Zia *et al. Complex Adapt Syst Model (2017) 5:1*
DOI 10.1186/s40294-016-0040-9

Complex Adaptive Systems Modeling

SHORT REPORT Open Access

A Cognitive Agent-based Model for Multi-Robot Coverage ata City Scale

Kashif Zia[1*], Ahmad Din[2], Khurram Shahzad[3] and Alois Ferscha[4]

*Correspondence:
kzia@soharuni.edu.om
[1] Faculty of Computing and Information Technology, Sohar University, Sohar, Oman
Full list of author information is available at the end of the article

Abstract

Background: In this article, we model a behavior-based strategy of autonomous coverage and exploration at the scale of a city with multiple robots. The behavioral components are motivated by Cepeda et al. (Sensors 12 (9): 12772–12797, 2012) and extended to incorporate into a generic cellular-automata based agent model. These agents are representing homogenous robots with reactive control. Deliberative approaches requires large scale map and large memory, which slowdowns the execution. Our approach is reactive and simple, that is, robots have no prior information about the environment and do not generate a route map as they traverse. However, other robots in neighborhood are detected using local sensors.

Findings: A city-scale map-driven simulation is designed and model's efficiency is evaluated for different deployment possibilities. It is evidenced that even with this simple model, the agents are able to explore a significant percentage of the environment.

Conclusion: For a city-scale multi-robotic exploration, our simple but efficient model does not require explicit communication and data sharing (and hence representation and storage of navigated map) because possibility of encountering and influencing another agent is quite low, due to spatial dynamics of the environment.

Keywords: Multi-robotic, Distributed coverage problem, Agent based models

Background

Multi-robot system (MRS) has emerged as a product of cheap sensing and actuating capabilities of small, and moderately sophisticated robots and advancements in distributed problem-solving. In application domains such as sweeping, distribution and exploration of an unknown environment, MRS usually have mobile and autonomous participating robots, performing the task cooperatively.

It has been reported that exploration and coverage is a fundamental problem in MRS González-Banos and Latombe (2002). In the coverage problem, the robots cooperatively sweep the unknown environment, possibly searching for a goal (Lidoris et al. 2009). Hence, these two terms are related with each other and can be used interchangeably. In Cepeda et al. (2012), authors have defined the main goal of robotic exploration, relating exploration and coverage with each other, as minimizing "the overall time for covering an unknown environment".

The most acceptable mechanism of distributed exploration in an unknown environment is based on a greedy approach Yamauchi (1998), in which a *frontier* draws the

boundary between known and unknown space, and a *next-best-view* (González-Banos and Latombe 2002) is chosen across the frontier. It is evident that the real problem in exploration is the efficient distribution of the frontier areas to a pool of robots. There are different perspectives that can be used to address this problem, including economy-based negotiations (Sheng et al. 2006), cost-benefit analysis (Burgard et al. 2005), and environment-driven considerations (such as spatial features and capabilities of robots) Stachniss et al. (2008), Rooker and Birk (2007).

However, such concise algorithmic approaches are not suitable for unstructured and dynamic environments (Cepeda et al. 2012), such as cities. The city-scale multi-robot coverage have many potential applications, such as sweeping (garbage-collection, mine sweeping, pesticide and water spray etc.), searching (rescue operations, ammunition and drug search etc.), and data gathering (futuristic Internet of Things applications and transportation etc.).

In this article, we consider the problem of large scale coverage of unknown environments using MRS, where simple robots are modeled as the agents in a multi-agent system (MAS) Macal and North (2005). Using MAS, we use a bottom-up approach, in which local behavior (without global knowledge, optimum plan and permanent storage) of agents converges into global exploration.

The motivation of our mechanism is from Cepeda et al. (2012), in which, a behavior based strategy of multi-robot exploration is presented. The authors have introduced four behaviors, named as "exploration", "dispersion", "obstacle avoidance", and "avoiding past". The mechanism relies on locating an open area along the frontier during exploration phase. The robots have limited memory so that they can just remember recently visited areas, and avoid them during exploration. The obstacle avoidance is achieved through sensing the obstacles within a range, and collision avoidance is achieved through simple mechanism of dispersion. The mechanism was simulated for a regular space of 200 m^2.

In the mechanism presented in this paper, we have generalized the robot's behavior [presented in Cepeda et al. (2012)] into an agent's behavior, and extended the model in such a way that it has more explicit representation of space and sensed parameters. Hence, our model can be used with any of the agent modeling simulator which has a space representation in cellular automata. Additionally, our model requires much simpler sensing requirements.

