Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 97–106

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

DynFloR: A Flow Approach for Data Delivery Optimization in Multi-Robot Network Patrolling

Vlad-Sebastian Ionescu[a], Zsuzsanna Oneţ-Marian[a], Marin-Georgian Bădiţă[a,*], Gabriela Czibula[a], Mihai-Ioan Popescu[b], Jilles S. Dibangoye[b], Olivier Simonin[b]

[a] *Department of Computer Science, Babeş-Bolyai University*
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania
[b] *INSA Lyon, University of Lyon, CITI Lab, Inria Chroma, France*

Abstract

Deploying fleets of mobile robots in real scenarios and environments raises several scientific challenges. One of them concerns the ability of the robots to adapt to the dynamics of their environment. We introduce *DynFloR*, a dynamic network flow based approach for finding optimal policies for *data delivery* in *multi-robot network patrolling* where the robots can communicate instantly and free of charge one to another when they meet, there is a periodicity of the robot meetings and the distribution of the data collected during the patrol is regular. Experiments on randomly generated synthetic examples are performed for evaluating the performance of the *DynFloR* method. The performed experiments empirically show that independent of the problem setting (such as number of robots, memory of the robots) the amount of data transferred to a base station per unit of time converges to an equilibrium state. The case of lost data has been also examined through various experiments, but it requires further experimentation as well as in-depth analysis.

Keywords: Optimization; Decision making; Multi-robot patrolling; Network flow
2000 MSC: 68T40; 93C85; 05C21

1. Introduction

Patrolling represents the repeated visit of certain places, over an area [4]. The *multi-robot patrolling* (also called *multi-agent patrolling*) problem (MRP) consists in minimizing the time between two consecutive visits of the same place (called *idleness*), using two or more robots. In the literature, this problem is generally considered to be NP-hard, while solutions often involve cyclic paths [10]. Applications of the multi-agent patrolling problem include:

* Corresponding author. Tel.: +4-264-405-327; fax: +4-264-591-906.
* Corresponding author. Tel.: +4-264-405-327; fax: +4-264-591-906.
E-mail address: bmir2236@scs.ubbcluj.ro

1877-0509
10.1016/j.procs.2019.09.164

surveillance tasks (e.g. anomaly detection, intruder detection), area coverage, data collection tasks (e.g. for the case of wireless sensor networks - WSNs), rescue operations (e.g. people or objects in dangerous situations) [20, 8].

A major challenge when deploying fleets of mobile robots in real scenarios is the ability of the robots to adapt to the complexity of their environment. In dynamic environments the robots need to be able to provide robust solutions to complex tasks and to adapt to changes in their environment. Multi-agent patrolling tasks have been investigated within the multi-agent research community and various algorithms based on reactive and cognitive architectures have been introduced [17]. Still, the existing patrolling approaches are dedicated to static environments and those regarding dynamic environments (*dynamic multi-robot patrolling* - DMRP) are in early phases. The present work considers a simplified, deterministic setting for the MRP problem, in which the robots can communicate instantly and free of charge one to another when they meet, there is a periodicity of the robot meetings and the distribution of the data collected during the patrol is regular. We assume that the robots meet periodically during the patrol at rendez-vous places, and they communicate in order to cooperate and pass the collected data between them up to a *base station* (*sink*).

The study performed in this paper represents the starting point of a research that is being conducted for optimizing data delivery in DMRP using *reinforcement learning* (RL) [2], in the general case where the environment is uncertain (i.e. there is no periodicity for the robot meetings). In this case, the robots will have to learn through *reinforcement* [13] the data-transfer policy, in order to maximize the quantity of transmitted data.

The contribution of this paper consists in a new approach, *DynFloR*, based on dynamic network flows for determining the optimal policy for delivering data in multi-robot network patrolling under the following assumptions: the environment is deterministic and the robots have periodic meetings. The proposed *DynFloR* method adapts the approach proposed by Kotnyek [14] for the problem of MRP. It can be also extended for the DMRP under the previously mentioned assumptions. We assume the data transfer to be dynamic and we address the following research question: *How to optimize data-passing between each robot patrol cycle in order to maximize the quantity of data transmitted to the sink?* Experiments and simulations performed on several synthetic and randomly generated examples emphasize that our proposal is able to determine a policy that maximizes the data transferred to the base station up to a given moment in time. The answer to the previously stated RQ through the *DynFloR* approach will provide us additional insights on the non-deterministic MRP/DMRP problem and its modelling as a reinforcement learning task. To the best of our knowledge, *DynFloR* approach is new in the MRP literature.

