POLITECNICO DI MILANO
Corso di Laurea (MAGISTRALE ?) in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

TITOLO DELLA TESI

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Ing. Matteo Matteucci
Correlatore: Prof. Andrea Bonarini

Tesi di Laurea di:
Bud Spencer, matricola 634845
Terence Hill, matricola 641067

A qualcuno...

Sommario

Il sommario deve contenere 3 o 4 frasi tratte dall'introduzione di cui la prima inquadra l'area dove si svolge il lavoro (eventualmente la seconda inquadra la sottoarea più specifica del lavoro), la seconda o la terza frase dovrebbe iniziare con le parole "Lo scopo della tesi è ..." e infine la terza o quarta frase riassume brevemente l'attività svolta, i risultati ottenuti ed eventuali valutazioni di questi.

NB: se il relatore effettivo è interno al Politecnico di Milano nel frontesizo si scrive Relatore, se vi è la collaborazione di un altro studioso lo si riporta come Correlatore come sopra. Nel caso il relatore effettivo sia esterno si scrive Relatore esterno e poi bisogna inserire anche il Relatore interno. Nel caso il relatore sia un ricercatore allora il suo Nome COGNOME dovrà essere preceduto da Ing. oppure Dott., a seconda dei casi.

Ringraziamenti

Ringrazio .....

Chapter 1

Introduzione

"Terence: Rotta a nord con circospezione
Bud: Ehi, gli ordini li do io qui!
Terence: Ok, comante
Bud: Rotta a nord
Terence: Soltanto?
Bud: Con circospezione!"

Chi Trova un Amico Trova un Tesoro

L'introduzione deve essere atomica, quindi non deve contenere nè sottosezioni nè paragrafi nè altro. Il titolo, il sommario e l'introduzione devono sembrare delle scatole cinesi, nel senso che lette in quest'ordine devono progressivamente svelare informazioni sul contenuto per incatenare l'attenzione del lettore e indurlo a leggere l'opera fino in fondo. L'introduzione deve essere tripartita, non graficamente ma logicamente:

1.1 Inquadramento generale

La prima parte contiene una frase che spiega l'area generale dove si svolge il lavoro; una che spiega la sottoarea più specifica dove si svolge il lavoro e la terza, che dovrebbe cominciare con le seguenti parole "lo scopo della tesi è ...", illustra l'obbiettivo del lavoro. Poi vi devono essere una o due

1.2 Breve descrizione del lavoro

La seconda parte deve essere una esplosione della prima e deve quindi mostrare in maniera più esplicita l'area dove si svolge il lavoro, le fonti bibliografiche più importanti su cui si fonda il lavoro in maniera sintetica (una pagina) evidenziando i lavori in letteratura che presentano attinenza con il lavoro affrontato in modo da mostrare da dove e perché è sorta la tematica di studio. Poi si mostrano esplicitamente le realizzazioni, le direttive future di ricerca, quali sono i problemi aperti e quali quelli affrontati e si ripete lo scopo della tesi. Questa parte deve essere piena (ma non grondante come la sezione due) di citazioni bibliografiche e deve essere lunga circa 4 facciate.

1.3 Struttura della tesi

La terza parte contiene la descrizione della struttura della tesi ed è organizzata nel modo seguente. "La tesi è strutturata nel modo seguente.

Nella sezione due si mostra ...

Nella sez. tre si illustra ...

Nella sez. quattro si descrive ...

Nelle conclusioni si riassumono gli scopi, le valutazioni di questi e le prospettive future ...

Nell'appendice A si riporta ... (Dopo ogni sezione o appendice ci vuole un punto)."

I titoli delle sezioni da 2 a M-1 sono indicativi, ma bisogna cercare di mantenere un significato equipollente nel caso si vogliano cambiare. Queste sezioni possono contenere eventuali sottosezioni.

Chapter 2

Problem setting and definition

2.1 Exploration problem

The exploration problem can be defined as the problem of mapping an unknown environment by means of a team of robots. Thus, the objective is the production of a map of the environment but this introduces further problems, concerning for example the localization of each robot dealing with the various sources of uncertainty, how to distribute the team and how to coordinate them. These are three of the main aspects considered in the development of a solution to this problem, providing the major impact on performances. To compare different approaches the main metric used is the time taken by the exploration [3, 10, 4], and also the distance traveled by the robots is sometimes considered [11].

A formal definition of the *exploration problem*, in the following abbreviated as *EP*, can be provided through a 4-ple $\langle A, E, P_0, T \rangle$ where:

- A is the set of robots used for the exploration;
- E is the environment to explore;
- P_0 is a vector of the initial poses of the team;
- T is the termination criterion.

This four elements characterize the instance of EP addressed, while the

provided as a vector (x_c, y_c). Each cell contains a variable, whose value tells if that cell is clear, occupied by an obstacle or still unknown. Clearly, the first two values refer only to already explored cells, while the third refers to those cells which are still not scanned by any robot of the team.

The experiments presented in this work are run on MRESim, a 2D simulator aimed at testing multi-robot exploration scenarios [1]. In particular, it allows to create an instance of *EP* through a configuration file which provides the robots in the team, their initial poses and a PNG image providing the map of the environment to explore. The termination criterion used is the percentage of explored area and it can be modified by varying a variable in the code.

The following sections describe the settings for the experiments, focusing at first on the environment and the agents modeling, then moving the attention to the exploration strategy and the coordination mechanisms.

2.2 Localization and mapping

The problem of mapping an unknown environment and simultaneously localize the robot on the partial map built is one of the fundamental problems of robotics, called *SLAM problem,* from *Simultaneous Localization And Mapping.* The robot knows its initial pose and as it moves, the uncertainty about its motion increases, due to uncertainty in the odometry. For this reason it becomes necessary to localize it on the map, even a partial one. A formal description of it, as provided in [5], is suitably done in a probabilistic framework and distinguishes two versions, the *online* and the *full SLAM problem*. Let X_T, U_T and Z_T be three statistical variables representing respectively the sequence of locations assumed by the robot, i.e. the *path*, the sequence of odometry measurements and the sequence of measurements provided by the sensors. Let also m be the true map of the environment. The *full SLAM problem* is then defined as the problem of estimating the posterior probability of the map together with the whole path traveled by the robot, that is $p(X_T, m|Z_T, U_T)$. The *online SLAM problem* aims at estimating the posterior of the actual location of the robot, rather than the whole path, together with the map. Thus, $p(x_T, m|Z_T, U_T)$.

What happens in practice is that as the robot moves in the unknown environment, it perceives the surroundings through its sensors, has an estimate of its odometry measures and these are used to reduce the uncertainty

the environment and it is also possible to get rid of the noise in the measurements. The model for the map can be both topological [6] or metric [7], in particular in this work the representation provided is metric, consisting in an occupancy grid.

2.2.1 Environments

This work considers only indoor environments. Reasons behind this restriction are related to the work of [4], of which this thesis is an extension, in order to provide a comparison against the same type of environments. However, this focus is justified by the application contexts, being the exploration of indoor environments more common with respect to outdoor ones.

The representation of the map provided by the SLAM algorithm used in this work is a two-dimensional occupancy grid. Moreover, being applied to a multi-robot scenario, comes out the difficulty of merging the maps computed by solving the SLAM problem for each agent. The solution to this is a centralized approach by means of a base station, which aim is to collect all the individual local maps and combine them into a single global map.

