Strategies for Parallel Unaware Cleaners

Christian Ortolf and Christian Schindelhauer

University of Freiburg, Department of Computer Science, Computer Networks
{ortolf,schindel}@informatik.uni-freiburg.de

Abstract. We investigate the parallel traversal of a graph with multiple robots unaware of each other. All robots traverse the graph in parallel forever and the goal is to minimize the time needed until the last node is visited (*first visit time*) and the time between revisits of a node (*revisit time*). We also want to minimize the *visit time*, i.e. the maximum of the first visit time and the time between revisits of a node. We present randomized algorithms for uncoordinated robots, which can compete with the optimal coordinated traversal by a small factor, the so-called *competitive ratio*.
For ring and path graph simple traversal strategies allow constant competitive factors even in the worst case. For grid and torus graphs with n nodes there is a $\mathcal{O}(\log n)$-competitive algorithm for both visit problems succeeding with high probability, i.e. with probability $1 - n^{-\mathcal{O}(1)}$. For general graphs we present an $\mathcal{O}(\log^2 n)$-competitive algorithm for the first visit problem, while for the visit problem we show an $\mathcal{O}(\log^3 n)$-competitive algorithm both succeeding with high probability.

Keywords: visit time, competitive analysis, mobile agent, robot, multi-robot graph exploration

1 Introduction

Today, we are used to robotic lawn mowers and robotic vacuum cleaning. The current best-selling technology relies on robots which have no communication features and in some cases use maps of the environment. If we model the environment as an undirected graph, then its traversal by a single robot is an NP-hard minimum Traveling Salesman problem, for which efficient constant factor approximation algorithms are known [3]. Now, the robot owner deploys additional robots. How well do these robots perform? Can we guarantee that two parallel unaware lawn mowers will cut all grass better than one? And how do they compare to a couple of perfectly choreographed mowers? What about more robots, where each robot has no clue how many co-working devices exist nor where they are?

Here, we investigate these questions. We model the cleaning area by a graph with identifiable nodes and edges. All robots know only their own position and the graph. They will never learn how many robots are involved, nor any other robots' positioning data. So, we assume that robots pass each other on the same node without noticing. We are looking for a traversal strategy of the graph which is self-compatible, since we assume that all robots are clones performing the same strategy.

It appears apparent that such a strategy must be probabilistic, since robots starting from the same node would otherwise follow identical routes, which would not allow for any speedup. However, we will see that this is not the case for all graphs.

Related Work To our knowledge this unaware parallel cleaning model is new, therefore we will point out similarities to other problems.

The parallel unaware cleaning can be seen as a variation of the multi robot exploration [4, 6, 8, 5, 15]. The goal of the online multi-robot exploration is to steer a group of robots to visit every node of an unknown graph. The term *unknown* means that the exploring algorithm knows only edges adjacent to formerly visited nodes. The performance of such online algorithms is usually provided by a competitive analysis comparing the online solution to the optimal offline strategy, where an algorithm is given knowledge of the whole graph beforehand. This model is close to our first visit time model with two important differences: In parallel unaware cleaning each robot knows the full graph, while multi-robot exploration robots know only the explored graph. In our model there is no communication, while in robot exploration robots exchange their graph information.

It was recently shown that if more than dn robots are used in the multi-robot exploration problem, where d is the diameter of the graph and n the number of nodes, then one can achieve a constant competitive factor for multi-robot exploration [4]. The competing offline exploration can explore a graph in time $\Theta(\frac{n}{k} + d)$, therefore an exploration using $k = \frac{n}{d}$ robots is of special interest, because it allows the offline algorithm to make full use of all its robots.

For this scenario Dynia et al. [6] showed the online exploration of trees to be at best $\Omega(\frac{\log k}{\log \log k})$-competitive. If algorithms are restricted to greedy exploration an even stronger bound of $\Omega(k/\log k)$ is shown by Higashikawa et al. [11]. This bound matches the best known upper bound by Fraigniauds et al.'s greedy exploration algorithm in [8]. For further restricted graphs better bounds have been shown. An algorithm depending on a *density* parameter p was presented by Dynia et al. [5] with $\mathcal{O}(d^{1-1/p})$ competitiveness, e.g. $\mathcal{O}(d^{1/2})$ for trees embeddable in grids. For grids with convex obstacles, an polylogarithmic competitive bound of $\mathcal{O}(\log^2 n)$ was shown in [15], along with the lower bound of $\Omega(\frac{\log k}{\log \log k})$ matching the identical lower bound for trees.

Our problem also bears resemblance to the multi traveling salesman problem (mTSP) [2, 9], a generalization of the well-known traveling salesman problem (TSP) [12]. TSP is NP-hard even in the seemingly simpler Euclidean version [16], but can be efficiently approximated if it is allowed to visit nodes more than once [18].

The mTSP trys to cover the graph with a set of tours and minimize the length of the longest tour. This corresponds to the offline parallel cleaning problem, if we use the distance between nodes in the graph as cost measure between nodes in mTSP. Even if salesmen start at different nodes the problem can still be reduced to the regular mTSP [10].

A similar definition to our first visit time is the notion of *cover time* for random walks, likewise visit time can be compared to the *hitting time* $H(i, j)$, the expected time starting from node i to reach node j. Our robots are not forced to use random walks. So, the Lollipop graph, a lower bound construction for the cover time of $\Omega(n^3)$ [13] and obtained by joining a complete graph to a path graph with a bridge, can be cleaned quite efficiently by parallel unaware cleaners.

Patrolling algorithms [17] also require robots to repeatedly visit the same area. To the best of our knowledge no work there has similarly restricted robots.

2 Model

In our model k robots are initially positioned on depot/starting nodes $S = (s_1, ..., s_k)$ and their task is to visit all nodes V of an undirected connected graph $G = (V, E)$ and then repeat their visits as fast as possible. Nodes and edges can be identified and time is measured in rounds. An algorithm has to decide for each robot r in each round which edge to traverse to visit another node in the following round. This decision is based on the starting node s_r, the graph and the previous decisions of the robot. Each robot never learns the number and positions of other robots.