The rest of the paper is structured as follows. Section 2 presents the fundamental concepts regarding the multi-robot patrolling and presents the problem statement. Our *DynFloR* approach for data delivery optimization in a deterministic setting of the MRP is introduced in Section 3. Section 4 presents the experimental results obtained by evaluating *DynFloR* and discussion about the current solution and its advantages and limitations. Existing approaches to multi-robot patrolling are reviewed in Section 5 and the differences between *DynFloR* and similar existing work are also highlighted. Section 6 contains the conclusions of the paper and directions to continue our research.

2. Problem Setting

For a better understanding of the problem, a background on multi-robot patrolling is presented hereafter. Then, the problem statement is described, together with the theoretical model of the problem.

2.1. Multi-robot patrolling (MRP)

We consider the general case of patrolling with a fleet of robots. *Multi-robot patrolling* consists in repeatedly visiting a set of targets in the area, with several agents (e.g. drones, autonomous vehicles etc.), while trying to optimize the visit frequency of the targets. A formalization of the problem of multi-robot patrolling with data collection constraints has been formalized in [18] and addressed through clustering-based partitioning. Thereby, considering that $T=\{t_1, t_2, \cdots, t_n\}$ is a set of targets that have to be patrolled, a patrolling solution consists of a partition $\mathcal{K}=\{K_1, K_2, \cdots, K_p\}$ of T ($K_i \subseteq T, K_i \neq \emptyset, \forall 1 \leq i \leq p$ and $T = \bigcup_{i=1}^{p} K_i$ and $K_i \cap K_j = \emptyset,\ \forall 1 \leq i, j \leq p, i \neq j$) and cycles C_i's built on every cluster K_i, respectively.

Assuming such a solution, while the robots patrol a cluster of targets by following the cyclic path and collect data from the targets, they have to pass the collected data to their neighbor robots (cycles) in order to deliver it to the base station. Each robot can have the choice of transmitting the collected data to different neighbors: data exchange with

close cycles. By doing so, the robots form a network which allows the flow of information reach the base station (sink). A challenge in such a setting is to ensure that the robots communicate efficiently in order to optimize the delivery of the collected data to the base station.

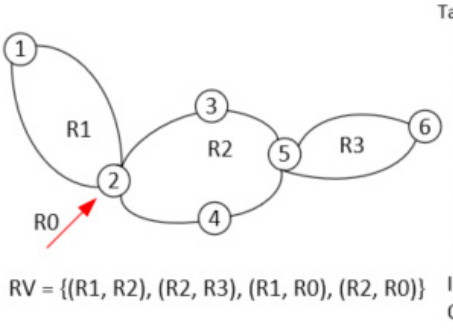
2.2. Problem definition. Theoretical model

The problem we are approaching in this paper is defined in the following. We assume the particular case of a deterministic environment, in which the network of robots is centralized and offline: *There are n robots, which can meet and exchange data. The robots are continuously collecting data on the cycles they patrol. There are initial meeting times for the robots and meeting periods. The goal is to maximize the quantity of data arriving to a sink (a base station) in a given time T.*

Let us consider the following theoretical model. We denote by $\mathcal{R} = \{R_1, R_2, \cdots, R_n\}$ the set of patrolling robots. We assume the sink is denoted by R_0. Each robot $R_i \in \mathcal{R}$ patrols a certain cycle C_i consisting of k_i targets, i.e. $C_i = \{t_i^1, t_i^2, \ldots t_i^{k_i}\}$. At each visit of the target t_i^j, an amount d_i^j of data will be collected by robot R_i. Hence, after patrolling a complete cycle, robot R_i collects a quantity of data $D_i = \sum_{j=1}^{k_i} d_i^j$. This data is stored in robot R_i's memory, of maximum capacity $MaxCapacity_i$. We consider that the sink has no memory limit, i.e. $MaxCapacity_0 = \infty$. For data exchange, two robots R_i and R_j have periodical meetings (*rendez-vous* points) starting at time T_{ij} when they first meet, with a period of π_{ij} time steps.

In our approach, we assume that the incoming data model is *regular* and *known* and that there is a single base station (called *sink*). The data is continuously collected by the robots after patrolling their cycle and has to be transmitted to the sink. We also assume that it takes one unit of time for a robot to go from one target to the next one and that transferring data does not take time.

Figure 1 depicts a synthetically generated example, a much simplified version of the MRP problem with 4 robots and 6 targets, in which data is static (i.e. collected only once by a robot, at the beginning of the simulation), the robots' meetings are periodical and the solution is the minimum time required for transferring all collected data to the *sink* (denoted by R0). In Figure 1 *RV* represents the meetings between the robots, T_{ij} is the time stamp when robots i and j first meet and π_{ij} is the meeting period for robots i and j. Assuming that the initial data for every robot is 10 and the maximum memory for all robots is 15, the minimum time in which all data can be transferred to the sink is 8. The right side table from Figure 1 depicts the decision making strategy for the agents, at each time moment, obtained using a brute force approach. Each line from the table represent a time moment and the columns from the second to the fifth indicate the target visited by the robots (including the sink) at a given time. If two robots visit the same target at a given time, there is a meeting between them and data can be transferred. The last three columns on a row (time) present the data transfer strategy between the robots at that moment, i.e. robot **From** transfers to robot **To** the quantity **Quant**.