However, the main contribution of our work is simulation of the model at a very large scale. The simulation is based on real GIS map of a city spanned across several kilometers of space and synchronous update scheme has been used. We have also set up two deployment scenarios of the agents [(i) a bunch of agents deployed at the center of the city, and (ii) agents individually dispersed at four corners of the city] for evaluation of the exploration efficiency, comparatively. Due to the stark contrast between the number of the agents and the size of the space, we also evaluate the potential of agent-to-agent interaction towards the improvement in exploration efficiency. Hence, no explicit communication is required, example of explicit communication is to share history, which can be useful in small environments. For large environments, it cannot be useful because agents refresh their memory after certain time due to limited memory available onboard. On other hand robots are equipped with local sensors, which are used to detection other

robots in its communication zone. Mean position is computed using the information of robots in the neighborhood coming from the local sensors.

This paper is structured as follows: first, related work is detailed in "Related work" section; followed by the model in "Model" section; "Simulation and results" section focus on simulations and results; and finally, we conclude the paper in "Conclusions and outlook" section, including an outlook.

Related work

An extensive survey of exploration and coverage approaches using a multi-robotic system has been presented in Julia et al. (2012) and Galceran and Carreras (2013).

Several researchers have investigated potential field approach for decentralized multi-robotic control and coverage (Baxter et al. 2007). In the potential field approach, robots feels force of attraction for the goal (e.g. frontier), and repulsion for other robots and the obstacles. Main limitation of this technique is the presence of the local minima, which may restrain robots from exploring the environment completely. This problem was resolved in Renzaglia and Martinelli (2010), by introducing a leader in the team with different control law, i.e. frontier based exploration, while the followers are potential-field driven. Similarly, in Zheng et al. (2010), the authors have proposed Space-Based Potential Field technique, which focus on dispersion of robots in the environment, while avoiding overlap performance and the local minima. Recently decentralized variation of this technique (Liu and Lyons 2015) is proposed, which includes monotonic factor for coverage to avoid local minima problem.

Other techniques include physics based coverage mechanism, which exploits the kinetic energy for the coverage task. In Spears et al. (2006), authors have proposed physics based coverage algorithm to control multi-robots for exploration. It requires limited communication, sensing and global knowledge. This algorithm is inspired by the motion of gas particles, which is effective to cover the unstructured environments. This work was extended by Maxim and Spears (2010) which presented physics based uniform coverage algorithm for the environments that contain non-convex regions.

Many of the multi-robot coverage algorithms are inspired by nature. Various nature-tested mechanisms, which are proven to be successful in biological systems, have been effective for robotics as well. In Florea et al. (2015), the authors took inspiration from ants and presented ant-foraging based approach, in which the ants leave pheromone-like scent marker in the environment for coordination in coverage process. Other bio-mimetic approach includes the swarm based coverage techniques (Kantaros et al. 2015). Swarm is also used with behavior-based social interactions (Şahin 2005). In Cepeda et al. (2012), a behavior based technique for multi-robot coverage is presented. A set of simple behaviors are defined for exploration, therefore, no explicit target location is required. Rather a local spatial information memory is combined with the finite state automata to ensure the dispersion of the robots over the environment. Major issue in this approach is the communication and synchronization during exploration to avoid already visited cells and to choose an appropriate direction. In Vizzari et al. (2013), dispersion behavior of a group is presented, which preserves cohesion in group of pedestrians instead of area coverage.

One of the first multi-robotic exploration technique was developed (Rekleitis et al. 1997, 1998) using rule- and heuristic-based approach. Simple robots with limited

computation, memory and communication were used to cover unknown terrain collaboratively. Robots were divided into two teams; the first team explored the area, while the second team provided relative measurements, hence acting like a landmark. Apparently, this technique was not truly autonomous and distributed and required quite a lot of resources and time. Another team based approach was proposed in Cohen (1996), in which collaborating teams built a map of an arena using probabilistic techniques in semi-distributed fashion Simmons et al. (2000). In Batalin and Sukhatme (2002), a simple heuristic-based coverage algorithm was proposed to disperse the robots in the environment. But for effective navigation of the robots and coordination, efficient localization and mapping techniques were required.

Another probabilistic, but centralized approach for coverage was introduced in Burgard et al. (2005) that assigned goals to the robots based on their localization and the closet frontier cell. It probabilistically tried to balance the travel cost to the unexplored area and the utility gained. Frontier-based coverage algorithm was originally presented in Yamauchi (1998), where grid maps were built from the sensor input and the algorithm computed the frontier between the robot location and the unexplored cells. This work was extended to a more generalized and distributed mechanism of building a global map (Birk and Carpin 2006), in which the robots moved to the nearest frontier cell, while avoid visiting the frontier cells traversed by other robots. A more generalized cell-based research flourished as grid-based coverage techniques, in which environment was divided into a set of uniform grid cells. In such an environment, a Multi-Robot Spanning Tree Coverage approach was introduced in off-line (Hazon and Kaminka 2008) as well as on-line (Hazon et al. 2006) mode. In Zheng et al. (2005), the authors built a variation of this method called multi-robot forest coverage, in which the grid cells were kept large and these cell were further divided into sub-cells.