The **first visit time of a node** is the number of the round, when a robot visits this node for the first time. The **visit time of a node** is the supremum of all time intervals between any two visits of a node (revisit) including the time interval necessary for the first visit by any robot. The **long term visit time of a node** is the supremum of time intervals between any two visits of a node by any robot after an arbitrarily long time. The corresponding definitions for the full graph is given by the maximum (first/long term) visit time of all nodes. Note that the robots do neither know and nor are they able to compute the visit times. These times can only be known by an external observer.

The term **with high probability** refers to an event which occurs with probability $1 - n^{-c}$ with constant $c \geq 1$. In all of our results, this constant c can be arbitrarily increased if one allows a larger constant factor for the run-time.

The term **distance** refers to the number of edges on a shortest path between two nodes.

The benchmark for our solution is the time of an algorithm with full knowledge, i.e. the number and positions of all robots. The quotient between the unaware visit time and the full knowledge visit time is our measure, also known as the competitive factor. The worst case setting can be seen as an adversary placing the robots for a given algorithm.

3 Simple cleaning examples

As an illustration and starting example we show how differently a circle graph and a path graph behave in this setting, see Fig. 1 and Fig. 2. The simple algorithm sending robots in one direction in the circle, or just to one end on the line, then returning to the other end, performs quite differently for both graphs.

On the circle the right traversal strategy performs very well, the first visit time may be improved by a knowing algorithm at most by a factor of 2, since the largest distance r between two robots at the beginning lowerbounds the optimal offline strategy by $r/2$. The deterministic right traversal strategy on the cycle visits all nodes in r rounds for the first round and revisits them in this frequency thereafter.

For the path graph, the overhead of such an algorithm is a factor of n. If one end node is not covered and all robots walk first to the right end and then return, no robot can visit the left node in less than n rounds. A smarter oblivious algorithm could improve

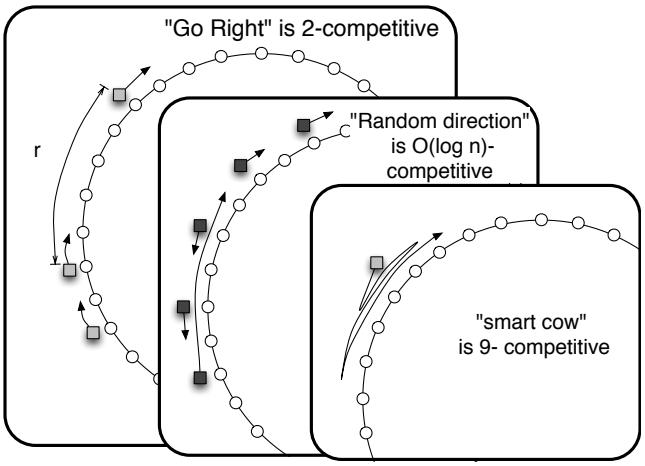

Fig. 1. Parallel unaware cleaning algorithms for the cycle graph. Illustrating competitive ratio for first visit.

this by sending robots into a random direction instead, yielding a competitive factor of $\mathcal{O}(\log n)$ in the expectation. However, a deterministic solution exists: the smart cow algorithm [1], which in the i-th phase for $i = 1, 2, \ldots, n$ explores 2^i nodes first to the left and then 2^i nodes to right from the starting node. While the smart cow algorithm is designed to find a hole in a fence, which it does within a competitive factor of nine, the same competitive factor can be shown for the cycle and the path graph. This shows that for these simple graphs deterministic competitive visiting strategies exist.

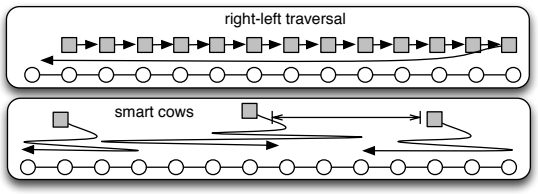

Fig. 2. Parallel unaware cleaning algorithms for the Path graph

However, for the long term visit problem the situation is different. Symmetry cannot be resolved by any deterministic algorithm. If all robots have the same starting node no competitive ratio better than $\mathcal{O}(n)$ can be achieved for these algorithms. The following chapter shows a simple solution to the long term visit problem.

4 Canonical cleaning and general observations

Now we present first general strategies and techniques. For $u \in V$ let $N_\ell(u)$ denote the set of nodes in G within distance of at most ℓ to the node u. For a set $A \subseteq V$ let $N_\ell(A) = \bigcup_{u \in A} N_\ell(u)$. The following lemma is the key technique, which provides a lower bound for the number of robots in the vicinity.

Lemma 1. *Given a graph with a robot placement with a first visit time of t_f. Then, for any set of nodes A the number of robots in the node set $N_\ell(A)$ is at least $\lceil |A|/(t_f+1) \rceil$ for $\ell \geq t_f$.*

Proof. First note that for each cleaning strategy it is not possible that robots outside of $N_{t_f}(A) \subseteq N_\ell(A)$ can reach any node within A in at most t_f steps. Let k be the number of robots that explore A within time t_f. At the beginning at most k nodes can be occupied by k robots. Then, in every subsequent round at most k additional nodes of A can be visited. In order to visit all nodes in A we have $k(t_f+1) \geq |A|$. This implies $k \geq \frac{|A|}{t_f+1}$.

Later on, we use this lemma in a bait-and-switch strategy. We use A as bait to ensure that enough robots exist in a region for the offline strategy. Then we switch and let these robots work on other areas.

While randomization is necessary for dispersing the robots, too many probabilistic decisions are problematic, because the chance that some nodes remain unvisited for long times may grow over time. Therefore, we present only algorithms that use a finite number of randomized decisions. This technique is presented in the canonical algorithm, which is the base for some of our strategies. It requires the algorithms *cycle-start-node* and *waiting time* to provide where and when the robot should start cycling the graph.