Targets cycled

R1: 1-2-1
R2: 3-5-4-2-3
R3: 6-5-6

Meeting times

$T_{12} = 4$	$\pi_{12} = 4$	4, 8, 12, 16, ...
$T_{23} = 2$	$\pi_{23} = 4$	2, 6, 10, 14, ...
$T_{10} = 2$	$\pi_{10} = 2$	2, 4, 6, 8, 10, ...
$T_{20} = 4$	$\pi_{20} = 4$	4, 8, 12, 16, ...

Initial data in memory: 10
Capacity: 15

	Target				Transfer		
Time	R1	R2	R3	R0	From	To	Quant.
1	1	3	6	2	No meeting		
2	2	5	5	2	R1	R0	10
3	1	4	6	2	No meeting		
4	2	2	5	2	R2	R0	10
5	1	3	6	2	No meeting		
6	2	5	5	2	R3	R2	10
7	1	4	6	2	No meeting		
8	2	2	5	2	R2	R0	10

Fig. 1: Synthetic example.

3. Methodology

This section introduces our centralized and offline *DynFloR* approach for data delivery optimization in a deterministic setting of the MRP. We mention that *DynFloR* is applicable in a deterministic setting of DMRP, as well.

3.1. The proposed DynFloR approach

For modeling the dynamic data collection process from the multi-robot network previously defined in Section 2.2, we introduce a dynamic flow network based model which captures both the temporality of the patrolling process, as well as the dynamicity of the data collection and transfer.

3.1.1. The proposed dynamic network model

Our proposed model adapts the dynamic network flow approach introduced in [14] by taking into account the specific requirements of multi-robot network patrolling. The flow network graph G is constructed from moment 0 up to a moment of time $Time$. For each time step, the graph captures the state of each robot and the sink. Each of the states is represented by a node in the graph, which includes information such as robot decisions (data currently collected, data currently transferred, robot met for data exchange) or the amount of data stored. There are $(n+1)\cdot(Time+1)+1$ **nodes** in the dynamic flow graph. For each robot R_i including the sink ($\forall i,\ 0 \le i \le n$) there are $Time$+1 nodes, denoted by i_t ($\forall t,\ 0 \le t \le Time$). The node i_t corresponds to robot R_i at time t. The *sink* is viewed, in the proposed model, as robot R_0. Besides the nodes corresponding to the robots, there is an additional *source* node s connected to all robots (except the sink) at time step 0.

The directed **edges** (connections) between the flow network's nodes and their capacities are set as follows. There is a connection between the source node s and node i_0 ($\forall i,\ 1 \le i \le n$) and its capacity is set to the the initial data Q_i held by the robot R_i at the beginning of the simulation. For each time moment t when two robots R_i and R_j meet, two edges are added: one from i_t to j_t with capacity $MaxCapacity_j$ (i.e., the maximum memory of R_j) and one from j_t to i_t with capacity $MaxCapacity_i$ (i.e., the maximum memory of R_i). For each robot R_i (including the sink) an edge is created between i_t and i_{t+1}, $\forall t,\ 0 \le t \le Time-1$ (the edge has the capacity $MaxCapacity_i$). For each time moment t when new data is collected by a robot R_i, an edge connecting the source node s and the node i_t is added and its capacity is set to D_i.

We mention that the currently proposed model contains nodes corresponding to an entire cycle patrolled by a robot (without detailing the targets patrolled during each cycle). We also assume that a robot collects the data after it patrolled an entire cycle, this is why we add one edge with capacity D_i after k_i time units, instead of adding one edge with capacity d_i to every node. The previous assumptions do not reduce the generality of *DynFloR*, as we can easily extend the model by adding new data at every time step to every robot and the *DynFloR* algorithm remains the same. In this case, new data will be collected by a robot in a node corresponding to a target and not to an entire cycle (as in the current model).

3.1.2. DynFloR algorithm

After the flow network graph G is built as described in Section 3.1.1, a maximum flow [14] from the source s to the network sink (a node $0_t, t \le Time$) will give the total amount of data that can be transferred to the base station in t time steps. Our proposal *DynFloR* outputs the decision making strategy for each robot R_i, at each rendez-vous time, in order to increase the quantity of data received by the sink while respecting the memory constraint for each robot.

The pseudocode for constructing the flow graph G is presented in Algorithm 2 and the *DynFloR* algorithm is given in Algorithm 1. For determining the maximum flow in a graph G, the Preflow-Push algorithm [16] denoted by $getFlow(G)$ is used. We also denoted by *addNode* and *addEdge* the operations for creating a node and an edge with a certain capacity, respectively.