In this article, we consider the problem of large scale coverage of unknown environments using multi-robotic system. There are very few research efforts in this domain. For example, the Robocop urban search and rescue (USAR) aims to promote research in collaborative problem solving by multiple robots in a city scale arenas (Visser et al. 2015). However, the emphasis of this competition is on cooperative mapping, search and rescue, disaster management, coordination etc., rather than coverage and exploration. Hence, the notion of application overrides the notion of coverage. We have opted a entirely opposite approach, in which coverage is the primary function, keeping in view the possible applications of it.

The city-scale multi-robot coverage have many potential applications, but they share the same operational characteristics:

- The robots use autonomous and decentralized approach and have the same role (no concepts of diversified teams).
- The robots main activity is mobility, requiring much resources, hence depleting resources for storage and processing. Hence, we cannot utilize (unlimited) occupancy-based and stored maps, or even potential maps.
- The robots are not many when compared with the size of the world. Hence, explicit communication is ignored and local sensors are used to detect other robots in the communication zone.

These characteristics are the building blocks of the mechanism presented in this paper.

Model

As shown in Fig. 1, the decision-making process consists of four behaviors; Disperse, Explore, Avoid Obstacle and Avoid Past; all destined to change the direction of motion of the agent, possibly providing a New Heading to be used in the next Move. Each agent, in each iteration, starts with the dispersion phase, followed by the exploration phase. If the agent is in an area where avoiding an obstacle is required, then avoiding the obstacle is behaviorally prioritized over coverage extension, i.e. motive of avoiding recent past before the actual move. After the move, the history constituted by the most recently visited cells is updated.

Dispersion

Algorithm 1 Disperse

```
1: if Should_Avoid_Obstacle? then
2:     Do nothing
3: else
4:     neighbors ← agentset in radius equal to COMFORT-ZONE
5:     if count(neighbors) == 0 then
6:         Do nothing
7:     else
8:         meanPosition ← mean position of neighbors
9:         if distance to meanPosition < DEAD-ZONE then
10:            Apply_Fixed_Translation
11:        else
12:            angleTo ← angle towards meanPosition
13:            if angleTo < 30 then
14:                newHeading ← heading + deviation {deviation is +45 or −45}
15:                heading ← Set_Direction (newHeading)
16:            end if
17:        end if
18:    end if
19: end if
```

The algorithm of dispersion behavior is given in Algorithm 1. If the agent is in an area where avoiding an obstacle is required, then avoiding the obstacle, gets priority (in fact,

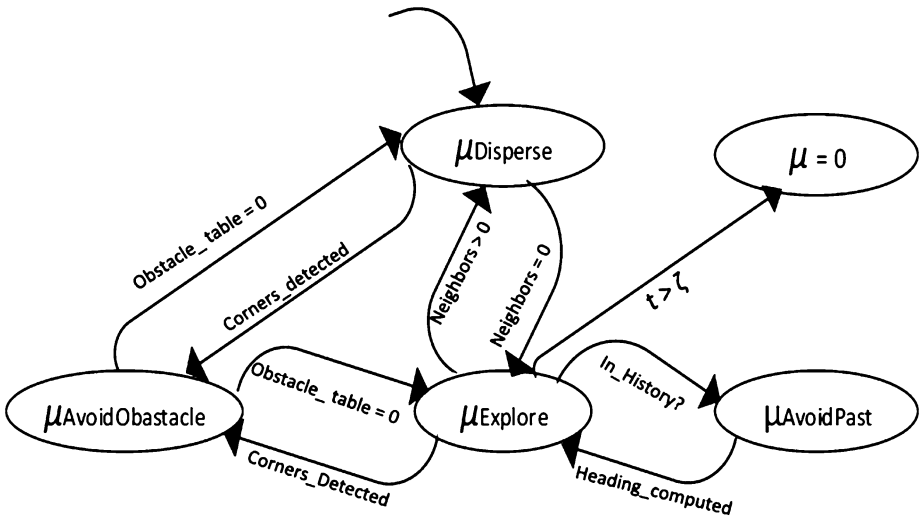

Fig. 1 Agent behavioral flow: performed at each time-stamp of discrete time simulation

leaving it for its turn) over dispersion. The dispersion behavior depends on *neighbors* of the agent in its COMFORT-ZONE (settable global variable). If there is no agent in this zone, there is no need to disperse. Otherwise, the dispersion depends on *meanPosition* of the *neighbors*. If the *meanPosition* is too close (within a settable DEAD-ZONE), a fixed translation is applied. If fixed translation is not required, the angle towards the *meanPosition* is calculated. This angle should not be too less (<30° where 0 represents no difference in current heading and angle towards the *meanPosition* of neighbors). But at the same time, it should not be too much (>45°). This assures that neither the agent bumps into the bunch of neighbors, nor, the agent acquires an entirely different direction from its original direction of motion as a result of dispersion.