Algorithm 1: CANONICAL CLEANING algorithm for robot r using algorithms cycle-start-node and waiting-time

```
Traverse the graph by DFS yielding a cycle P with V(P) = V of length 2n
v_s ← cycle-start-node(s_r)
Move robot r on the shortest path to v_s
w ← waiting time(s_r, v_s)
Wait w rounds
if v_s occurs more than once in P then
    Choose a random occurrence in P
end
while true do
    Walk to the next node of P
end
```

Because of the coupon collector's problem, a basic problem of probability theory [14], one cannot expect a better competitive factor than $\mathcal{O}(\log n)$. Therefore, in the long run the problem can be solved by the canonical algorithm.

Theorem 1. *Using the* CANONICAL CLEANING *it is possible to achieve a long-term visit time of $\mathcal{O}((n/k)\log n)$ and a visit time of diameter$(G) + \mathcal{O}((n/k)\log n)$ with high probability.*

We refer to the Appendix A.1 for the proof.

For graphs with small diameter this results in a logarithmic competitive ratio. E.g. in balanced trees the diameter is bounded by $\mathcal{O}(\log n)$. So, the CANONICAL CLEANING algorithm gives us the following bound.

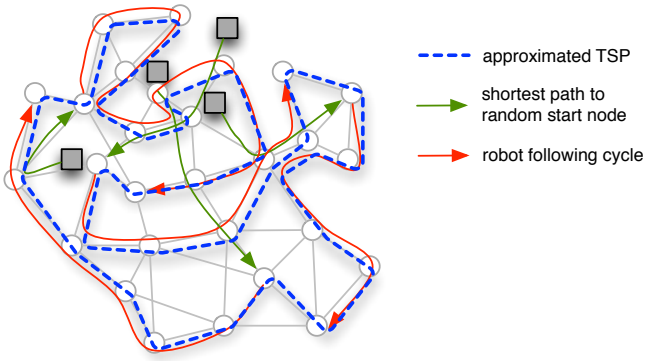

Fig. 3. A canonical algorithm guarantees a $\mathcal{O}(\frac{n}{k}\log n)$ long-term visit time.

Corollary 1. *Graphs with diameter of $\mathcal{O}(\log n)$ have a competitive ratio of $\mathcal{O}(\log n)$ for the first and revisit visit time with high probability.*

Proof. Let *cycle-start-node*(u) map to a uniform random node v of the tree. And let *waiting-time*$(u, v) =$ *diameter*$(G) - |u, v|$. Let t_f^* and t_v^* be the optimal first and visit times and let $k \leq n$ be the number of robots.

Theorem 1 states that the first visit and visit time is bounded by *diameter*$(G) + \mathcal{O}((n/k)\log n) = \mathcal{O}(\log n + (n/k)\log n) = \mathcal{O}((n/k)\log n)$. From Lemma 1 it follows for $A = V$ that $t_f^* \geq n/k - 1$ and $t_v^* \geq n/k$. This implies a competitive ratio of $\mathcal{O}(\log n)$ for $k \leq n$. If $t_f^* > 0$ it also holds for $k \geq n$. In the case of $t_f^* = 0$, the robots already cover all nodes and every algorithm is optimal for the first visit time.

Another interesting technique is to transform a probabilistic first visit time strategy into a visit time algorithm succeeding with high probability. The only drawback is, that the first visit time and the visit probability for all nodes must be known.

Lemma 2. *Assume there exists a parallel unaware cleaner algorithm $\mathcal{A}$ for k robots on a graph with n nodes, where for all nodes u the probability that the first visit time is less or equal than t_f is at least $p > 0$. Furthermore, t_f and p are* **known**. *Then, this cleaning algorithm can be transformed into a canonical algorithm having visit time $\mathcal{O}(\frac{1}{p}t_f \log n)$ with high probability.*

The proof sketch is the following. Let $P(r)$ with $|P(r)| \leq t_f$ be the resulting path of robot r performing algorithm $\mathcal{A}$. Then, the *cycle-start-node* of the canonical algorithm is defined by choosing a random uniform node v_s from $P(r)$. We use *waiting-time*$(s_r, v_s) = 0$. In the Appendix A.2 a detailed proof is given.

5 The Torus and the Grid Graph

Now we consider torus and grid graphs, where we present optimal unaware cleaner strategies.

Define a $m \times m$-Torus $G_T = (V, E_T)$ graph by $V = [0, \ldots, m-1] \times [0, \ldots, m-1]$ and with edges $\{(i,j), (i+1 \bmod m, j)\}$ and $\{(i,j), (i, j+1 \bmod m)\}$ for $(i,j) \in V$. Every node has four neighbors, where we call the directions *right*, *left*, *up*, and *down* in the standard way. Parallel unaware robots can clean the torus graph with only a small overhead.

Algorithm 2: Competitive torus cleaner strategy for robot r

```
(x, y) ← (s_{r.x}, s_{r.y}) starting position
for i ← 1, 2, ..., √n do
    if random event occurs with probability (x − s_{r.x} + 1)/(i + 1) then
        x ← x + 1
    else
        y ← y + 1
    end
    Move to (x, y)
end
H := cycle of Fig. 5.
while true do
    Move to the next node of H
end
```

Theorem 2. *Algorithm 2 is a high probability $\mathcal{O}(\log n)$-competitive visit algorithm for the $m \times m$-torus graph.*

We refer to the Appendix A.3 for analysis of Algorithm 2.

The first technique, the for loop of Algorithm 2, is that the cleaner uses a probabilistic process to create a uniform probability distribution over a linear growing and moving set of diagonal nodes. A pure random walk would create a binomial distribution. So, the probability distribution "pushes" to the corners, see Fig. 4.