We note that the *objective function* which has to be maximized in our approach is $f(T) = \frac{data\ \ to\ \ sink\ \ until\ \ time\ \ T}{T}$. Alternatively, we envisage the minimization of $g(T) = \frac{input\ \ data\ -\ data\ \ to\ \ sink\ \ until\ \ time\ \ T}{T}$.

A brief analysis of the time complexity of *DynFloR* is given below. The time complexity of the *CreateGraph* sub-algorithm for constructing the dynamic network G is $O(|RV|\cdot Time)$. Therefore, the overall complexity of *DynFloR*, including computing the maximum flow, is $O(\mathcal{F}(n\cdot Time, |RV|\cdot Time))$, where $\mathcal{F}(V,E)$ is the time complexity of a flow algorithm applied on a graph with V nodes and E edges (in our graph we have $n\cdot Time$ nodes and $|RV|\cdot Time$ directed edges). For example, for the highest label preflow-push algorithm [7], we have $\mathcal{F} = O\left((n\cdot Time)^2\cdot\sqrt{|RV|\cdot Time}\right)$.

Algorithm 1 Algorithm *DynFloR*

Algorithm *DynFloR* is:
// determines the optimal policy for the robots' data delivery
Require: n, k - the number of robots and a list containing the robots' cycle lengths
$D, Q, MaxCapacity$ - lists containing the amount of data collected by each robot, its initial data and its maximum memory
$RV, Time$ - the list containing the pairs of robots that meet and the final moment of time considered
T, π - the initial meeting time for each pair of robots from RV and their meeting period
Ensure: returns the data delivery strategy for the robots from RV (the path that gives the maximum flow in G)
// read the input data
// construct the flow network graph G
$G \leftarrow CreateGraph(n,k,D,Q, MaxCapacity,RV,Time,T,\pi)$
// the data delivery policy *Policy* is computed from the path that corresponds to the maximum flow $MaxFlow$ in the graph G
$(MaxFlow, Policy) \leftarrow getFlow(G)$
EndAlgorithm

3.2. Example

For exemplifying how *DynFloR* works, let us consider a very simple example of MRP illustrated in Figure 2. It consists of only two robots, denoted by R1 and R2. The sink is marked as R0. Figure 2 also illustrates the first meeting time T_{ij} between robots i and j and the meeting period π_{ij} for robots i and j. We are also assuming that the data collected by a robot is 15, and the maximum memory $MaxCapacity_i$ for all robots is 55.

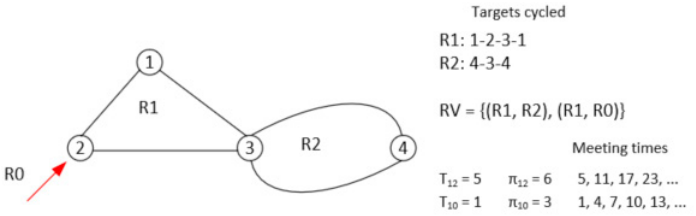


Fig. 2: Simple example.

For the MRP example from Figure 2, the associated flow network is presented in Figure 3. In our example, the moment of time T until the network is presented is 20. The amount of data which is collected by the robots is marked with green. For a better readability of Figure 3, we did not connect the green arrows to the source (as described in Section 3.1.1), but in the implementation they are connected. The decision making strategy for the robots, which is the output of the *DynFloR* algorithm proposed in Section 3.1.2, is marked with red. If an arc between two nodes i_t and j_t is labelled with a value v, it means that at time t robot R_i transfers to robot R_j the amount v of data. If the arc between i_t and i_{t+1} is labeled with v, it suggests that robot R_i stores in its memory the amount of data v for one unit of time (from t to $t + 1$).

From Figure 3 we can observe that 225 units of data were transferred to the sink (robot R0) by $t = 20$. We can also observe that there are 3 green arrows (robot R2 at times 14, 18 and 20) that do not have incoming data. This data is not lost, it is stored in the memory of the robots, but since until the time limit we have set for the simulation ($t = 20$) it cannot arrive to the sink, the flow algorithm does not consider it. But if we extended the time limit to include another meeting between robots R1 and R2, most of these data would arrive to the sink.