Similarly to [4], the environment are classified according to some features characterizing them. In particular, the aspects considered are the dimension, the openness, and the parallelizability. The first two are easy to formally define. For the third one, a formal definition can hardly be provided being influenced by many factors. To make the definitions more readable, recalling that M denotes the map of the environment modeled as an occupancy grid, let $S : M \to \mathbb{R}$ be an auxiliary function which takes as input a cell of the map and provides its area. The features considered can be defined in the following way.

- Dimension: the amount of free area in the map. Thus, $D(M) = \sum_i S(f_i)$ with f_i being a clear cell. The size of the environment clearly affects the average time taken by robots to explore it. Smaller the environment, lower the amount of resources needed to efficiently map it. From this, the distinction into small (S) and large (L) environments.

- Openness: the property of an environment to be composed of large open spaces (O) rather than cluttered ones (C). It is related to obstacle

[S-O-LP environment]Catteneo/MRESim/environments/Tesi/env$_3$"*.eps*"

[L-C-HP environment]Catteneo/MRESim/environments/Tesi/env$_5$"*.eps*"

Figure 2.1: Example of two different environments

- Parallelizability: the property of an environment to enforce the spreading of the robots during the exploration. A formal definition to this is hard to provide, but it can be informally presented considering that if the environment allows the team to spread, it is highly parallelizable (HP). Otherwise, if the robots are forced to stick together, it is lightly parallelizable (LP).

To make these distinctions clearer, it is worth looking at some of the environments used in this work and at how they are classified on these features. In Figure 1, two very different environments are presented.

The environment in Figure 1.a is composed by a series of large corridors,

the robots follow almost similar paths, forced by the long corridors and they are not allowed to spread into it.

The second one (Figure 1.b) is likely to be the map of an office. It is composed by a lot of rooms, corridors and spaces with different dimensions. It is classified as L-C-HP because the amount of free area is really large and is mainly composed of rooms and limited spaces. For the configuration of the corridors, the robots tend to spread in different directions, without sticking one another. For this reason, the environment is classified as highly parallelizable.

2.2.2 Agents

In MRESim, an agent is characterized through:

- a number and an ID to uniquely identify it;
- its pose, expressed by a vector $p = [x, y, \phi]$, where (x, y) are the coordinates of the grid cells occupied by the robot and ϕ is its orientation;
- the sensing range, set to the default value;
- the communication range, assumed to be infinite;
- the battery life, assumed to be infinite as well;
- its type, it can be the base station, a relay or an explorer.

The communication range and the battery life are assumed to be infinite because the focus of this work is on the results provided by the use of different coordination mechanisms. In this way is possible to simplify the mechanisms not to consider scenarios in which the robot runs out of fuel or they are unable to communicate, even if these situations are of particular interest in a more realistic context and object of study as in [2] and [3], respectively.

As stated above, the type of the robot, not to be confused with its role, can be: base station, relay or explorer. The *relay* type is never used in the simulation performed for this work, being useful in cases in which communication between a robot and the base station is not possible. Indeed, each team used is composed by one base station and from two to eleven explorers.

one. It is important for the purpose of exploration because allows agents to coordinate by means of a centralized unit, rather than a decentralized approach, which would make the merging of the map way more difficult. An example of this is in [8], where only as soon as two robots meet, they are able to merge their map and then proceed with the exploration based on this updated one.

The *explorer* is the main kind of agents employed, being the one which moves in the environment to map it. Apart from the configuration parameters presented above, it is also characterized by a finite value of speed. They are equipped with a laser sensor allowing them to scan a semicircle of radius fixed to the sensing range in front of them. The value for the sensing range is specified in the configuration file. As an explorer perceives previously unknown cells with the laser beam, communicates the measurements to the base station which merges them with the stored map and updates it.

2.3 Exploration strategy

The frontier-based exploration presented in [9] is applied to identify the candidate locations to explore. A frontier is the boundary region between known and unknown space and by moving towards frontiers, this boundary is pushed accordingly.

As highlighted in the previous section, the model for the map is an occupancy grid. Each robot has its own local map, updated through the sensor measurements, and merged at the base station to provide a global map, which is the one used in the frontiers identification phase.

Their position is identified by at first drawing the polygon of the known space. Each side of this polygon which separates the known area from the unknown one is suitable to provide frontiers. If the area of this suitable locations is higher than a certain threshold, then the cell in the middle is considered as a possible navigation goal. There is also the possibility that the area is too large, in which case it is split into more than one frontier.

2.4 Coordination mechanisms

A coordination mechanism is the algorithm providing the allocation of robots to the possible navigation goals computed by the exploration strategy. To-

the sensors and these data, together with the initial location, are used as input to the SLAM algorithm, which provides a partial raw map as output. This is given to the exploration strategy that, as stated above, performs a discretization of the polygon surrounding the known area and computes the frontiers, excluding the ones too small. At this point, the coordination mechanism is applied to find out an allocation of robots to frontiers.

The different coordination mechanisms are the main focus of this work. In the following are presented both the base mechanisms from [10] and their extensions provided by [4].

2.4.1 Base mechanisms

In [10], three mechanisms are presented, namely *reserve, buddy system* and *divide and conquer.* They are all focused on how *proactive* are the team members not strictly needed for the exploration. The team can be separated in two sub-teams, the *active* and the *idle set*, with the first one composed by the robots which goal is to explore one of the frontiers detected, the second one is made up by the remaining robots. An example is useful to clarify this distinction.

Suppose that at the beginning of the exploration, the sensed environment provides only two frontiers (may add a figure) and the team is composed by four robots. The size of the team is greater than the number of frontiers, thus only two robots are needed to explore them. Therefore, they compose the active set and the remaining two robots form the idle set. The policy for the active set is the same for all the mechanisms, to explore the closest frontier. The differences come out dealing with the idle set, in fact the way in which robots are proactively moved allows to distinguish among each method. Moreover, this theme of a proactive use of the idle set is the leitmotiv linking the work started in [10] and extended both by [4] and this thesis.

Reserve is the less proactive mechanism because as the name implies, the idle set is left as a reserve at the initial location. As the exploration starts and the first frontiers are detected, the robots are split into the active set and the idle set, in a similar way to the example provided before. Then, the robots in the active set move towards their navigation goal and the ones in the idle set remain still in their initial position. As the exploration goes on and new frontiers are found, their number may be higher than the size

in an uncoordinated way.

Divide and conquer is the most proactive mechanism presented in that work because the idle set moves together with the active set. At the beginning of the exploration, active agents are assigned to the frontiers and the idle set is split in several subsets, one for each active agent. For example, assume that at the beginning, only a frontier is found. Then, an agent is marked as *leader* and assigned to explore it. The other robots follow it as it moves towards its navigation goal, until at least another frontier is detected. For the sake of the example, assume that at this point there are two frontiers. In this case, a robot from the idle set is turned into active and marked as leader. The idle set is split in two: one half follows the first leader, the other half follows the other one. This goes on until the idle set is empty, after which the exploration proceeds in an uncoordinated way. Differently from the reserve mechanism, this approach allows to have robots from the idle set nearer to the navigation goal to which they are going to be assigned. In this sense, the idle team members are considered to be more proactive with respect to reserve, where waiting at the initial location makes the distance between the robot turned into active and the navigation goal higher.