Likewise in the canonical algorithm we switch after some time to a deterministic cycling algorithm. The difference is, that this cycle is adapted to the first phase and is a perfect Hamiltonian cycle, see Fig. 5.

The proof relies on the bait-and-switch-strategy, where the bait is a diagonal field of length t and width $2t_f$. In the neighborhood of such a field at least $\Omega(t)$ robots must be placed at the beginning or the offline strategy does not succeed within first visit time t_f. The first phase of the cleaner strategy moves these robots to a given target node with probability $\mathcal{O}(1/t)$. So, a constant number of robots pass any target node within any time frame of length $\mathcal{O}(t_f)$. Since, the robots' random decisions are independent an increase of a factor of $\mathcal{O}(\log n)$ gives the time bound for the first phase.

For the second cycling phase, we have chosen the cycle with respect to the first phase, such that the same argument can be reused in order to estimate the maximum distance between two nodes on this cycle. The full proof can be found in the Appendix A.3.

This algorithm can be easily adapted for the grid graph, which consists of the same node set, but edges $\{(i,j), (i+1, j)\}$ for $i \neq m, (i,j) \in V$ and $\{(i,j), (i, j+1)\}$ for $j \neq m, (i,j) \in V$.

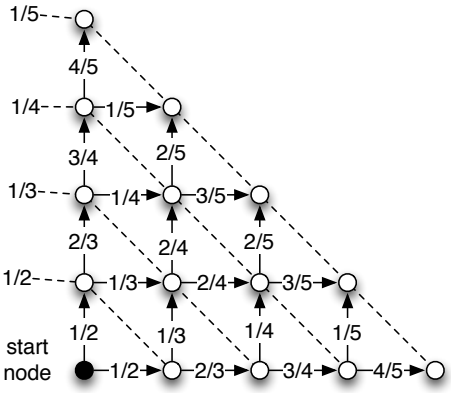

Fig. 4. Torus cleaner strategy

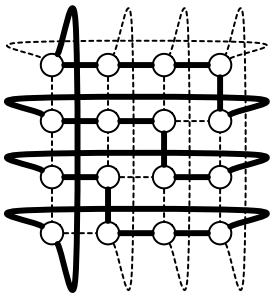

Fig. 5. Final cycle through the torus

Theorem 3. *There exists a high probability $\mathcal{O}(\log n)$-competitive visit time cleaning algorithm for the $m \times m$-grid graph with $n = m^2$ nodes.*

Proof. We embed a $2m \times 2m$-torus graph G_T on the $m\times$-grid graph G_G by mapping the four nodes $(x, y), (2m - x + 1, y), (x, 2m - y + 1), (2m - x + 1, 2m - y + 1)$ onto the node $(x, y) \in V(G_G)$. Note that the edges of the torus map to edges in the grid.

At the beginning we choose for a robot a random representative in the torus graph and then we follow the algorithm for the torus graph. The proof is analogous to the one of the torus graph presented in the appendix except to a constant factor increase of the competitive factor.

6 Unaware Parallel Traversal of General Graphs

For general graphs we use a partition of the graph, which balances the work load of the robots. For the randomized partition we are inspired by the techniques of embedding tree metrics for graphs [7].

We partition the graph into disjoint recruitment areas $R_1, \ldots, R_n \subseteq V$. All robots in a recruitment area R_i have to visit the nodes in a working area W_i which is a proper subset of R_i. These sets are defined by a random process such that each node has a constant probability to be contained in a working area and we show that the number of robots in the recruitment area is large enough to ensure that this node is visited with constant probability. This constant probability will be increased later on by repeating the partitioning several times.

We give a formal description of the sets used in Algorithm 3. The recruitment partition uses center nodes $c_1, \ldots, c_n$ which are given by a random permutation π of all nodes $V = \{v_1, \ldots, v_n\}$, i.e. $c_i = v_{\pi(i)}$. The partition is based on the neighborhood set $N_\ell(u)$, which is the set of nodes v for which the distance to u is at most ℓ. So, we define for a radius ℓ and for all $i \in \{1, \ldots, n\}$.

$$R_i := N_\ell(v_{\pi(i)}) \setminus \bigcup_{j=1}^{i-1} N_\ell(v_{\pi(j)}) \,. \tag{1}$$

The working areas are defined for radius ℓ and an estimation of the first visit time $t \in [t_f, 2t_f]$ as

$$U_i := N_{l-2t}\left(v_{\pi(i)}\right) \setminus \bigcup_{j=1}^{i-1} N_{\ell+2t}\left(v_{\pi(j)}\right) \tag{2}$$

$$W_i := N_t\left(U_i\right) \tag{3}$$

We denote by $W = \bigcup_{i=1}^{n} W_i$ the set of nodes that will be worked on and let $U := \bigcup_{i=1}^{n} U_i$.

These definitions are used for a probabilistic cleaning Algorithm 3, which covers a constant part of the graph. The ONE-SHOT-CLEANING algorithm makes use of an

Algorithm 3: ONE-SHOT-CLEANING $G = (V, E)$ using $V = R_1 \dot\cup \cdots \dot\cup R_n$ and $W_1, \ldots, W_n \subseteq V$

Choose i such that $s_r \in R_i$
$T_i \leftarrow$ STEINER-TREE-APPROXIMATION(W_i)
$C_i \leftarrow$ DFS-Cycle(T_i)
Move to a random node of C_i
Walk on C_i for $68t \log n$ rounds
Move back to s_r

straight-forward constant factor Steiner-tree approximation based on Prim's minimum spanning tree algorithm, presented as Algorithm 4.

Algorithm 4: STEINER-TREE-APPROXIMATION with input $G = (V, E)$, $W \subseteq V$

$(C_1, \ldots, C_p) \leftarrow$ connected components of W in G
while $p > 1$ **do**
  Choose the component C_j with the nearest node to C_1
  $W \leftarrow W \cup$ (node set of shortest path between C_1 and C_j to W)
  $(C_1, \ldots, C_p) \leftarrow$ connected components of W
end
return spanning tree of C_1

The following lemma shows that every node is chosen with probability of at least $\frac{1}{4}$ to be the target of a robot cleaning in some area W_i.