There is also the possibility to have lost data, a situation in which no matter how long we would extend the time limit, the data would never arrive to the sink. This is mainly due to the maximum memory capacity of the robots and the frequency of the collected data. For instance, if we considered the value 40 as the maximum capacity for robot R2 (and left the other settings used for Figure 3 the same), by time 5, when R2 first meets R1 and has the chance to

Algorithm 2 Function *CreateGraph*

function $CreateGraph(n, k, D, Q, MaxCapacity, RV, Time, T, \pi)$
// Create the flow network graph corresponding to the MRP problem
Require: $n, k, D, Q, MaxCapacity, RV, Time, T, \pi$ the input parameters for the *DynFloR* algorithm
Ensure: returns the graph G constructed as described in Section 3.1.1
// Create the nodes, edges and their capacities
$addNode(G,s)$ // a node corresponding to the source
for $i \leftarrow 0, n$ **do**
for $t \leftarrow 0, Time$ **do**
$addNode(G,i_t)$ // a node corresponding to robot R_i at time t
if $t \neq 0$ **then**
if $i = 0$ **then**
$addEdge(G,i_{t-1},i_t,\infty)$ // add an edge between 0_{t-1} and 0_t with capacity ∞
else
$addEdge(G,i_{t-1},i_t,MaxCapacity_i)$
end if
end if
end for
if $i \neq 0$ **then**
$addEdge(G,s,i_0,Q_i)$
end if
end for
for each pair $(i, j) \in RV$ **do**
for $t \leftarrow T_{ij}, Time, \pi_{ij}$ **do**
$addEdge(G,i_t,j_t,MaxCapacity_i)$
if $j \neq 0$ **then** //if j is not the sink
$addEdge(G,j_t,i_t,MaxCapacity_j)$
end if
end for
end for
// Add additional edges to model the data dynamically collected by the robots
for $i \leftarrow 1, n$ **do**
for $t \leftarrow 1, Time$ **do**
if $t \ mod \ k_i = 0$ // if an entire cycle was patrolled by robot R_i **then**
// add an edge from the source to i_t for modelling that new amount of data D_i arrives in the node
$addEdge(G,s,i_t,D_i)$
end if
end for
end for
// Return the graph G
return G
end function

transfer data, it should have accumulated 45 units of data, which is impossible. In our model, if the robot's memory is full, the new data will not be collected (instead of overwriting existing data with the new one).

In order to determine which data is lost and which data is in the memory of a robot (and consequently will arrive to the sink in the future), we used the following method: after measuring the flow for time $Time$, as presented in Algorithm 1, we build a new, extended graph for time $2 \cdot Time$. In this extended graph we added the extra edges denoting new data coming from the source to the robots only until time step $Time$, so second half of the graph has no extra data coming. We measure the flow in this extended graph and the difference between the flows denotes the data

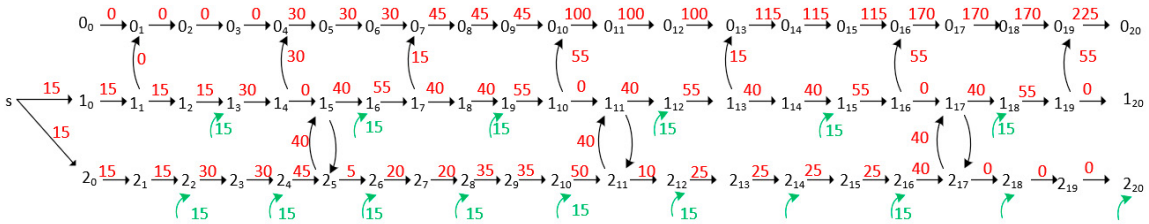


Fig. 3: The dynamic network for the example from Figure 2. The decision making strategy for the robots is marked with red.

that is in the memory of the robots at time $Time$ and will arrive to the sink later. We also know the total data that should enter the graph (the sum of the capacities of the outgoing arcs of the source node) and the difference between this data and the flow of the extended graph is the quantity of the data that is lost. The robot that loses data is determined by the flow algorithm, and in the current implementation we only try to optimize the total quantity of data that arrives to the sink, and do not try to balance the lost quantities between robots.

4. Results and discussion

In this section we present our current results obtained using the algorithm presented previously. The network flow part of our DynFloR algorithm was implemented using the Python NetworkX [11] graph library.

We performed incipient simulations on randomly generated inputs of 20 robots with the same $MaxCapacity$ for each robot. Figures 4 and 5 show the evolution of the $f(T)$ and $g(T)$ objective functions for two different values of $MaxCapacity$. We observe that both functions converge to an equilibrium state and this convergence is independent of the memory capacity. Multiple random experiments confirm this.

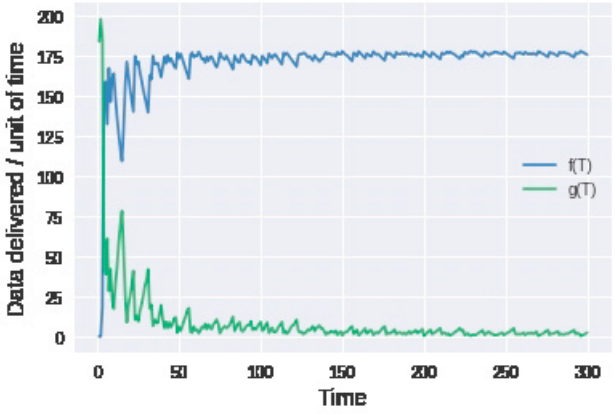


Fig. 4: Results for 20 robots and $MaxCapacity = 1000$.