Buddy system is considered to be halfway between the two for what concerns proactivity of the agents. This comes clearer by looking at how the system handles the idle set. As soon as robots are deployed on the environment, couples are formed, composed by a robot marked as *leader* and another marked as *follower*, each one is the *buddy* of the other. Once frontiers are identified, the minimum number of leader agents is turned into active and assigned to them. Each follower follows its own leader towards this task. The other couples remain still at the initial location, similarly to the idle set in the reserve mechanism. When a branching point is met, that is a point where there are two or more frontiers, the couple is split and the leader is assigned to a frontier, while the follower to the other one. Here, the analogy with the divide and conquer mechanism can be seen. If a robot split from its buddy finds another branching point, a couple from the idle set is called and assigned to one of the two frontiers, while the single robot goes towards the other. This clarifies why the buddy system can be seen as a mix of the reserve mechanism and the divide and conquer. It both keeps the idle set waiting at the initial location for the moment in which it is needed, as reserve, and each leader goes with a follower from which splits when a branching point is met, similarly to the divide and conquer.

2.4.2 Proactive mechanisms

In [4], three modifications for the mechanisms presented in [10] are provided. As stated previously, they focus on enhancing how proactive is the idle set. The way in which this is done is pretty straightforward for what concerns reserve and buddy system, while it is a little more tricky for divide and conquer. The mechanisms proposed are named *proactive reserve, proactive buddy system* and *side follower.*

The main problem with reserve mechanism is that once a robot from the idle set is turned into active, it has to move from the initial location to its goal. The distance it has to travel may be lower if the robot is moved in a nearer position while it is still idle. This is exactly what this modified mechanism tries to do by moving the idle robots to the barycenter of the active robots positions. In this way, they are likely to be nearer to the newly detected frontiers which have to be explored.

The buddy system faces a similar problem to reserve mechanism: the robots in the idle set wait to be turned into active at the initial location. Thus, the proactive version of the buddy system moves the idle couples at the barycenter of the active agents locations, for the same reason explained above concerning the proactive reserve.

The problem of divide and conquer is different, being linked with the interference caused by moving the whole team of robots together. Therefore, the team of robots is split into groups of three at the beginning of the exploration and roles are assigned to them. The central robot is the leader, the right one is the right follower and the left one is the left follower. This roles are statically determined and never modified. Moreover, they affect the assignment of frontiers to the group members, being the frontier along the direction of the movement assigned to the leader, the ones on its right assigned to the right follower and the ones on the left to the left follower symmetrically. In this way, there is a trade-off between the interference caused by the number of robots moving together and the distance from the frontiers, reducing the first one without affecting too much the second one.

Bibliography

[1] J. de Hoog, A. Visser, S. Cameron. MRESim, a Multi-robot Exploration Simulator for the Rescue Simulation League. 2015.

[2] M. Betke, B. Awerbuch and R. Rivest. Piecemeal graph exploration by a mobile robot. In Proceedings of the 8th Annual ACM Conference on Computational Learning Theory (COLT), pages 374–386, 1995.

[3] M. Moors, W. Burgard and M. Schneider. Coordinated multi-robot exploration. IEEE Transactions on Robotics, 21(3):376–378, 2005.

[4] M. Cattaneo. An Analysis of Coordination Mechanisms for Multi-Robot Exploration of Indoor Environments. 2017.

[5] S. Thrun, J.J. Leonard. Handbook of Robotics, chapter 37. Springer, 2008.

[6] G. Grisetti, R. Kümmerle, C. Stachniss, W. Burgard. A Tutorial on Graph-Based SLAM. In IEEE Intelligent Transportation Systems Magazine 2(4):31-43, 2010.

[7] D. Koller, M. Montemerlo, S. Thrun. A Factored Solution to the Simultaneous Localization and Mapping Problem. In Proceedings of the National Conference on Artificial Intelligence (Edmonton 2002), volume 37, pages 81–91, 2002.

[8] A. Giannitrapani D. Benedettelli, A. Garulli. Cooperative SLAM using M-Space Representation of Linear Features. Robotics and Autonomous Systems, 60:1267–1278, 2012.

[10] C. Nieto-Granda H. Christensesn and J.G. Rogers III. Coordination strate- gies for multi-robot exploration and mapping. The International Journal of Robotics Research, 33(4):519–533, 2014.

[11] F. Amigoni. Experimental Evaluation of Some Exploration Strategies for Mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 2818–2823, 2008.

Chapter 3

Modeling and evaluation

In the previous chapter, the coordination mechanisms developed in [**?**] have been presented, giving particular attention to the contribution provided by proactivity. Concerning buddy system and reserve, their proactive versions move the idle set at the barycenter of the positions of the active agents. This ensures good results when compared with the base method, in particular for proactive reserve. However, the location for the idle set is computed naively and might be exploiting only a part of the information available at the moment related to the structure of the environment. For example, the barycenter of the positions of the active robots may fall into a small room, from which a robot in the idle set would have to get off once turned into active, making the move into the room almost useless. On the contrary, we are more interested in moving the robots in the idle set towards positions providing a good starting point for them.

To include the structure of the environment directly into the coordination mechanism, graphs are considered. They provide a representation of the free space and computing some suitable metrics, called centrality metrics, it is possible to find out a subset of nodes more influential on connectivity between spaces. In this way, the proactivity location, i.e., the location where the idle set is proactively moved, is a central point of the environment. The concept of centrality varies accordingly to the centrality metric used to compute such subset of nodes. In the following, the graphs and the centrality

3.1 Graphs

The use of graphs allows including a topological component in the coordination, which is done employing an embedded graph. A graph G embedded to a surface Σ is a representation of G on Σ such that points of Σ and arcs in it are associated with vertices and edges of G respectively. This concept is applied by creating a graph Υ embedded on the map M and then through the measures of the centrality of a node, nodes are ranked based on their values for the two centrality measures used in this work.

The definition of an embedded graph is independent of the actual implementation of it and this leaves the freedom to adapt the correspondence points-nodes and arcs-edges on the needs. Based on this, two different types of graphs are defined in the next sections.

These graphs differ strongly on how they are defined and how they model the environment. The first one is more focused on the topological properties of the environment, nodes and edges are computed based on the structure of the known free space. The model it provides is not strictly affected by team-dependent factors like agents distribution or team size, being built only considering the connectivity properties of the map. On the contrary, the second kind of graph defined below is built according to the positions of the agents moving in the environment. In this way, it includes information related to the disposition of the agents and the path followed. Moreover, it is also indirectly affected by the structure of the environment, being the agents only capable of moving in the free space, the routes they travel outline the connectivity among the different spaces. In the following, how frontiers are introduced into the graph building process is also explained.

The following sections explain in detail how the two types of graphs are built during the exploration process, starting with the topological graph and then moving on to the visibility one.

3.1.1 Topological graph

From now on, as *topological graph* will be identified a graph isomorphic to the graph used by the simulator to compute the path followed by the agents during the exploration. As presented in [1], the construction of this one is based on the structure of the occupancy grid known up to that moment, in which the skeleton of the free space is found by performing thinning.

process is sufficient to provide a simple topological graph embedded on the map of the environment.

On one side, the use of this graph is backed up by the implementation of the navigation system, because it is computed just once for both the applications and in this way it does not introduce further costs in the building and the update. On the other side, it is pretty limited in its representation. It suffices to provide a topological view of the environment, but it is hard to extend with elements like frontier nodes or information about the location of the robots. Two aspects that characterize the second kind of graph tested, the visibility graph.