The Set Direction procedure sets the ranging of a calculated intended angle keeping it within allowable limits of 0° to 359°. The Apply Fixed Translation inverts the heading, moves the agent x time speed, thus placing it at probably a vacant location, updating the History, and, inverting the direction again, so that it keeps moving in the same direction, after being sufficiently dispersed.

A procedure Detect Corners acts as the basis of procedure Should Avoid Obstacle and Avoid Obstacle behavior. For each of the possible orientations; (i) front (relative angle equal to 0 between current heading and the obstacle), (ii) left (relative angle equal to 45 between current heading and the obstacle on the left), and (iii) right (relative angle equal to 45 between current heading and the obstacle on the right); a table is populated indicating the obstacle's direction and distance. Starting from a distance equal to 1 up to settable CORNER-DISTANCE, an obstacle is identified as soon as it is detected. For example, the table may have an entry, "front, 3", which represents that there an obstacle at front and at a distance 3. The table may have another entry for left, and no entry for right.

The Should Avoid Obstacle procedure just transforms state of the table returned by the procedure Detect Corners into a boolean value; true, if length of the table is greater than 0, false, otherwise.

Exploration

Algorithm 2 Explore (Locate Free Space)

```
1:  ArrayObstacles [5]
2:  i ← 1
3:  while i < INTENDED-DISTANCE do
4:
5:      if orientation 45 i is not walkable then
6:
7:          if ArrayObstacles [x] >=MAX then
8:              ArrayObstacles [x] = i
9:          end if
10:     end if
11:
12:     if patch-right-and-ahead 45 i is not walkable then
13:
14:         if ArrayObstacles [y] >=MAX then
15:             ArrayObstacles [1] = i
16:         end if
17:     end if
18:
19:     if patch-left-and-ahead 90 i is not walkable then
20:
21:         if ArrayObstacles [y] >=MAX then
22:             ArrayObstacles [y] = i
23:         end if
24:     end if
25:
26:     if patch-right-and-ahead 90 i is not walkable then
27:
28:         if ArrayObstacles [3] >=MAX then
29:             ArrayObstacles [3] = i
30:         end if
31:     end if
32:
33:     if patch-ahead i is not walkable then
34:
35:         if ArrayObstacles [4] >=MAX then
36:             ArrayObstacles [4] = i
37:         end if
38:     end if
39: end while
40: i ← index of maximum value of ArrayObstacles
41: if i = 0 then
42:     heading ← Set_Direction (heading − 45)
43: end if
44: if i = 1 then
45:     heading ← Set_Direction (heading + 45)
46: end if
47: if i = 2 then
48:     heading ← Set_Direction (heading − 90)
49: end if
50: if i = 3 then
51:     heading ← Set_Direction (heading + 90)
52: end if
53: if i = 4 then
54:     heading ← Set_Direction (heading + 0)
55: end if
```

Exploration is not performed in each iteration; instead after expiry of WANDERING-LIMIT. Each agent locates free space compatible to its orientation as shown in Algorithm 2. Five orientations are explored relative to current heading of the agent; which are, 45° to the left and right, 90° to the left and right, and 0° (i.e. same as current heading). The array *Array-Obstacles* stores the minimum walkable distance before an obstacle is encountered in these five directions. The array is initialized with a value sufficiently large at each index.

The array is populated with distance in a specific direction starting from i equal to 1 to a settable variable INTENDED-DISTANCE. As soon as an obstacle is found, the array element against that direction is updated with the distance, indicated by while loop index. Let i represents the index of the array against which the maximum number is stored (it can be at the most the initialization value that was never updated, indicating a free space in that direction without any obstacle within INTENDED-DISTANCE), the heading of the agent is updated corresponding to that direction.

Obstacle avoidance

The detectCorner procedure returns a table indicating the obstacles' distances at "front", "right" and "left". If the table is empty, there is no obstacle to avoid. Otherwise, the heading of the agent is inverted in case of obstacle at front. If there is no obstacle at front, but there is an obstacle towards left or right, the heading changes diagonally accordingly, where left have preference over right (with no particular reason). The algorithm of obstacle avoisance is given in Algorithm 3.