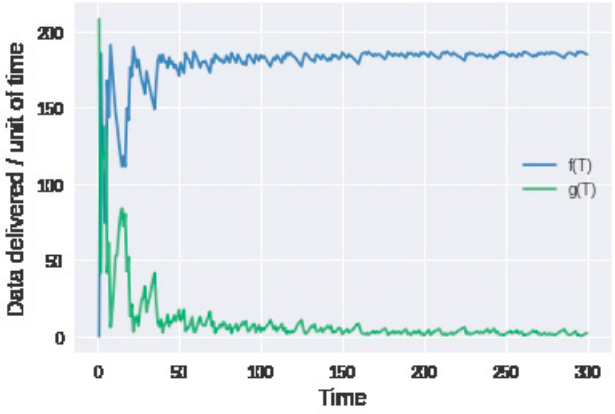


Fig. 5: Results for 20 robots and $MaxCapacity = 800$.

In order to better visualize the relation between data that enters the system, data that is currently in the memory of robots and data that is lost, we have performed some experiments using the simple example from Figure 2, setting the quantity of initial data and new data that is collected after every cycle to 15. We have considered different values for $Time$ and $MaxCapacity$ (but considering the same capacity for both robots). The results are presented in Table 1.

The value in parentheses after $Time$ represents the total quantity of data that enters the system until that time moment and which is divided into data that arrived to the sink, data that is currently in the memory of the robots, and data that is lost. We can see that as the $MaxCapacity$ of the robots increases, so does the amount that arrives to the sink and consequently the quantity of lost data decreases. From Table 1 it looks like setting the $MaxCapacity$ to 60 is enough to make sure that no data is lost, but in order to be certain of this, more experiments and maybe mathematical proof is necessary, because at time 20, there was no lost data for $MaxCapacity$ 55 either, but later this changed. Data

that is currently in the memory of the robots is always bounded by the total memory the robots have, in our case $2 \cdot MaxCapacity$.

Table 1: Quantity of data in memory, at the sink and lost, for different maximum capacities and time limits for the example from Figure 2. Initial data and new data received after every completed cycle is 15.

Time	Maximum capacity	Data arrived at sink	Data in memory	Lost data	Time	Maximum capacity	Data arrived at sink	Data in memory	Lost data
20 (270)	40	180	40	50	40 (420)	40	360	40	125
	45	195	45	30		45	390	45	90
	50	210	45	15		50	420	50	55
	55	225	45	0		55	450	55	20
	60	240	30	0		60	480	45	0
60 (705)	40	525	70	185	80 (1020)	40	730	40	250
	45	570	75	135		45	795	45	180
	50	615	80	85		50	860	45	115
	55	600	85	35		55	925	45	50
	60	705	75	0		60	990	30	0

From Table 1 we can also observe that for a given $MaxCapacity$ the quantity of lost data is not double if we double $Time$. For example, for $MaxCapacity$ 50, we have 15 units of lost data after $Time$ 20, but if we double $Time$, the quantity of lost data becomes 55. We performed some experiments in order to better understand how the quantity of lost data changes if time increases, for a fixed $MaxCapacity$. In Table 2 we included two scenarios: the left part of the table was computed for $MaxCapacity$ 50, while the right part was computed for $MaxCapacity$ 40. For both scenarios, we computed also the rate of Lost data / Time (in the 4th and 8th column).

Table 2: Progression of lost data as time increases for different $MaxCapacity$ values

Maximum capacity	Time	Lost Data	Lost data / Time	Maximum capacity	Time	Lost Data	Lost data / Time
50	10	5	0.5	40	10	25	2.5
	20	15	0.75		20	50	2.5
	50	65	1.3		50	150	3
	100	155	1.55		100	325	3.25
	200	315	1.575		200	650	3.25
	500	815	1.63		500	1650	3.3
	1000	1655	1.655		1000	3325	3.325
	5000	8315	1.663		5000	16650	3.33
	10000	16655	1.6655		10000	33325	3.3325
	50000	83315	1.6663		50000	166650	3.333
	100000	166655	1.66655		10000	333325	3.33325

As expected, Table 2 reveals that the quantity of lost data per time significantly decreases as the maximum memory capacity of the robots increases. Additionally, as illustrated in Figure 6, it seems that the lost data per time converges to an equilibrium state, independent of the maximum capacity of the robots. This can be observed more clearly by running the proposed flow algorithm for more time steps, thus allowing all the data to arrive at the sink. As we have previously discussed, the data which does not arrive to the sink until a certain time limit is not necessarily lost, as it may arrive to the destination if we extend the time limit. Still, there is also the possibility to have lost data, a situation in which the data would never arrive to the sink no matter how long we would extend the time limit. Further experiments and theoretical analyses will be performed in this direction.