In Figure 1.a, the topological graph of the environment presented in the previous chapter is shown. This one has been produced by applying the building procedure on the whole map of the environment when it has been completely mapped. This has been considered to show more clearly the properties of this graph. The disposition of the nodes is almost uniform and this can be particularly noticed by looking at their distribution in the lateral rooms. Being these spaces almost equal in structure, nodes are placed with the same pattern. Moreover, the distance among nodes tends to be almost the same on the entire map, with a slight reduction at the conjunctions. This almost uniform distribution highlights strongly the difference in structure with the visibility graph, shown in Figure 1.b, where nodes are provided with varying concentrations among the various spaces.

3.1.2 Visibility graph

The *visibility graph* as defined in this work consists of a graph built on the notion of visibility, rather than the navigability from node to node. It is composed of two different types of nodes, the *pose nodes,* and the *frontier nodes*.

The first kind is the one composing the vast majority of the graph and each node has an historical meaning, being a pose assumed by an active robot during the exploration. As active agents proceed in their mapping task, their locations are stored as nodes every time a re-plan for one of them or a new location for the robots in the idle set is needed. These two events happen quite frequently in the initial part of the exploration, and thus they trigger the graph building function enough times. However, the frequency with which this is done is variable, for this reason, the distribution of the

node. Nevertheless, as long as connectivity is ensured, the effect on the information obtainable by the graph is almost the same and by keeping the size of the graph reduced, the complexity of computing centrality measures remains manageable. To enforce this aspect, the location of an active agent is added as a pose node as long as there are no other pose nodes within a certain radius. Once a pose node is added to the graph, it is fixed and never modified as the exploration goes on. The motion of agents in the idle set is excluded to avoid an excessive concentration of nodes in some crucial areas, which would negatively affect the computation of centrality measures.

The second kind of node is frontier nodes. As the name suggests, they allow to include the frontiers computed by the exploration strategy into the graph. This is a fundamental characterization of this graph in the distinction from the topological one. At each step in which the list of frontiers is updated, the same is done for the list of frontier nodes: the old ones which have been explored are removed and the new ones are added, making this type of nodes variable in time differently from pose nodes which are fixed once they are put in the graph.

An edge models the notion of visibility: two nodes n_1 and n_2 are linked by an edge if and only if $n_1$is within the sensing range of a robot placed in n_2 and there are no obstacles along the straight line connecting them. This relation holds in both the directions, thus the visibility graph is an undirected graph by construction, same as the topological one. Edges are also characterized by a weight equal to the distance between the nodes.

Figure 1.b shows the visibility graph built by a team of robots during the exploration. Accordingly to what stated previously, the distribution of nodes depends strongly on the path followed by the robots and on the frequency with which the graph is updated. The structural differences with the topological graph in Figure 1.a are clean just by looking at the two figures and all relate to the distribution of nodes, being widely less uniform. Frontier nodes, here painted in red, are also a discriminating factor between the two.

3.2 Centrality measures

Centrality measures have been exploited widely in the literature, especially in fields related to social networks [**?**], power grids [**?**], disease [**?**] and computer virus spreading [**?**]. This is possible since centrality measures are

[Topological graph. Green dots represent the nodes]Catteneo/MRESim/logs/disegni/d3t".eps"

[Visibility graph. Green dots represent pose nodes, red ones are the frontier nodes]Catteneo/MRESim/logs/disegni/d3v".eps"

Figure 3.1: Examples of graphs at the end of the exploration of the environment presented in the previous chapter. Edges are omitted for clarity

Figure 3.2: Kite graph. The pink node has the highest degree, the orange node has the highest betweenness and the blue nodes have the highest closeness. Green nodes are the remaining nodes.

topology, because of the mismatching concept of central node. An example of this is provided by the kite graph [**?**]. It is a simple graph composed of 10 nodes and 18 edges and it is depicted in Figure 2. The particularity of it is to be the smallest possible graph for which the nodes having the highest values of the three most basic centrality measures, namely degree, closeness, and betweenness, are all different.

To the author's knowledge, there are no previous works trying to apply the use of centrality measures to enhance the coordination of a team of robots and due to the variability depending on the centrality measure used, in this work both closeness and betweenness have been tested. In the following sections, they are formally introduced with the reasons for their use.

3.2.1 Closeness

The closeness centrality of a node is defined as the reciprocal of the sum of the length of the shortest paths between the node and all the other nodes in the graph [**?**]. This definition is strongly dependent on the number of nodes N in the graph, thus closeness is usually normalized by dividing for $N-1$. In this way, closeness can be defined as the reciprocal of the average distance between the node and all the other nodes and allows to compare its value for graphs of different sizes.

Formally, let x and y be nodes of the graph, and let d be a real-valued function which provides the length of the shortest path connecting two nodes, then the normalized closeness value $C(x)$ is defined as

$$C(x) = \frac{N-1}{\sum_{y \neq x} d(x,y)}$$

with N the total number of nodes in the graph, as defined above.

The original formulation of closeness centrality considers only unweighted graphs, but it has been extended to be applied also to weighted ones. The formal definition is the same assuming an appropriate modification in the implementation of the distance function d. Indeed, in an unweighted graph, it simply has to count the number of edges along the shortest path linking

[Topological graph. Green dots are the nodes with a value of closeness lower than the 75% of the max value.]Catteneo/MRESim/logs/disegni/d3t c".eps"
[Visibility graph. Green dots are the remaining pose nodes and red ones are the frontier nodes.]Catteneo/MRESim/logs/disegni/d3v c".eps"

Figure 3.3: Distribution of nodes with a closeness higher than the 75% of the max value for the topological and the visibility graphs. Blue dots are the highest closeness nodes, cyan dots are the ones with a high value of closeness but not the maximum, and edges are omitted for clarity.

x and y are nodes of the graph, E is the set of edges of the graph composing the shortest path from x to y, and w is a real-valued function returning the weight of the edge.

Closeness tends to consider central nodes the ones in which distance from all the other nodes is lower on average. Therefore, going back to an exploration context, placing an agent in the location corresponding to the highest closeness node makes it possible to reach an assigned position, not known before, in an expected time lower than any other starting location with a lower closeness. This reasoning has been applied to the idle set, which placed in the node with the highest closeness is likely to already be in a good spot when turned into active.

In Figure 3 an example of the distribution of high closeness nodes is shown for each kind of graph. For the sake of the example, as high closeness nodes are considered nodes with a value of closeness higher than the 75% of the max value. It is interesting to point out how these nodes are distributed mostly on one corner of the central space for the topological graph, while they wrap the central obstacle in the visibility case. Despite the marked differences in their structures and the number and disposition of high closeness nodes, it is remarkable that the distance among the highest closeness nodes is really low.

3.2.2 Betweenness

Betweenness centrality for a certain node of the graph measures how much of the total number of shortest paths between other nodes passes through

the node v is defined as

$$B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where the sum is performed over each pair of nodes in the graph. The graph needs to be connected, otherwise, each σ_{st} where s or t is a disconnected node would result in a division by zero. However, this is always granted in the context of this work because of the way in which graphs are built.

Betweenness followed an evolution similar to the one of closeness, being at first defined on unweighted graphs [**?**] and then extended to the case of weighted ones [**?**]. In the case of weighted graphs, weights impact how shortest paths are computed, making necessary the use of algorithms like Dijkstra's or Breadth-First Search to deal with them. Once the shortest paths are provided, the algorithm to compute the betweenness is the same. Similarly to closeness, where the introduction of weights on the edges only affects the computation of the distances between nodes.