Algorithm 3 Avoid Obstacle

```
corners ← detectCorner()
if length of table corners > 0 then
    if table corners have key = front then
        heading ← Set_Direction (heading + 180)
    else
        if table corners have key = left then
            heading ← Set_Direction (heading + 45)
        else
            if table corners have key = right then
                heading ← Set_Direction (heading − 45)
            end if
        end if
    end if
end if
```

Avoid past

The algorithm of avoiding past is given in Algorithm 4. The procedure uses a counter *freeSlot* to keep track of number of cells in the History table. After reaching to a maximum value equal to HISTORY-PERIOD, the counter is reset to zero. After each move, freeSlot is incremented and visited cell is stored in HISTORY-PERIOD against the index equal to freeSlot. So, practically this procedure executes after multiple of freeSlot's maximum value, to avoid frequent changes in agent's orientation. If the procedure is executed, the following is the mechanism which is followed, given that procedures used are obvious and easily understood by their names. If the *patch-ahead* (Netlogo routine) is *notOccupied* and *notInHistory*, then there is no need to change the heading. Else, if, the patch on the left and at the diagonal angle, is *notOccupied* and *notInHistory*, the heading is changed towards that patch. Else, if, the patch on the right and at the diagonal angle, is *notOccupied* and *notInHistory*, the heading is changed towards that patch. If all three directions are occupied, the heading of the agent is orthogonally inverted.

Algorithm 4 Avoid Past

```
if freeSlot < HISTORY − PERIOD then
    Do Nothing
else
    if notInHistory (patch-ahead speed) and notOccupied (patch-ahead speed) then
        Do nothing
    else
        if notInHistory (patch-left-and-ahead 45 speed) and notOccupied (patch-left-and-ahead 45 speed) then
            heading ← heading - 45
        else
            if notInHistory (patch-right-and-ahead 45 speed) and notOccupied (patch-right-and-ahead 45 speed)
            then
                heading ← heading + 45
            else
                heading ← Set_Direction (heading + 180)
            end if
        end if
    end if
end if
```

Behavior selection

The behavior selection is a mechanism for coordination of behaviors and it decides which behavior will get control in a given situation. The behavior selection is achieved for autonomous exploration using finite state automata (FSA) shown in Fig. 1. Initial state is *dispersion* which ensures if robots in a very near (in a DEAD ZONE) then it disperses them, if obstacle is detected during this behavior, *avoid obstacle* behavior is activated to ensure safe navigation of robot during dispersion. If there is no robot in a dead zone then *explore* behavior is triggered which locate largest free space available for navigation. During this behavior to avoid revisits of same location *avoid past* behavior is triggered. Simulation terminated after *specific* time stamps.

Simulation and results

The simulation was performed in agent modeling tool NetLogo (Seth and Wilensky 2004). Shape files (representing points, lines and polygons) of the center of a Central European city were used to generate the simulated World, using the GIS support provided in Netlogo. For simplicity, various structural features of the city such as buildings, open-spaces and greenery were categorized as places where one cannot walk. Only pathways, streets and roads were considered walking-friendly. Thus, each patch (cell) of the Netlogo World stored this information in a boolean variable "walkable?". The patch size is equal to $2/3 \times 2/3$ m^2, and there are 220,452 walkable patches and 609,609 patches are not walkable.

Three scenarios were simulated:

1. Single: single agent starting at the center of the city.
2. FourCen: four agents staring at the center of the city.
3. FourDis: four agents distributed across four corners of the city.

For each scenario, the discrete-time simulation was run for 12 h; a logical time in which each simulation step represent a single second. In each time-stamp, an agent performs a process represented in Fig. 1.

Figure 2 presents a snapshot of the simulation after 500 iterations (around 8 min). The highlighted section of larger map (at top) is updated below with extend of explored space (in black color) in case of Single (left), FourCen (middle) and FourDis (right) scenario. A more quantitative analysis is presented in Fig. 3. It is evident from the graph that the case FourDis outperforms other two cases. The coverage is around 40% in FourDis case and around 36% in FourCen case. However, this does not represent the actual picture. Due to vastness of the space, many cells remain uncovered even though an agent has traversed through the street in which they lie, due to designation of only those cells as covered who are within a specified range. If a street having traversed is the criteria of coverage, then the coverage percentage is much higher, as evident from Fig. 4.

Another aspect that we explored that what is the percentage of cells which are discovered more than once. A quantitative analysis is presented in Fig. 5. A re-discovery rate reaching up to 20% is not a good outcome, but it is inevitable in a memory-less mechanism that we have.