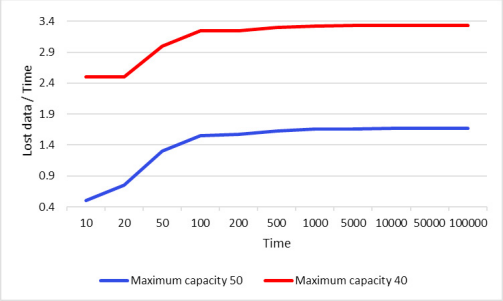


Fig. 6: Variation of lost data per time for the example from Figure 2.

5. Literature review

The literature contains various approaches related to multi-robot patrolling, but we have not found approaches similar to *DynFloR* for optimizing data delivery in MRP. The research in the multi-robot patrolling field is a relatively recent research area, with various contributions which are mainly based on operational research.

Graph based algorithms were used for various patrolling tasks: Hamiltonian cycles for assuring that each point in the target area is covered at the same optimal frequency [9], cyclic strategies which use heuristics to compute TSP cycles [5], path-finding techniques for implementing the decision making strategy for the agents. Alternative approaches in multi-robot patrolling use classical techniques for computational agents coordination in certain environments [19], cooperative auction systems [12] and algorithms based on swarm intelligence [6]. Santana et al. modelled the patrolling problem as a reinforcement learning problem [22], for allowing the agents to automatically adapt to their environment with the goal of optimizing (maximizing or minimizing) a certain performance criterion (e.g. minimizing the nodes idleness). Liemhetcharat et al. [15] approached the problem of foraging and item delivery with multi-robot networks. The authors proposed a formal model for the multi-robot item delivery problem and showed that the continuous foraging problem is a particular case of it. Distributed multi-robot algorithms were also proposed to solve the item delivery and foraging problems and experimented on simulated robots using a Java simulator. Chen et al. [3] investigated a multi-agent patrolling problem in uncertain environments. For dealing with uncertainty and possible threats, the environment was modelled as multi-state Markov chains, whose states are partially observable until the location is visited by an agent. The main goal of the approach from [3] was to maximize the amount of information gathered by the agents while reducing the damage incurred. fFarinelli et al. [1] formalized the problem of coordination (i.e. establishing collaborative interactions between robots to achieve individual and collective goals) as a Distributed Constrained Optimization problem, providing a solution based on the binary max-sum algorithm. The problem of *dynamic multi-robot patrolling* was recently approached by Othmani-Guibourg et al. [17]. The authors proposed a formal model for dynamic environment and on edge-markovian evolving graphs and they introduced and analyzed two strategies for the agents for patrolling in dynamic environments.

Our work differs from all the above-mentioned approaches, as it has a different target than the previously described related work. As far as we know, approaches similar to *DynFloR* do not exist in the MRP literature. The most similar approach to ours is the one of Liemhetcharat et al. [15]. The problem approached in the current paper differs from the one of Liemhetcharat et al. [15], even if both papers are dealing with the data delivery problem. The perspectives are different, since the paper [15] considers data delivery from a central location to several local targets, while in our paper the perspective is the opposite one. A major advantage of *DynFloR* is its polynomial time complexity, compared to the classical brute-force exponential solutions [21]. In addition, *DynFloR* works well for dynamic data, i.e. data which continuously appear on each robot cycle. Another advantage of *DynFlor* is that it finds the global optima, unlike the metaheuristic approaches (such as *genetic algorithms*) [19]) which may provide a local optima. Besides, another difficulty when modelling the MRP using GAs relies on the chromosomes' modelling and on incorporating the continuous data collection into the model.

One of the current limitations of *DynFloR* is given by the assumption regarding the periodicity of the robots' meetings. In the general case, in real MRP scenarios, instead of having initial meeting times and meeting periods, there is an uncertainty of the meetings as the robots' speed is not constant. For offering a solution to the data delivery opti-

mization for the general case, *reinforcement learning* would be further investigated for adapting the robots' decision making policy to the uncertainty of the world.

6. Conclusions and future work

In this paper we have investigated the data delivery optimization in a simplified setting for the *multi-robot patrolling* problem in which the environment is deterministic. We introduced accordingly a centralized and offline approach *DynFloR* based on flows in dynamic networks. The goal of *DynFloR* is to determine the robots' policy for maximizing the quantity of data delivered to a base station in a given time, assuming that the robots are continuously collecting data during the patrolling. The more general aim of the research initiated in this paper is to explore the general case of the problem of optimizing the data delivery in DMRP when the environment is non-deterministic and uncertain (i.e. there are communication failures).

