all of their targets in the grid.

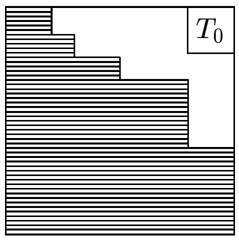

Figure 2: Configuration of robots after Step (1)

More formally, the protocol consists of the following steps.

(1) The robots first pack towards the left in each row and then towards the bottom in each column, thus ending in a "staircase" configuration descending from northwest to southeast in the grid (see Figure 2). Let $R_{i,k}$ denote the robots that occupy the k-th quadrant of T_i at the end of this step, for $0 \leq i < c^2$ and $1 \leq k \leq 4$.
Comment: At the end of Step (1), T_0 is empty.

```
(2) For every $1 \le i < c^2$ do
    (2.1) For $0 \le j < c^2$ do
        (2.1.a) For each $k$, $1 \le k \le 4$, the robots in $R_{i,k}$ move from $T_{(i+j) \bmod c^2}$ to $T_{(i-1+j) \bmod c^2}$, visit their destinations therein using the protocol from Section 3.1, and return to $T_{(i+j) \bmod c^2}$.
        (2.1.b) All robots shift by one $\delta$-tile forward along the Hamiltonian cycle, maintaining the same relative positions within the $\delta$-tile.
    (2.2) All robots in $T_i$ move to $T_{i-1}$, maintaining the same relative positions within the $\delta$-tile.
```

Theorem 3 *Any instance of the MRG(n, m, d) problem with $m \le (1-\epsilon)\, n$ and $d \le n$ can be solved in optimal $\Theta\left(\sqrt{dn}\right)$ time, in the worst case.*

Proof: We begin by showing the correctness of the protocol. First, note that at the end of Step (1) T_0 must be empty, or otherwise there would be more than

$$\left(\sqrt{n} - \sqrt{\delta}\right)^2 > \left(1 - \frac{2}{c}\right) n \ge (1-\epsilon)\, n$$

robots on the grid, which is a contradiction, since $m \le (1-\epsilon)\, n$. Let $R_i = \cup_{k=1}^4 R_{i,k}$, for $0 \le i < c^2$. An easy inductive argument shows that for every $1 \le i < c^2$ and $0 \le j < c^2$, at the beginning of Iteration j of Substep (2.1), within Iteration i of Step (2), all robots in R_i reside in $T_{(i+j) \bmod c^2}$, and $T_{(i+j-1) \bmod c^2}$ is empty. As a consequence, in Iteration i of Step (2) all robots in R_i are able to visit all of their targets, hence the protocol is correct. Let us now analyze the running time. Step (1) is easily executed in $O\left(\sqrt{n}\right)$ time. Time $O\left(\sqrt{n}\right)$ is also sufficient for every execution of Substeps (2.1.b) and (2.2), while, from Theorem 2, each execution of Substep (2.1.a) takes $O\left(\sqrt{dn}\right)$ time. The overall protocol's running time claimed in the theorem follows by noting that since $c = O(1)$ both Step (2) and Substep (2.1) are repeated a constant number of times. □

4 Conclusions and Open Problems

We studied the complexity of moving a set of m robots with limited sensory capabilities, in a multi robot grid system of size $\sqrt{n} \times \sqrt{n}$. We provided an $O\left(\sqrt{dn}\right)$ deterministic protocol that governs the movement of $m \leq (1-\epsilon)\, n$ robots for any small constant $\epsilon > 0$, where each robot may visit up to $d \leq n$ distinct locations, but no two robots visit the same location (MRG(n, m, d) problem). We also proved a lower bound showing that our protocol is optimal. Our investigation leaves open the problem of studying the complexity of moving $m = n - o(n)$ robots. It would be interesting to extend the protocol to achieve optimality in such extreme scenario. Clearly, our protocol could be employed to this end if the rules governing the system were relaxed so as to allow a robot to stay initially outside the grid for an arbitrary amount of time, and to disappear from the grid as soon as it visits its target, which is a tacit assumption made by the protocols in [PU96]. Finally, another interesting open problem concerns the extension of the protocol to allow distinct robots to visit the same location. To the best of our knowledge, no result is known for this setting, except for the trivial $O(n)$-time protocol based on a Hamiltonian tour of the grid nodes.

Acknowledgments The authors would like to thank Elena Lodi, Fabrizio Luccio, Linda Pagli and Ugo Vaccaro for many interesting discussions and comments during the early stages of this work, and the referees, who provided a number of useful suggestions. Some of these suggestions lead us to the results described in Section 3.2.

References

[Bat68] Batcher, K.E. (1968), Sorting networks and their applications, *in* "Proceedings, AFIPS Spring Joint Computer Conference," pp. 307–314.

[Lei92] Leighton, F.T. (1992), "Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes," Morgan Kaufmann, San Mateo, CA.

[Mor66] Morton, G. (1966), "A computer oriented geodetic data base and a new technique in file sequencing," IBM Ltd. Internal Report.

[NS95] Newman, I., and Schuster, A. (1995), Hot-potato algorithms for permutation routing, *IEEE Trans. Parallel and Distribut. Comput.* **PDC-6**, 1168–1176.

[PU96] Preminger, S., and Upfal, E. (1996), Safe and efficient traffic laws for mobile robots, in "Proceedings, 5th Scandinavian Workshop on Algorithm Theory," (R.

Karlsson and A. Lingas, Eds), pp. 356–367, Springer-Verlag LNCS 1097, Berlin, Germany.

[Sib95] Sibeyn, J.F. (1995), "Overview of mesh results," Technical Report MPI-95-1-018, Max-Planck Institut für Informatik, Saarbrücken, Germany.

[ST92] Shoham, Y., and Tennenholtz, M. (1992), On traffic laws for mobile robots, *in* "Artificial intelligence planning systems: Proceedings of the first international conference," pp. 231–252, Morgan Kaufmann, San Mateo, CA.

[ST95] Shoham, Y., and Tennenholtz, M. (1995), On social laws for artificial agent societies: Off-line design, *Artificial Intelligence* **73**, 231–252.

Figure Captions

Figure 1: (Left) Logical partition of the grid into four quadrants. The Peano indexing for the whole grid is recursively defined as the concatenation of the Peano indexings for $H_{t\ell}$, $H_{b\ell}$, H_{tr}, and H_{br}, in that order. (Right) The 64 tiles of a 16×16 grid ordered according to the Peano indexing. Each square represents a tile with 2×2 grid nodes and contains one robot at most, after the balancing phase.

Figure 2: Configuration of robots after Step (1).