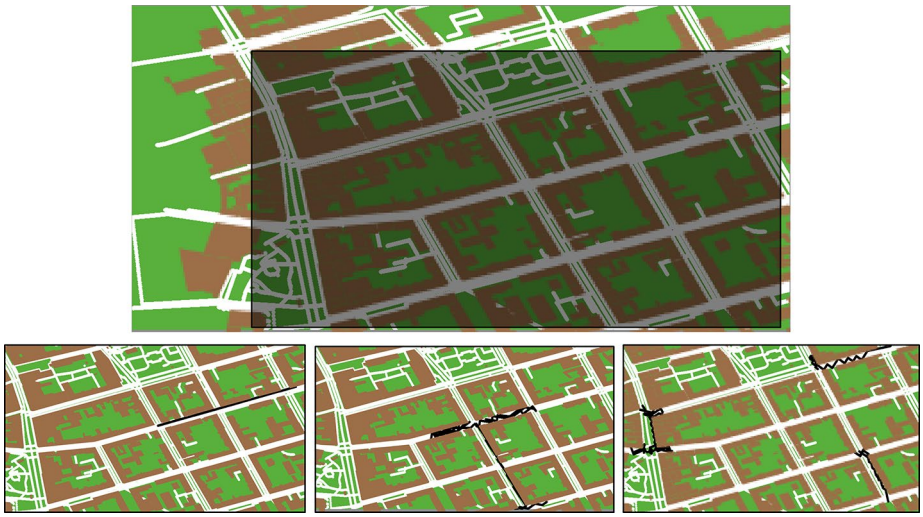

Fig. 2 The part of the city map simulated (*above*). *Below* Explored space after 500 iterations, single (*left*), FourCen (*middle*), and FourDis (*right*)

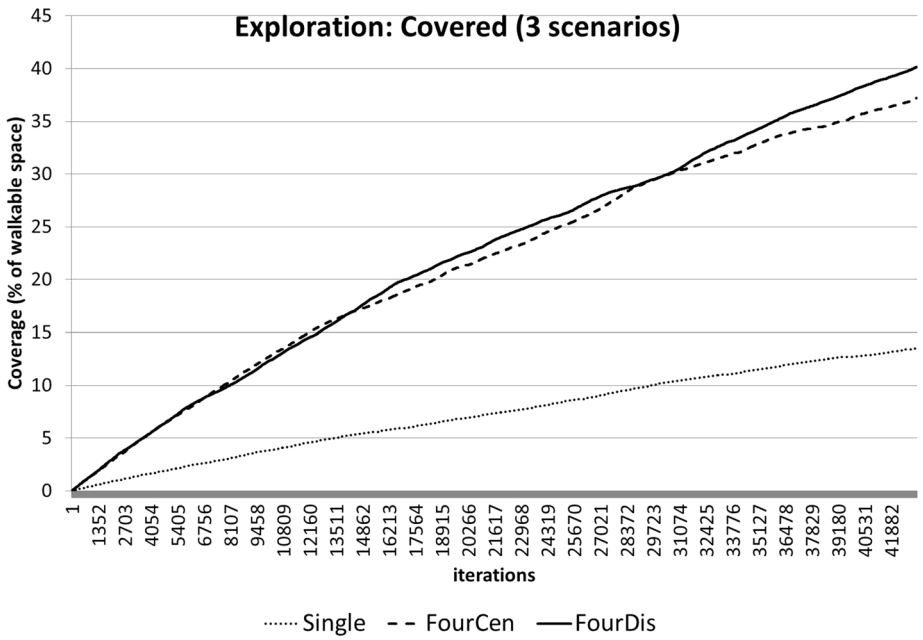

Fig. 3 Comparison of 3 scenarios—covered space

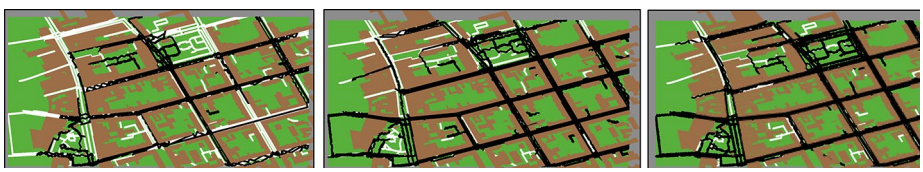

Fig. 4 Explored space after 12 h (end of simulation), Single (*left*), FourCen (*middle*), and FourDis (*right*)

Finally, we also analyzed the number of times two agents came close to each other, to judge the potential of information sharing. It was just 78 time (just above 1 min) during simulation of 12 h. Even during such sparse encounters, 80% of time the agents did not change their direction drastically, even when the dispersion was applied.

Conclusions and outlook

In this article, the proposed solution for multi-robots coverage problem is analyzed based on the results obtained through extensive simulations. The advantage of simulating the model at the scale of a city is that, we do not need explicit evaluation for different environmental features, such as large open spaces, narrow cluttered environments, dead-end corridors, and rooms in a building. A city have all these environmental features embedded within itself. The solution comprising of four sub-behaviors namely, dispersion, exploration, obstacle avoidance and past avoidance, and produces significant efficiency for an agent with such a simple function.