The idea behind this metric is to assign higher importance to the nodes which are along more shortest paths linking couples of other nodes. In different works about social networks analysis [**?**], betweenness is used to find out which are the nodes having more control over the information flow. Nodes with a high value of betweenness are along more shortest paths, thus more information goes through them and an eventual disconnection may cause loss of information or a separation of the graph into two sub-graphs. However, a node of this kind is likely to be fundamental for what concerns the connectivity of the graph, differently from a node with low betweenness. According to this, the approach based on betweenness has been conceived. Being interested in an effective positioning of the idle set, a location with a high value of betweenness is an ideal candidate because it guarantees to be a crucial point for the connectivity of the environment and when the idle set is turned into active, it is likely to be in a good position to navigate towards the assigned frontier.

In Figure 4, the nodes with high betweenness are plotted over the representation of the graphs provided before. The first thing that catches the eye is the difference in the number of this kind of nodes between the topological and the visibility graphs, being a lot more in the first one. This can be justified by considering the reduced size of the graph in that case,

[Topological graph. Green dots are the nodes with a value of betweenness lower than the 50% of the max

value.]Catteneo/MRESim/logs/disegni/d3t b".eps"
[Visibility graph. Green dots are the remaining pose nodes and red ones are

the frontier nodes.]Catteneo/MRESim/logs/disegni/d3v b".eps"

Figure 3.4: Graphs with nodes having a high value of betweenness highlighted. Orange nodes are the ones with the highest betweenness, while in yellow nodes with a betweenness comprised between the 50% and the max value are drawn. Edges are omitted for clarity.

and the even more marked difference in the distribution of high betweenness nodes, the highest values of the metric are measured in two extremely near locations. This enforces the idea that these metrics can characterize the environment mapped and by an analysis of them, it might be even possible to discriminate among different types of it.

3.3 Comparative metrics

The different methods proposed are compared based both on practical measures, as the time taken to fulfill the termination criterion and the distance traveled by the robots, and on theory-based measures, namely interference among robots and their availability.

The use of time and distance traveled as comparison metrics is intuitive if looking at some of the application contexts. Teams of robots are often used in search and rescue scenarios, where the time needed to complete the exploration is a fundamental aspect to take into account. Thus, a faster approach is overall preferable to a slower one in such a scenario. The distance traveled is a less important factor in discriminating among different mechanisms, but it may provide an interesting and more complete overview of an approach compared to others.

In the simulations run in this work, the termination criterion used is the exploration of the 95% of the environment. At the end of each run, the number of discrete time steps taken and the average distance traveled by the robots are stored and then compared during the analysis of the results.

more computationally intensive than the ones based on the barycenter computation, and comparing their results on the effective time taken would have been faked by the computing capability of the machine running simulations.

These two metrics have been considered sufficient to characterize from a practical point of view each mechanism analyzed. Nevertheless, two further measures are used to examine the mechanisms, which are named interference and availability. They are at first introduced in [**?**] and formalized in [**?**].

3.3.1 Interference

Interference quantifies the average distance held by agents during the exploration and the higher the distance, the lower the value. A high value of interference is desirable because as the average distance among robots increases, it reduces the possibility of incidents and the complexity in the management of the system. Moreover, it is also an indirect measure of how parallel the exploration is being carried on because it increases as the robots are spread on the environment. According to this, the value of interference for a particular coordination mechanism can provide useful information about the amount of parallelizability exploited as compared to other mechanisms.

The value of interference λ for an agent a is computed at each step t of the exploration by calculating the average distance between a and all the other agents a' in the team A, thus it can be formally written as

$$\lambda_t(a) = \frac{1}{N-1} \sum_{a' \in A | a' \neq a} d\left(P_t(a), P_t\left(a'\right)\right)$$

where N is the size of the team, P_t is a function providing the position of an agent at time t, while d is the function that computes the distance between two positions of the environment.

This definition only relates to one agent at a particular instant of the exploration. To completely characterize the value of interference for the whole exploration, it needs to be at first generalized over the set of agents composing the team and then over the entire time needed to complete the exploration. In this way, its value computed for a specific coordination mechanism is comparable with the ones provided by other mechanisms over the same exploration problem.

The first generalization consists in extending the definition of interference to all the robots in the team, rather than considering only a single robot, and this is simply done by averaging the values of interference of each agent:

At this point, it is possible to integrate this expression over the entire time taken by the exploration, which is considered to take values in $[0, \dots, t_T]$ with t_T being the time step of the fulfillment of the termination criterion T. The definition of the interference for the coordination mechanism applied to a specific instance of the exploration problem considered is

$$\lambda = \frac{1}{t_T + 1} \sum_{t=0}^{t_T} \lambda_t$$

This states that the interference for the whole exploration can be computed by averaging over the total time taken the values of interference for the team. In this way, the value of interference obtained can be exploited to compare different coordination mechanisms, avoiding that differences in the duration of the exploration impact on this measure.

3.3.2 Availability

Availability is a measure of the distance between an agent and its assigned location. It can be formally defined at first for a single agent a in a certain time step t as

$$\alpha_t(a) = d(P_t(a), G_t(a))$$

where α is the symbol used for the availability and G_t is a function returning the location of the frontier assigned to the agent in input. The two auxiliary functions d and P_t are the same presented in the previous section to define the interference.

Similarly to what done for the interference, this definition can be generalized to the whole team and the whole exploration. Following similar reasoning and with the same meaning of the symbols, the availability for the whole team at a certain instant t of the exploration is

$$\alpha_t = \frac{1}{N} \sum_{a \in A} \alpha_t(a)$$

recalling that A is the set of agents composing the team and N is its cardinality. Averaging this result over the whole exploration time t_T defines availability for a coordination mechanism applied to a specific instance of the exploration problem, thus it turns out to be

$$\alpha = \frac{1}{t_T + 1} \sum_{t}^{t_T} \alpha_t$$

coordination mechanisms, without being affected by their respective performance in terms of time. It is important to highlight how availability has the opposite trend of interference. Indeed, a mechanism with low availability assigns robots to locations near their positions, which makes the agents update the environment more frequently than a scenario in which robots have to travel a long distance before scanning unknown portions of it.

This metric has a two-fold interpretation. From one side, it shows whether a mechanism assigns robots to far or close targets. On the other side, it can provide insights on the effectiveness of the proactivity when comparing two mechanisms which differ only in this aspect. This, in particular, is the setting of this thesis. The differences of the mechanisms analyzed concerns only the proactive allocation of the idle set, and thus a mechanism that has a lower value of availability is likely to place this set of robots in a position nearer to the frontiers. Strong evidence of this relation between proactivity and availability is provided by [**?**] in the comparison between reserve and proactive reserve, where by moving the idle set towards the barycenter of locations of the active agents, the latter method ensures agents to travel a shorter distance once turned into active with respect to the former one.

It is worth to point out also that the absolute value of both this metric and the interference is highly affected by the particular configuration of the environment explored and by the team size, for this reason, comparisons among mechanisms based on them do make sense only if done on the same instance of exploration problem.

Bibliography

[1] J. de Hoog, A. Visser, S. Cameron. MRESim, a Multi-robot Exploration Simulator for the Rescue Simulation League. 2015.

[2] D. Krackhardt. Assessing the Political Landscape: Structure, Cognition, and Power in Organizations. Administrative Science Quarterly, 35(2), pages 342-369, 1990.

[3] L. C. Freeman. Centrality in Social Networks Conceptual Clarification. Social Networks, 1, pages 215-239, 1978.

[4] M. E. J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. Physical Review E, 64, pages 016132, 2001.