Lemma 3. *For a graph G, a node $v \in V$, β chosen randomly from $[1, 2]$, a random permutation π over $\{1, \ldots, n\}$, and for $l = 8\beta t \log n$ the probability that $v \in W$ is at least $\frac{1}{4}$.*

We refer to the appendix A.4 for the proof.

Now, we investigate whether there are enough robots in the recruitment area R_i in order to explore W_i. The number is large enough if a given node is explored with a constant probability. However, there is a major problem: U_i, W_i, or R_i might be disconnected. So robots might travel long routes between the nodes in W_i outside of W_i or even R_i.

Therefore, we need an upper bound on the size of these connecting routes. This has been the motivation to extend U with a surrounding of t neighborhood nodes. So, for $\beta \in [1, 2]$ we have the following lemma.

Lemma 4. *For $\ell = 8\beta t \log n$, let T_i be the tree connecting all nodes in W_i constructed in Algorithm 4. Then,*

$$|V(T_i)| \leq 17|W_i| \log n \, .$$

Proof. Each of the p connected components $C_1, \ldots, C_p$ of W_i has at least one node of U and its t-neighborhood. So, C_j has at least t nodes, implying $|W_i| \geq pt$. Every node of W_i has distance of at most $\ell = 8\beta t \log n$ to $v_{\pi(i)}$. The maximum distance between two components is thus at most $16\beta t \log n$ because of the triangle inequality. Which implies that at most $16(p-1)\beta t \log n$ nodes are added to connect the original p connected components. So,

$$\begin{aligned} |V(T_i)| &\leq 16(p-1)\beta t \log n + |W_i| \\ &\leq 16 \, \frac{p-1}{p} \, |W_i| \log n + |W_i| \\ &\leq 17|W_i| \log n \, . \end{aligned}$$

The following lemma shows that the ONE-SHOT-CLEANING algorithm needs only a logarithmic overhead.

Lemma 5. *The number of moves of a robot using* ONE-SHOT-CLEANING *for $\ell = 8\beta t \log n$ and $\beta \in [1, 2]$ is at most $100t \log n$.*

Proof. The maximum distance of any node from u to W_i is at most $\ell - t = 8\beta t \log n - t \leq 16t \log n$. So, moving to the start node and moving back to the start node needs at most $32t \log n$ rounds. Moving on C_i needs $68t \log n$ rounds resulting in $100t \log n$ rounds.

Now, we need to show that the number of robots in the recruitment area R_i is large enough. This follows by Lemma 1 substituting $A = W_i$.

Lemma 6. *If the robots are placed such that a first visit time of t_f is possible, and $t \in [t_f, 2t_f]$, then for the number k_i of robots originally placed in R_i we have*

$$k_i \; \geq \; \frac{|W_i|}{t_f + 1} \; \geq \; \frac{|W_i|}{2t} \, .$$

Proof. A single robot can explore at most $t_f + 1$ nodes in the first t_f rounds. Therefore the minimum amount of nodes to be explored by all robots in R_i is $k_i(t_f + 1) \leq 2k_i t_f$.

These observations allows us to find a general strategy for the first visit problem for unaware parallel cleaners.

Theorem 4. *Algorithm 5 is a high probability $\mathcal{O}(\log^2 n)$-competitive first visit algorithm for every undirected graph.*

Repeating the ONE-SHOT-CLEANING $\mathcal{O}(\log n)$ times gives us high probability. The full proof can be found in Appendix A.5.

The visit time problem needs more moves, since a robot may make a fast first visit, but does not know when to end. Our solution is to guess the first visit time.

Algorithm 5: High probability first visit cleaner of $G = (V, E)$

for $i \in \{1, 2, \ldots, \log n\}$ **do**
  $t \leftarrow 2^i$
  for $j \in \{1, \ldots, 4(c+1) \ln n\}$ **do**
    Choose randomly $\beta \in [1, 2]$
    Choose random permutation π over V
    ONE-SHOT-CLEANING$(G, \ell = 8\beta t \log n, t, \pi)$
  end
end

Algorithm 6: High probability visit of $G = (V, E)$

Choose uniform at random $i \in \{1, 2, \ldots, \log n\}$
$t \leftarrow 2^i$
Choose randomly $\beta \in [1, 2]$
Choose random permutation π over V
ONE-SHOT-CLEANING(G, t, β, π)
Traverse the graph by DFS yielding a cycle C with $V(C) = V$ of length $2n$
Go to a random node visited during the one shot cleaning
while `true` **do**
  Walk to the next node of C
end

Theorem 5. *Algorithm 6 is an high probability $\mathcal{O}(\log^3 n)$-competitive visit algorithm for every undirected graph.*

Proof. Lemma 3 implies that $P(w \in W_i) \geq \frac{1}{4}$ if $\ell = 8\beta t \log n$. The probability that a robot chooses the correct value $t = 2^i \in [t_f, 2t_f]$ is $1/\log n$. So, the probability that a node is visited within first visit time $800ct_f \log n$ is at least $p = \frac{1}{4 \log n}$. By Lemma 2 this implies a visit time algorithm with high probability with time $\mathcal{O}(t_f \log^3 n)$.

7 Conclusion

We discuss a central question of distributed algorithms: How much do we benefit from communication? Or to put it otherwise: Can we cope with a parallel problem if communication is not available? We have shown that first visit can be achieved with an overhead of $\mathcal{O}(\log^2 n)$ and visit with $\mathcal{O}(\log^3 n)$ in general graphs. This means that we can cope quite well without any communication.