The behavior of the agent can easily be mapped on to real life robots for practical applications. For dispersion and exploration behaviors, robots are required to be able to move from one position to another, which is a basic characteristic implemented in almost all robots and, therefore, does not present any special design/implementation challenge. For obstacle avoidance, a robot shall be capable of detecting obstacles from a safe distance as well as capable of deciding new course of direction. The former characteristic can be achieved through laser and/or sonar sensors that allow to the determine distances from obstacle based on the transmitted and received waves from a sensor. Once the distance to an obstacle is determined an intelligent controller (typically in the form a micro-controller or a micro-processor) could be used to decide new direction for exploration. As each robot would be equipped with an intelligent controller and memory

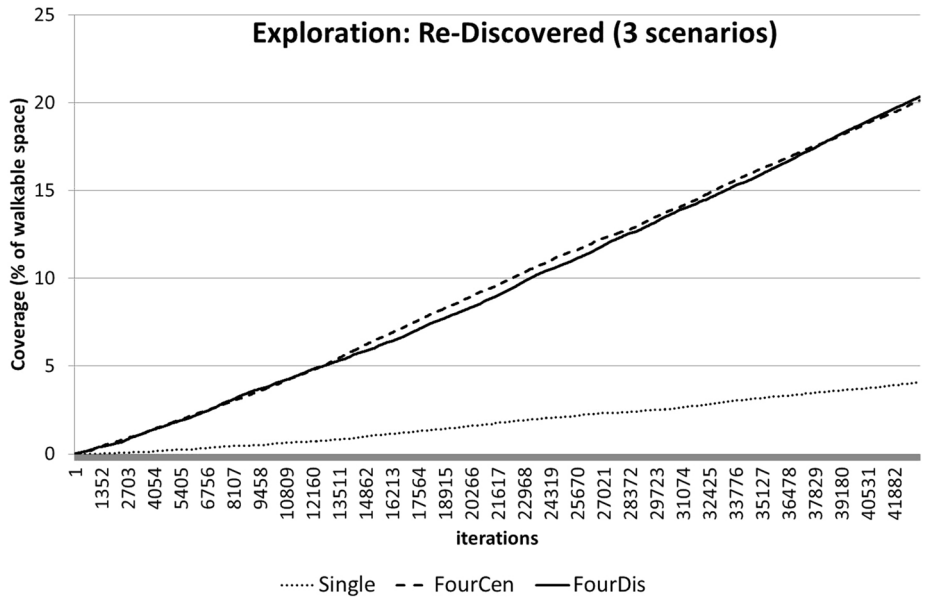

Fig. 5 Comparison of 3 scenarios—re-discovered space

module, avoiding past could be implemented by keep track of each visited cell in the memory.

A robot with above mentioned characteristic could be either designed/implemented from the scratch so as to customize the other requirements such as robustness, cost, etc. for a given practical application or commercially available robots for scientific research and practical applications purposes could be purchased for validating the simulation results and for deployment of robots for an target application, respectively. One limitation of this study is simulations instead of real robots but the experiments in the physical environments are complex and difficult to scale for large scale arenas using real robots; secondly such experiments are not possible with limited finance.

Abbreviations

MRS: multi-robot systems; MAS: multi-agent system; GIS: geographic information system; USAR: Urban Search and Rescue; FSA: finite state automata.

Authors' contributions

This has been a teamwork and all authors have contributed equally. All authors read and approved the final manuscript.

Author details

[1] Faculty of Computing and Information Technology, Sohar University, Sohar, Oman. [2] Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan. [3] Department of Computer Engineering, Bahria University, Islamabad, Pakistan. [4] Institute of Pervasive Computing, Johannes Kepler University, Linz, Austria.

Competing interests

The authors declare that they have no competing interests.

Received: 13 May 2016 Accepted: 10 December 2016
Published online: 03 January 2017

References

Batalin MA, Sukhatme GS (2002) Spreading out: a local approach to multi-robot coverage. In: Proceedings of 6th international symposium on distributed autonomous robotic systems, pp 373–382

Baxter JL, Burke E, Garibaldi JM, Norman M (2007) Multi-robot search and rescue: a potential field based approach. In: Autonomous robots and agents. Springer, Berlin, pp 9–16

Birk A, Carpin S (2006) Merging occupancy grid maps from multiple robots. Proc IEEE 94(7):1384–1397

Burgard W, Moors M, Stachniss C, Schneider FE (2005) Coordinated multi-robot exploration. IEEE Trans Robot 21(3):376–386