[5] M. Cattaneo. An Analysis of Coordination Mechanisms for Multi-Robot Exploration of Indoor Environments. 2017. Master Thesis, Politecnico di Milano, Italy.

[6] C. Nieto-Granda H. Christensesn and J.G. Rogers III. Coordination strategies for multi-robot exploration and mapping. The International Journal of Robotics Research, 33(4):519–533, 2014.

[7] A. B. M. Nasiruzzaman, H. R. Pota, M. A. Mahmud. Application of Centrality Measures of Complex Network Framework in Power Grid.

[8] A. Dekker. Network Centrality and Super-Spreaders in Infectious Disease Epidemiology. In Proceedings of the 20th International Congress on Modelling and Simulation. 2013.

[9] Y. Wang, H. Liu, B. Ren, J. Chen. Node centrality analysis of multiplex networks under Computer virus spreading. In Proceedings of the 3rd

Chapter 4

Coordination mechanisms

This thesis is aimed at exploiting measures related to the environment into a proactive allocation of idle agents. This tries to be a further improvement of the mechanisms proposed in [Christensen], with respect to [Cattaneo]. The proposed mechanisms are analyzed in detail in this chapter, giving particular attention to how the various elements presented before, namely graphs and centrality measures, are employed.

At first, a common structure for the coordination mechanisms is defined, and it is shown how a different implementation of the different functions allows characterizing each one. This is done starting from reserve and buddy system, as presented in [Christensen], being the building blocks of the other mechanisms proposed. Then, the focus shifts to the proactive versions developed by [Cattaneo].

After this, it is presented how the proposed proactive mechanisms modify some components of this common structure to include the use of topological aspects.

4.1 Base mechanisms

Reserve and buddy system have been introduced intuitively in the previous chapters, and here a more detailed look at how they are implemented is given. Recalling that the buddy system is conceived as a mix from reserve and divide and conquer mechanisms, it is clear why they share some algorithms.

The general structure for the coordination mechanisms analyzed can be

- activation function;
- goal function;
- proactivity function.

For some aspects, the planning function works as a common interface provided by the mechanism. As the simulation requires an agent to move, the step to take is returned by this function. It calls the appropriate function coherently with the state of the agent, whether it is active or idle. It also initializes the exploration by setting the starting agent and forming the active and the idle sets.

The remaining three functions allow characterizing a coordination mechanism. They enclose the logic about the assignment of frontiers to active agents, what triggers idle agents to be turned into active and where they are kept until that moment.

In particular, the proactivity function is the focus of this work. In the previous chapter, the importance of graphs and centrality measures has been introduced. How these are included in the proposed coordination mechanisms is shown in the last section, giving also attention to some optimization performed to avoid an excessive impact of their computation.

4.1.1 Reserve

Reserve mechanism divides the team of robots into two sub-teams, one composed of active agents and the other composed of idle ones. This separation is done by the planning function as soon as the exploration begins, once the first set of frontiers is computed.

At first, the distance of each agent from each frontier is retrieved, then the agent of the pair with the lowest distance is set as the starting agent and assigned to that frontier. This assignment is iterated until every frontier is assigned to an agent or vice versa. The set of assigned agents composes the active set, while the remaining agents if any, form the idle set. After this initialization step, the planning function is invoked every time an agent either has reached its goal or a certain amount of time has passed since the last planning. In both cases, it is in charge of calling the right function among the activation function, the goal function or the proactivity one.

Once an agent is active, which frontier is assigned to it is computed by

The activation function can be invoked only for agents of the idle set, being the function aimed at turning them into active. This function checks whether there are any not assigned frontiers and if so, the agent is turned into active and assigned to the closest one. If this is not the case and all the frontiers are assigned, the proactivity function is called.

The reserve policy is that the idle set waits at the initial location and thus, the proactivity function always returns the position of the agent. In this way, agents of the idle set stay still, until the activation function assigns them to a frontier.

4.1.2 Buddy system

Buddy system proceeds in a way similar to the reserve one. This method is characterized by the use of paired robots and each robot in the pair is the buddy of the other, from this the name of the mechanism. Pairs are created before the exploration begins, by assigning a role to each robot of the team. The roles are two, either an agent is a leader or a follower and they are assigned to split the team into two halves. In the case of odd teams, the extra agent is assigned leader role.

Apart from this division into leaders and followers, the team is also split into an active set and an idle set. The active set is composed of leaders assigned to frontiers and their buddies. The idle set is composed of the pairs waiting at their initial locations.

The first call to the planning function selects the starting pair. This is done similarly to the reserve, by looking for the robot closest to a frontier. If it is a leader, it is assigned to that frontier. If it is a follower, its buddy is retrieved and being it the leader, it is assigned to that frontier. As for reserve, this approach is iterated as long as there are no more leaders or frontiers to assign. All the leaders assigned to a frontier and their buddies compose the active set. The idle set is composed of the pairs whose leader is not assigned to a frontier if any.

Further calls to the planning function distinguish whether the calling agent is a leader or a follower. In the former case, it simply invokes the goal function, while in the latter it checks whether the pair needs to be split. This happens when the two closest frontiers are enough distant one from the other. This is a branching point and the pair is split. The leader is assigned to the closest. The follower is now considered a leader itself and

a leader or a follower.

The goal function for leaders is the same as reserve. It assigns the agent to the closest frontier, while the goal function for the follower moves the agent towards the frontier assigned to its leader. This makes the pair go together in the direction of the assigned frontier.

The activation function for the buddy system follows the same principle of the reserve one. If there are any not assigned frontiers, the closest idle leader to each one is computed and turned into active by assigning that frontier to it. This has the side effect of turning into active also its buddy because the goal function will now make the follower follow its leader, rather than waiting at the initial position.

Buddy system keeps the idle set waiting at the initial position. Therefore, when called by an agent, the proactivity function simply returns the current location of the agent.

4.1.3 Benchmark proactive mechanisms

The proactivity function in the previous cases consists only of returning the position of the agent calling it. This makes the agents stay still at their initial locations until the activation function turns them into active. As discussed in the previous chapters, this makes the average distance between the agents and the assigned frontier higher, penalizing performance. To overcome this, a new proactivity function has been proposed in [Cattaneo]. Robots of the idle set are moved to a position likely to be nearer the future assigned frontier and this speeds up exploration. Experimental results show this modification to actually be an enhancement in the case of reserve. The proactive version outperforms the base version on almost every environment on which it has been tested. The proactive buddy system, on the contrary, is affected less by the proactivity.

The previously defined functions are the same for both the algorithms. The only difference is in the proactivity function. For both proactive reserve and proactive buddy system, the proactivity function assigns to the agent a goal location computed as the barycenter of the locations of the active agents. It can be formalized as

$$G_t(a) = \frac{1}{\mid A \mid} \sum_{a' \in A} P_t(a')$$

$G_t(a)$ is the goal location assigned to the idle agent. The subscript t for the goal function is possible since the location of the active agents at time t are used to compute the barycenter. There is no delay in the propagation of the information.

This proactivity function is characterized only by the actual location of active agents. The environment is not explicitly included in this formula. However, it affects indirectly where the robots are moved. To take its features into account directly, an approach based on graphs and centrality measures is proposed in the following section, characterized by a different implementation of the proactivity function.

4.2 Proposed proactive mechanisms

The proposed proactive mechanisms differ from the previously defined ones mostly for the proactivity function. There are no other conceptual differences in the other elements of the structure. Even if a small variation to the implementation of the goal function is introduced, as will be pointed out in the next section.