For the grid and torus we show an even stronger bound of $\mathcal{O}(\log n)$. This matches the lower bound of $\Omega(\log n)$ given by the coupon collector's problem. Unlike the algorithm presented for general graphs the parallel unaware cleaner strategy for torus and grids have only small constant factors involved. Furthermore, the grid represents a typical application areas for such robots. So, we can very well envisage our cleaning strategies to be implemented onto current room cleaning and lawn mowing robots.

References

1. R. Baezayates, J. Culberson, and G. Rawlins. Searching in the Plane. *Information and Computation*, 106(2):234 – 252, 1993.
2. T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.
3. N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.
4. D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznanski. Fast Collaborative Graph Exploration. In *Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 520–532. Springer Berlin Heidelberg, 2013.
5. M. Dynia, J. Kutylowski, F. Heide, and C. Schindelhauer. Smart Robot Teams Exploring Sparse Trees. In *Mathematical Foundations of Computer Science 2006*, volume 4162 of *Lecture Notes in Computer Science*, pages 327–338. Springer Berlin Heidelberg, 2006.
6. M. Dynia, J. Lopuszanski, and C. Schindelhauer. Why robots need maps. In *Proceedings of the 14th international conference on Structural information and communication complexity*, SIROCCO'07, pages 41–50, Berlin, Heidelberg, 2007. Springer-Verlag.
7. J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.
8. P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Netw.*, 48:166–177, October 2006.
9. G. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 216–227, Oct 1976.
10. Y. GuoXing. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. *European Journal of Operational Research*, 81(3):557 – 560, 1995.
11. Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization*, pages 1–16, 2012.
12. R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.
13. L. Lovász. Random Walks on Graphs: A Survey. In D. Miklós, V. T. Sós, and T. Szőnyi, editors, *Combinatorics, Paul Erdős is Eighty*, volume 2, pages 353–398. János Bolyai Mathematical Society, Budapest, 1996.
14. D. J. Newman. The double dixie cup problem. *American Mathematical Monthly*, pages 58–61, 1960.
15. C. Ortolf and C. Schindelhauer. Online Multi-robot Exploration of Grid Graphs with Rectangular Obstacles. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 27–36, New York, NY, USA, 2012. ACM.
16. C. H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.
17. D. Portugal and R. Rocha. A survey on multi-robot patrolling algorithms. In L. Camarinha-Matos, editor, *Technological Innovation for Sustainability*, volume 349 of *IFIP Advances in Information and Communication Technology*, pages 139–146. Springer Berlin Heidelberg, 2011.
18. D. Rosenkrantz, R. Stearns, and P. Lewis. Approximate algorithms for the traveling salesperson problem. In *Switching and Automata Theory, 1974., IEEE Conference Record of 15th Annual Symposium on*, pages 33–42, Oct 1974.

A Appendix

A.1 Canonical Cleaning

Theorem 1. *Using the* CANONICAL CLEANING *it is possible to achieve a long-term visit time of* $\mathcal{O}((n/k)\log n)$ *and a visit time of diameter*$(G) + \mathcal{O}((n/k)\log n)$ *with high probability.*

Proof. We choose for each robot an independent uniform random choice of the nodes of the cycle P as the *cycle-start-node*. The *waiting-time* is defined as *diameter*$(G) - |s_r, v_s|$. So, all nodes start the traversal at the same time.

Let g be a subpath on the cycle P of length at most $2n$. The probability that no robots are in this subpath is $(1 - \frac{g}{|P|})^k$. For k robots a subpath $g \geq \frac{2cn\ln n}{k}$ is empty with probability

$$\left(1 - \frac{g}{|P|}\right)^k \leq \exp\left(-\frac{gk}{|P|}\right) \leq \exp\left(-\frac{gk}{2n}\right) \leq \exp\left(-c\ln n\right) \leq n^{-c}\,.$$

Hence, the maximum gap between two nodes on the cycle P is at most $\mathcal{O}((n/k)\log n)$ with high probability.

So, the long term visit time is bounded by this gap. From the waiting time, the first visit time follows. Note that after the first visit, the revisit time matches the long term visit time.

A.2 Canonical Algorithm First Visit

Lemma 2. *Assume there exists a parallel unaware cleaner algorithm* $\mathcal{A}$ *for* k *robots on a graph with* n *nodes, where for all nodes* u *the probability that the first visit time is less or equal than* t_f *is at least* $p > 0$*. Furthermore,* t_f *and* p *are* **known***. Then, this cleaning algorithm can be transformed into a canonical algorithm having visit time* $\mathcal{O}(\frac{1}{p}t_f \log n)$ *with high probability.*

Proof. Let $P(r)$ with $|P(r)| \leq t_f$ be the resulting path of robot r performing algorithm $\mathcal{A}$. Then, the *cycle-start-node* of the canonical algorithm is defined by choosing a random uniform node v_s from $P(r)$. We set *waiting-time*(r)=0.

We now show that this algorithm fulfills the time behavior.

1. The first visit time can be proved as follow.
   Each node is visited with probability of at least $\frac{p}{t_f}$. However, there are dependencies between these events, since nodes might be visited by the same robot. So, we consider the subpath before a node v of length $\frac{2ct_f\ln n}{p}$ on a cycle C of length $2n$ with $V(C) = V$. Then, at least $c\ln n$ different robots have positive probabilities to visit this interval. Let $1, \ldots, k$ be these robots and let p_i be the probability that one of these robots visits this interval. For these probabilities we have $\sum_{i=1}^{k} p_i \geq \frac{p}{t_f}\frac{ct_f\ln n}{p} = c\ln n$, since otherwise a node exists which is visited with smaller probability than $\frac{p}{t_f}$.
   The probability for not visiting this interval is therefore
   $$\prod_{i=1}^{k}(1-p_i) \leq \prod_{i=1}^{k}\exp\left(-p_i\right) \leq \exp\left(-\sum_{i=1}^{k}p_i\right) \leq \exp\left(-c\ln n\right) \leq n^{-c}\,.$$
   Since with high probability a cycle-start-node is chosen on the cycle P at most $(2ct_f\ln n)/p$ nodes before v, v will be visited after $t_f + 2\frac{c}{p}t_f\ln n$ steps for the first time w.h.p. From the union bound the claim follows.
2. The visit time follows by the following observation: From the observations above we know that the subpath of length $2ct_f\ln n$ on P before and after any node is visited within time t_f. Therefore the visit time of a node is at most $4ct_f\ln n + 2t_f$.