Further work will extend the experimental evaluation and theoretical analysis of *DynFloR* in order to better assess its performance. We also aim to decentralize the decision making process and to incorporate uncertainty in the communication between robots. An extension of *DynFloR* to an online approach and the incoming data model to an irregular one will be subjects to future developments. In order to achieve these goals, a *reinforcement learning* perspective will be considered.

References

[1] Alessandro Farinelli, N.B., Elena Zanotto, E.P., 2017. Advanced approaches for multi-robot coordination in logistic scenarios. Robotics and Autonomous Systems 90, 34–44.
[2] Asis, K.D., Hernandez-Garcia, J.F., Holland, G.Z., Sutton, R.S., 2018. Multi-step reinforcement learning: A unifying algorithm, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), pp. 2902–2909.
[3] Chen, S., Wu, F., Shen, L., Chen, J., Ramchurn, S., 2015. Multi-agent patrolling under uncertainty and threats. PLoS ONE 10, e0130154.
[4] Chevaleyre, Y., 2004a. Theoretical analysis of the multi-agent patrolling problem. IEEE / WIC / ACM International Conference on Intelligent Agent Technolog 00, 302–308.
[5] Chevaleyre, Y., 2004b. Theoretical analysis of the multi-agent patrolling problem. Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004 (IAT 2004) , 302–308.
[6] Chu, H., Glad, A., Simonin, O., Semp, F., Drogoul, A., Charpillet, F., 2007. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation, in: Int. Conf. on Tools with Art. Intelligence, France. pp. 442–449.
[7] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to Algorithms, Third Edition. 3rd ed., The MIT Press.
[8] Daniel, K., Dusza, B., Lewandowski, A., Wietfeld, C., 2009. AirShield: A system-of-systems MUAV remote sensing architecture for disaster response, in: 2009 3rd Annual IEEE Systems Conference, IEEE. pp. 196–200.
[9] Elmaliach, Y., Agmon, N., Kaminka, G.A., 2009. Multi-robot area patrol under frequency constraints. Annals of Mathematics and Artificial Intelligence 57.
[10] Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N.M. (Eds.), 2008. ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings. volume 178 of *Frontiers in Artificial Intelligence and Applications*, IOS Press.
[11] Hagberg, A.A., Schult, D.A., Swart, P.J., 2008. Exploring network structure, dynamics, and function using NetworkX, in: Varoquaux, G., Vaught, T., Millman, J. (Eds.), Proceedings of the 7th Python in Science Conference (SciPy2008), pp. 11–15.
[12] Hwang, K., Lin, J., Huang, H., 2009. Cooperative patrol planning of multi-robot systems by a competitive auction system, in: Proceedings of ICCAS-SICE, Fukuoka, Japan.
[13] Konidaris, G., Kuindersma, S., Grupen, R.A., Barto, A.G., 2012. Robot learning from demonstration by constructing skill trees. I. J. Robotics Res. 31, 360–375.
[14] Kotnyek, B., 2003. An annotated overview of dynamic network flows. Technical Report. INRIA.
[15] Liemhetcharat, S., Yan, R., Tee, K.P., Lee, M., 2015. Multi-robot item delivery and foraging: Two sides of a coin. Robotics 4, 365–397.
[16] NetworkX . Prepush-Flow maximum flow algorithm. https://networkx.github.io/documentation/stable//_modules/networkx/algorithms/flow/maxflow.html.
[17] Othmani-Guibourg, M., El Fallah Seghrouchni, A., Farges, J.L., Potop-Butucaru, M., 2017. Multi-agent patrolling in dynamic environments, in: International Conference on Agents, IEEE digital library , Beijing, China.
[18] Popescu, M.I., Rivano, H., Simonin, O., 2016. Multi-robot patrolling in wireless sensor networks using bounded cycle coverage, in: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 169–176.
[19] Portugal, D., Rocha, R., 2011. A survey on multi-robot patrolling algorithms, in: Technological Innovation for Sustainability, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 139–146.
[20] RoboCup - Rescue, . Building Rescue Systems of the Future . www.robocuprescue.org.
[21] Romeo, M., Banfi, J., Basilico, N., Amigoni, F., 2018. Multirobot Persistent Patrolling in Communication-Restricted Environments. Springer International Publishing. pp. 59–71.
[22] Santana, H., Ramalho, G., Corruble, V., Ratitch, B., 2004. Multi-agent patrolling with reinforcement learning, in: Proceedings of the Third Intern. Joint Conference on Autonomous Agents and Multiagent Systems - Vol. 3, IEEE Computer Society, USA. pp. 1122–1129.