Cepeda JS, Chaimowicz L, Soto R, Gordillo JL, Alanís-Reyes EA, Carrillo-Arce LC (2012) A behavior-based strategy for single and multi-robot autonomous exploration. Sensors 12(9):12772–12797

Cohen WW (1996) Adaptive mapping and navigation by teams of simple robots. Robot Auton Syst 18(4):411–434

Florea B-F, Grigore O, Datcu M (2015) Pheromone averaging exploration algorithm. In: International conference on advanced robotics (ICAR) 2015. IEEE, pp 617–622

Galceran E, Carreras M (2013) A survey on coverage path planning for robotics. Robot Auton Syst 61(12):1258–1276

González-Banos HH, Latombe J-C (2002) Navigation strategies for exploring indoor environments. Int J Robot Res 21(10–11):829–848

Hazon N, Kaminka GA (2008) On redundancy, efficiency, and robustness in coverage for multiple robots. Robot Auton Syst 56(12):1102–1114

Hazon N, Mieli F, Kaminka G, et al (2006) Towards robust on-line multi-robot coverage. In: Proceedings 2006 IEEE international conference on robotics and automation, ICRA 2006. IEEE pp 1710–1715

Julia M, Gil A, Reinoso O (2012) A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Auton Robots 33(4):427–444

Kantaros Y, Thanou M, Tzes A (2015) Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints. Automatica 53:195–207

Lidoris G, Rohrmüller F, Wollherr D, Buss M (2009) The autonomous city explorer (ace) project—mobile robot navigation in highly populated urban environments. In: IEEE international conference on robotics and automation, ICRA'09. IEEE, pp 1416–1422

Liu T-M, Lyons DM (2015) Leveraging area bounds information for autonomous decentralized multi-robot exploration. Robot Auton Syst 74:66–78

Macal CM, North MJ (2005) Tutorial on agent-based modeling and simulation. In: Proceedings of the 37th conference on winter simulation. Winter simulation conference, pp 2–15

Maxim PM, Spears WM (2010) Robotic uniform coverage of arbitrary-shaped connected regions. Technical report, DTIC Document

Rekleitis I, Dudek G, Milios E (1998) Accurate mapping of an unknown world and online landmark positioning. Proceedings of vision interface (VI), pp 455–461
Rekleitis IM, Dudek G, Milios EE (1997) Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In: International joint conference on artificial intelligence, vol 15. Lawrence Erlbaum Associates Ltd, pp 1340–1345
Renzaglia A, Martinelli A (2010) Potential field based approach for coordinate exploration with a multi-robot team. In: IEEE international workshop on safety security and rescue robotics (SSRR), 2010. IEEE, pp 1–6
Rooker MN, Birk A (2007) Multi-robot exploration under the constraints of wireless networking. Control Eng Pract 15(4):435–445
Şahin E (2005) Swarm robotics: from sources of inspiration to domains of application. In: Swarm robotics. Springer, Berlin, pp 10–20
Seth T, Wilensky U (2004) Netlogo: a simple environment for modeling complexity. In: International conference on complex systems, vol 21, pp 16–21
Sheng W, Yang Q, Tan J, Xi N (2006) Distributed multi-robot coordination in area exploration. Robot Auton Syst 54(12):945–955
Simmons R, Apfelbaum D, Burgard W, Fox D, Moors M, Thrun S Younes H (2000) Coordination for multi-robot exploration and mapping. In: AAAI/IAAI, pp 852–858
Spears D, Kerr W, Spears W (2006) Physics-based robot swarms for coverage problems. Int J Intell Control Syst 11(3):11–23
Stachniss C, Mozos ÓM, Burgard W (2008) Efficient exploration of unknown indoor environments using a team of mobile robots. Ann Math Artif Intell 52(2–4):205–227
Visser A, Ito N, Kleiner A (2015) Robocup rescue simulation innovation strategy. In: RoboCup 2014: robot World Cup XVIII. Springer, Berlin, pp 661–672
Vizzari G, Manenti L, Crociani L (2013) Adaptive pedestrian behaviour for the preservation of group cohesion. Complex Adapt Syst Model 1(1):1
Yamauchi B (1998) Frontier-based exploration using multiple robots. In: Proceedings of the second international conference on Autonomous agents. ACM, pp 47–53
Zheng X, Jain S, Koenig S, Kempe D (2005) Multi-robot forest coverage. In: 2005 IEEE/RSJ international conference on intelligent robots and systems (IROS 2005). IEEE, pp 3852–3857
Zheng X, Koenig S, Kempe D, Jain S (2010) Multirobot forest coverage for weighted and unweighted terrain. IEEE Trans Robot 26(6):1018–1031