A.3 Analysis of Torus Algorithm

Theorem 2. *Algorithm 2 is a high probability $\mathcal{O}(\log n)$-competitive visit cleaning algorithm for the $m \times m$-torus graph.*

Proof. The following Lemma shows that the torus algorithm distributes the robots with equal probabilities.

Lemma 3. *For all $t \in \{1, \ldots, \sqrt{n}\}$, $i \in \{0, \ldots, t\}$ the probability that a robot starting at node $(s_{r.x}, s_{r.y})$ is at node $(s_{r.x} + i, s_{r.y} + (t - i))$ after t rounds is $1/(t + 1)$.*

Proof. This follows by induction. For $t = 0$ the probability is 1 that the robot is at the start node $(s_{r.x}, s_{r.y})$. Assume that at round $t - 1$ the claim is true.

For the induction we have to consider three cases:

- If $x = s_{r.x}$ and $y = s_{r.y} + t$ then the probability to move to this point is the product of the stay probability at $(x, y - 1)$ and the probability to increment y. By induction this is $\frac{1}{t}\left(1 - \frac{1}{t+1}\right) = \frac{1}{t+1}$.
- If $y = s_{r.y}$ and $x = s_{r.x} + t$ then the probability to move to this point is the product of the stay probability at $(x, y - 1)$ and the probability to increment x. By induction this is again $\frac{1}{t}\left(1 - \frac{1}{t+1}\right) = \frac{1}{t+1}$.
- For all other cases we have to combine the probability to increment x and y, the sum of which is $\frac{t}{t+1}$. By induction we get as probability $\frac{1}{t}\frac{t}{t+1} = \frac{1}{t+1}$ claim follows.

Assume that t_f is the first visit time time for a robot placement in the torus. For the cleaning of a target node (x, y) we choose a set of nodes S with $t - 4t_f$ nodes at a diagonal in distance t, see Fig. 6. $A = N_{t_f}(S)$ is now the bait, i.e. the area, which guarantees the minimum number of robots the recruitment area $N_{t_f}(A)$. Lemma 1 states that at least $|A|/(t_f + 1)$ robots must be in this recruitment area $N_{t_f}(A)$. Now, the cleaning algorithm makes sure that all these robots pass through the target node during the time interval $[t - 2t_f, t + 2t_f]$ with a probability of at least $1/(t + 2t_f + 1)$. Now, the size of $|A|$ is at least $2t_f(t - 4t_f)$. So, the expected number of robots passing through the target node is at least

$$\frac{|A|}{(t_f + 1)(t + 2t_f + 1)} \geq \frac{2t_f(t - 4t_f)(t + 2t_f + 1)}{t_f + 1} \geq \frac{t - 4t_f}{t + 2t_f + 1} .$$

So for $t \geq 10t_f$ we expect at least a constant number of $\frac{1}{2}$ robots passing through any node in a time interval of length $3t_f$. If we increase the time interval to the size of some $ct_f \log n$ for some appropriately chosen constant c, applying a Chernoff bound ensures us to visit this node with at least one robot with high probability.

This proves that in the first phase of the algorithm we visit (and revisit) each node in every time intervals of length $\mathcal{O}(t_f \log n)$.

It remains to show that in the second phase, where the algorithm enters the cycle the distance on the cycle is bounded by $\mathcal{O}(t_f \log n)$. For this, we consider $4t_f < \sqrt{n}$ consecutive nodes on the cycle, which lie on $4t_f$ consecutive diagonals, see Fig. 7. So, all of the $|A|/(t_f + 1)$ robots in the recruitment area have a target node, which can be reached after $\sqrt{n}$ steps. For each of these target nodes, the probability to be reached by a robot on the corresponding diagonal is at least $\frac{1}{\sqrt{n}}$. The minimum size of $|A|$ is at least $\sqrt{n} - 2t_v$, which results in an expected number of at least

$$\frac{2t_f(\sqrt{n} - 2t_f)}{(2t_f + 1)\sqrt{n}} \geq 1 - \frac{t_f}{\sqrt{n}}$$

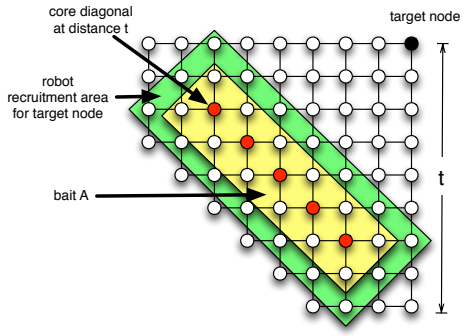

Fig. 6. The robot recruitment area for robots exploring the target node.

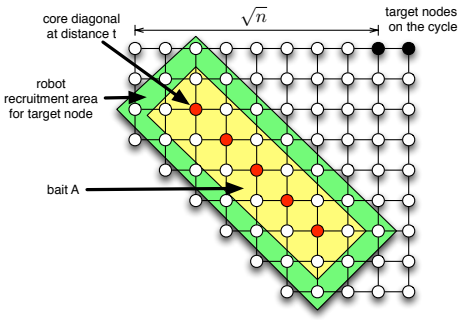

Fig. 7. The robot recruitment area for robots on the cycle.

robots on the target nodes of the cycle. For $t_f \leq \frac{1}{2}\sqrt{n}$ this means that the expected number of robots in an interval of length $4t_f$ is at least $\frac{1}{2}$. So, the longest empty interval has length of at most $\mathcal{O}(t_f \log n)$ by applying Chernoff bounds on $\mathcal{O}(\log n)$ neighbored intervals.