The proactivity function is modified in a way to include the computation of either the closeness or the betweenness and to use their values to obtain the proactivity point. Independently from the measure needed, the proactivity function follows the same steps. At first, it is checked if the graph has been modified since the last computation. If not, the proactivity point returned is the same as the previous step, avoiding useless computations. Otherwise, a new point has to be evaluated. This is done by determining the value of the considered metric for the nodes of the graph and then retrieving the most central one, i.e., the one with the highest value of the metric. If more than one node is found, the returned point is their barycenter. However, this case has a very low frequency and the number of nodes never exceeded two, during the experiments.

As presented in the previous chapter, the proposed proactive mechanisms exploit two different types of graphs, combined with two centrality measures. All their combinations have been tested both applied to the reserve and the buddy system, for a total of four different versions for each one.

In the following sections, it is presented how the mechanisms have been adapted to include the generation of the graphs and some optimizations aimed at allowing an efficient computation of the centrality measures. This

4.2.1 Graph building process

In the proposed mechanisms, there are two types of graphs tested. As already stated before, they are referred to as topological graph and visibility graph. Both have been implemented and tested to check how differently their use would affect performance. Their structures are built to enforce different aspects of the environment. The topological graph is built on the notion of navigability, that is it aims at capturing the connections among spaces. The visibility graph is based on the concept of visibility, which is more related to the possibility of sensing a certain location from another one.

Both graphs have been implemented employing adjacency lists. Recalling that the map is modeled as an occupancy grid, each node is characterized by its coordinates in it. It also holds a list containing all the nodes linked to it and the distance separating them. This has been preferred over a representation through adjacency matrices because both the graphs tend to be sparse, in particular the topological one. To confirm this, the average degree of the nodes of the topological graph is two, whereas it has a high variability for the visibility one. However, even in open environments, where the number of nodes visible from a node is higher, the average degree is always lower than half of the total number of nodes.

In the following, how these graphs are built is shown in detail, starting from the topological graph, then focusing on the visibility one.

Topological graph

The topological graph is a graph isomorphic to the one used by the navigation system to compute the paths followed by robots. In the following, this last one is referred to as navigation graph to avoid confusion. Whether it is a frontier for an active agent or a proactivity point for an idle one, every time the path from an agent position to its goal is needed, the navigation graph is checked. In case the time since the last update is higher than a threshold or the occupancy grid of the environment has changed, it is recomputed. This is done in the following way. The obstacle cells of the occupancy grid are progressively enlarged until the skeleton of the free space is found. It is then discretized into a set of nodes that are going to be the nodes of the navigation graph. The first kind of nodes correspond to branching points, then it looks for points filling the gaps among them. These points are added as nodes if the distance from the closest node is higher than a certain fixed

and this provides a partition of the map into different polygons, each one associated with a node. Two nodes are adjacent if the polygons associated with them share one side and this allows finding the edges of the graph, completing its construction.

Once the navigation graph is computed, the topological graph is built accordingly of the same set of nodes with the same adjacency relations. The only difference is that the edges of the navigation graph do not keep track of the distance between nodes, while the edges of the topological graph do. Indeed, the topological graph is a weighted undirected graph, where the weight function is a distance function. Given the structure of this graph, the difference between Euclidean distance and the effective length of the path between the two nodes is negligible, thus the first one has been applied for efficiency reasons.

The update of this graph is equivalent to the computation of a new navigation graph and is done every time a path for an agent is needed. Thus, referring to the general structure of a coordination mechanism presented before, it may be done by each of the four functions because each one implies the computation of a path for the agent. This ensures the topological graph to be updated frequently.

Visibility graph

As presented in Chapter 4, the visibility graph is composed of pose nodes and frontier nodes. Pose nodes map the positions assumed by the robots at various time instants. Frontier nodes are used to include the location of frontiers known at a certain moment into the graph.

It is initialized at the beginning of the exploration, considering as pose nodes the set of initial positions of the robots. The frontier nodes are included as soon as the first scan provides the first set of frontiers. After this, it is never rebuilt from scratch, like happens for the topological graph, rather it is progressively updated to match the movements of the robots and the disclosure of new frontiers. This holds because its construction is carried on entirely during the calls to the goal function. In both the reserve and the buddy system, this function comprises the update of the list of frontiers to properly assign robots. This provides the possibility to remove old frontier nodes and include the new ones. During this step, also the actual locations of active robots are used to generate nodes of the graph, the pose nodes, and

is way more restrictive than the navigability one, on which the topological graph is based. A simple example of this is the case of two nodes located on opposite sides of a wardrobe. It is clearly possible to define a path connecting them, but the sensors of a robot placed on one side can not provide measurements about what is on the other side. Moreover, edges built in this way are straight lines and as for the topological graph, the distance function providing the weight to them is the Euclidean distance.

4.2.2 Centrality measures

The main problem with closeness and betweenness is related to their computational complexity. They require the knowledge of the shortest paths connecting each pair of nodes and their length. This is particularly true for betweenness, while closeness only needs the last one. As the size of the graph grows, it may be difficult to compute the proactivity point in a reasonable amount of time. For this reason, two major optimizations have been performed. The first one is to compute the matrix of the distances between each pair of nodes in an efficient way and store it, allowing to avoid the repeated computations of the length of the shortest paths. The second one relates to how betweenness is computed.

Distance matrix

The distance matrix is a square matrix containing the distances between each pair of nodes of a graph. Both the topological and the visibility graphs are weighted undirected graphs, thus it is useless to have a square matrix, being it symmetric by construction. Moreover, the weights considered are always non-negative, for this reason, the distance of a node from itself is always zero. This allows reducing the size of the matrix from $N \times N$ to an $(N-1) \times (N-1)$ triangular matrix, where N is the number of nodes.

This data structure is essential in speeding up the computation of closeness. Recalling that the closeness of a node is defined as the inverse of the average distance of it from all the other nodes, it can be computed as

$$C(x) = \frac{1}{avg\left[row(x) \cup col(x)\right]}$$

where $row(x)$ and $col(x)$ provide respectively the elements of the row

sweep of the whole matrix, the value of the closeness of each node can be computed.

The main issue about the distance matrix is in its construction. It is an instance of the All Pairs Shortest Path problem, that is the problem of finding the shortest paths linking each pair of nodes. An algorithm able to solve this is the Floyd-Warshall algorithm. It is based on the concept of *intermediate nodes,* which is the set of nodes composing a path without the starting and the arrival nodes. Consider a graph G which nodes are $V = \{1, 2, \ldots, n\}$ and a subset $\{1, 2, \ldots, k\}$ of V for a generic k. Let also i and j be two nodes of G and consider all the paths connecting them, composed only of vertices in $\{1, 2, \ldots, k\}$. Let p be a shortest path connecting i and j. If k is an intermediate node of p, then the length of the path from i to k, plus the one from k to j is lower than the one from i to j having nodes in $\{1, 2, \ldots, k-1\}$ as intermediate nodes. On the contrary, if k is not an intermediate node, then this last path is shorter. By progressively increasing the value of k to consider the whole set of nodes as possible intermediate nodes, it is possible to find all the shortest paths for each pair of nodes of the graph.

The complexity of the algorithm is $\Theta\left(N^3\right)$. This is manageable in the case of the topological graph because N is limited by the graph building algorithm. On the other hand, the number of nodes in the visibility graph is much higher because no pruning is applied. The only device consists of forcing a minimum distance within pose nodes. However, as the exploration goes on, N might be sufficiently high to make the application of the Floyd-Warshall algorithm heavy. To overcome this issue, two factors allowed to come up with an efficient solution.