For $t_f \geq \frac{1}{2}\sqrt{n}$ we consider $\sqrt{n}$ consecutive nodes on consecutive diagonals. Every robot ends the first phase and starts the cycle within this interval with probability $\frac{1}{\sqrt{n}}$. The minimum number of robots to explore all n nodes is at least $\frac{n}{t_f+1}$, which follows by Lemma 1 for $A = V$. Now, for $c\frac{t_f}{\sqrt{n}} \log n$ neighbored intervals on the cycle each of length $\sqrt{n}$ the probability that a single robot chooses a node in this interval is at least

$$\frac{t_f}{\sqrt{n}} \frac{c \log n}{\sqrt{n}} = c\frac{t_f}{n} \log n \, .$$

So, the expected number of robots is $c\frac{n}{t_f}\frac{t_f}{n} \log n = c \log n$ for an time interval of length $c\frac{t_f}{\sqrt{n}}\sqrt{n} \log n = ct_f \log n$. Now, by Chernoff bounds the probability that we find this interval to be empty has a probability of at most $n^{-c'}$ for some constants c, c'.

So, the maximum distance of two robots on a cycle in the first and second phase is at most $\mathcal{O}(t_f \log n)$ with high probability. Since the visit time is at least the first visit time the competitive ratio of $\mathcal{O}(\log n)$ follows.

A.4 Proof of Lemma 3

Lemma 3. *For a graph G, a node $v \in V$, β chosen randomly from $[1, 2]$, a random permutation π over $\{1, \ldots, n\}$, and for $\ell = 8\beta t \log n$ the probability that $v \in W$ is at least $\frac{1}{4}$.*

Proof. We will prove that $P(v \in U) \geq \frac{1}{4}$, which implies the claim because $U \subset W$.

Consider the first node w in the $\ell + 2t$-neighborhood of v according to the random permutation π, i.e. $w = u_{\pi(i*)}$ where $i^* = \min\{i \mid |v, u_{\pi(i)}| \leq \ell + 2t\}$. If w is closer than $\ell - 2t$ to v, i.e. $|v, w| \leq \ell - 2t$, then v is in the working area of w (and U), since no node with smaller index can be closer than w, i.e. $w \in U_{i*} \subseteq U$. On the other hand if this node is in the critical distance $|v, w| \in (\ell - 2t, \ell + 2t]$, then it is excluded from U_{i*} and since i^* has the smallest index in the vicinity it is also not in any other working area, i.e. $v \notin U$. Since π is a random permutation the probability of $v \in W$ is given by the number of elements in the closer vicinity:

$$P_\ell(v \in U) = \frac{|N_{\ell-2t}(v)|}{|N_{\ell+2t}(v)|}$$

This implies

$$\prod_{i=0}^{2\log n} P_{\ell+4it}(v \in U) = \frac{|N_{\ell-t}(v)|}{|N_{\ell+8t\log n+2t}(v)|} \geq \frac{1}{n} \tag{4}$$

Now, we choose β randomly from $\{1, 1+\frac{1}{2\log n}, 1+\frac{2}{2\log n}, \ldots, 1+\frac{2\log n-1}{2\log n}\}$ and compute $\ell = 8\beta t \log n$. Hence,

$$P(v \in U) = \frac{1}{2\log n} \sum_{i=0}^{2\log n-1} P_{8t\log n+4it}(v \in W)$$

Assume that $P(v \in U) < \frac{1}{4}$, then at least half of all values of $(P_{8t\log n+4it}(v \in W))_{i\in\{0,\ldots,2\log n-1\}}$ are smaller than $\frac{1}{2}$. Then, we observe the following.

$$\prod_{i=0}^{2\log n} P_{8t\log n+4it}(v \in U) < \left(\frac{1}{2}\right)^{\log n} = \frac{1}{n}\,,$$

which contradicts (4). Therefore $P(v \in W) \geq P(v \in U) \geq \frac{1}{4}$.

The same argument holds, if we choose β randomly from the real interval $[1, 2]$.

A.5 Analysis of Algorithm 5

Theorem 3. *Algorithm 5 is a high probability $\mathcal{O}(\log^2 n)$-competitive first visit algorithm for every undirected graph.*

Proof. Consider the round of the outer loop, where $t = 2^i \in [t_f, 2t_f]$, where t_f is the first visit time of the optimal algorithm. We show that in this round all nodes will be explored with high probability. Lemma 5 states that the number of robot moves of ONE-SHOT-CLEANING is bounded by $100 \cdot 2^i \log n$. So, the overall number of each robot moves is bounded by $800(c+1)\log^2 n$.

For any node u the probability, that the ONE-SHOT-CLEANING algorithm for $\ell = 8\beta t \log n$ chooses $u \in W$ is at least $\frac{1}{4}$ following Lemma 3. If u resides in W_i, the number of robots performing the cleaning is at least $|W_i|/(2t)$ implied by Lemma 6. These k_i robots have to explore a cycle of length at most twice the size of the connected Steiner-tree computed in Algorithm 4. These are at most $34|W_i|\log n$ nodes. Now, Algorithm 3 starts with a random node node and explores $68t\log n$ nodes. So, after one execution of the ONE-SHOT-CLEANING algorithm the probability of a node not to be explored is at most

$$1 - \frac{1}{4}\frac{68t\log n}{34|W_i|\log n} = 1 - \frac{t}{2|W_i|}$$

The cleaning is be independently repeated for $k_i \geq \frac{|W_i|}{2t}$ times.

$$\left(1 - \frac{t}{2|W_i|}\right)^{\frac{|W_i|}{2t}} \leq e^{-\frac{1}{4}}$$

Hence, the maximum probability of a node not to be explored after $4(c+1)\ln n$ repetitions is at most $\frac{1}{n^c}$.