First of all the visibility graph is updated at every step and is never built from scratch, after the initialization. The second main aspect is that the Floyd-Warshall algorithm is an example of dynamic programming, thus the solution provided for the graph at time t can be used as a starting point to compute the solution for the graph at time $t' > t$. In fact, paths computed with a reduced set of intermediate nodes, suppose it to be $\{1, 2, \ldots, k-1\}$, are the shortest paths possible if the graph had node in $\{1, 2, \ldots, k-1\}$. Then, the extension of the intermediate nodes to $\{1, 2, \ldots, k\}$ provides the optimal solution for the graph with nodes in $\{1, 2, \ldots, k\}$. This idea can be applied by looking at the graph at time t like the one with intermediate nodes in $\{1, 2, \ldots, k-1\}$, while the one at time $t' > t$ as the one with

added to the graph. The complexity of each update is linear in the number of nodes, including the new ones.

Apart from computing all the distances between each pair of nodes, it is also possible to reconstruct the shortest paths linking them. Which modifications need to be done and how this is exploited are discussed in the following section.

Betweenness

Efficient computation of betweenness requires the preemptive computation of the shortest paths between each pair of agents. Assume to have these stored in a matrix, that, similarly to the distance matrix, is an $(N-1) \times (N-1)$ triangular matrix. Even if this is available, finding the betweenness $B(v)$ of a node v requires to go through each pair of nodes s and t, such that $s \neq v \neq t$, and retrieve the list of shortest paths linking them. This list needs to be iterated to count the number of shortest paths going through v and their total number for that pair of nodes. At this point, the ratio of these two quantities can be computed and the result has to be summed with the value for each other selection of s and t. Similarly to the application of the Floyd-Warshall algorithm, this algorithm may be a problem for the case of a visibility graph, where the number of nodes N might increase enough to make the computation unfeasible in a reasonable amount of time.

The main aspect of improvement is in the computation of the number of shortest paths from s to t going through v, which in the previous chapter was referred to as $\sigma_{st}(v)$. This is made possible by the *Bellman criterion,* which states that a node v of a graph lies on a shortest path between two nodes s and t, if and only if $d(s,t) = d(s,v) + d(v,t)$. Even if apparently obvious, this criterion allows to compute $\sigma_{st}(v)$ as

$$\begin{cases} 0 & if\ d(s,t) < d(s,v) + d(v,t) \\ \sigma_{sv} \cdot \sigma_{vt} & otherwise \end{cases}$$

where σ_{sv} is the value of the count of the shortest paths from node s to node v, and σ_{vt} is the analogous from v to t. The idea is that if v is along a shortest path from s to t, then $\sigma_{st}(v)$ can be computed as the number of shortest paths from node s to node v multiplied by the number of the ones from v to t, to include all the possible combinations.

On this, the betweenness can simply be computed in the same way as

to fill the matrix of counts by updating it as new shortest paths are found. This can be done while checking if the length of the path going through the considered intermediate node is lower than the previously known path. If the new path is lower, then the counter is reset to one. If the length is equal, then the counter is increased by one. If the new path is longer, then the counter remains equal because the new path is not a shortest path.

Once both these matrices are completed, computation of the betweenness for the whole set of nodes requires for each node to iterate over all the possible pairs of other nodes. Thus, the computation has cubic complexity in the number of nodes but it is independent of the length of the paths considered. On the contrary, this would affect the computation in the case the entire paths had to be scanned to check whether node v was present or not.

Reduced node-set

The optimizations provided before to speed up the computation of the measures, in particular for betweenness, work well in practice. This holds both for the topological graph and for the visibility graph, where the number of nodes is way higher, being even two to three times it. In some cases, this ratio can also go up to four times, producing an obliged impact on the computation. After all, the complexity holds a cubic relation with the number of nodes.

Moreover, another aspect taken into consideration is related exclusively to the visibility graph. The value of both the centrality measures for frontier nodes is almost uninformative. In fact, the value of betweenness is forced to be zero because no shortest paths between a pair of nodes go through a frontier. On the other hand, the value of closeness is always small as compared to the other value of closeness. This can be explained easily considering that frontier nodes are marginal nodes of the graph, because of their definition. Furthermore, being the sensing range way higher than the speed of the robot, the distance between pose nodes is lower than the distance between them and frontier nodes. Thus, the impact of frontier nodes on the value of closeness is negligible.

To deal with both these theoretical and empirical considerations, it has been decided to restrict the set of nodes for which centrality measures are computed. The reduced node-set R is then composed of pose nodes n such

nodes adjacent to n and indicating as F the set of frontier nodes. On this, an auxiliary set A can be defined as the set of nodes adjacent to a frontier node, namely

$$A = \bigcup_{f \in F} adj\,(f)$$

Then, the definition of R can be rewritten as

$$R = \left(A \cup \bigcup_{a \in A} adj\,(a) \right) \setminus F$$

R composed in this way allows a trade-off between considering the whole set of nodes and just the ones adjacent to a frontier node.

Thanks to all the tricks exposed so far, it has been possible to reduce the average simulation cycle time of the proposed mechanisms to values comparable to the ones of the mechanisms exploiting the barycenter.

Chapter 5

Realizzazioni sperimentali e valutazione

"Bambino: Questo è l'ultimo avviso per voi e i vostri rubagalline
Il pistolero si alza: Che avete detto?
Bambino: RUBAGALLINE
Il pistolero si risiede: Aaah."

Lo chiamavano Trinità ...

Si mostra il progetto dal punto di vista sperimentale, le cose materialmente realizzate. In questa sezione si mostrano le attività sperimentali svolte, si illustra il funzionamento del sistema (a grandi linee) e si spiegano i risultati ottenuti con la loro valutazione critica. Bisogna introdurre dati sulla complessità degli algoritmi e valutare l'efficienza del sistema.

Chapter 6

Direzioni future di ricerca e conclusioni

"Terence: Mi fai un gelato anche a me? Lo vorrei di pistacchio.
Bud: Non ce l'ho il pistacchio. C'ho la vaniglia, cioccolato, fragola, limone e caffé.
Terence: Ah bene. Allora fammi un cono di vaniglia e di pistacchio.
Bud: No, non ce l'ho il pistacchio. C'ho la vaniglia, cioccolato, fragola, limone e caffé.
Terence: Ah, va bene. Allora vediamo un po', fammelo al cioccolato, tutto coperto di pistacchio.
Bud: Ehi, macché sei sordo? Ti ho detto che il pistacchio non ce l'ho!
Terence: Ok ok, non c'è bisogno che t'arrabbi, no? Insomma, di che ce l'hai?
Bud: Ce l'ho di vaniglia, cioccolato, fragola, limone e caffé!
Terence: Ah, ho capito. Allora fammene uno misto: mettici la fragola, il cioccolato, la vaniglia, il limone e il caffé. Charlie, mi raccomando il pistacchio, eh."

Pari e dispari

Si mostrano le prospettive future di ricerca nell'area dove si è svolto il lavoro. Talvolta questa sezione può essere l'ultima sottosezione della precedente. Nelle conclusioni si deve richiamare l'area, lo scopo della tesi, cosa è stato fatto,come si valuta quello che si è fatto e si enfatizzano le prospettive future per mostrare come andare avanti nell'area di studio.