Instantaneous Multi-Sensor Task Allocation in Static and Dynamic Environments

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Diego Pizzocaro

December 2011

Cardiff University
School of Computer Science & Informatics

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

To Gina, Gabriele, Chiara and Barbara
for their patience and support.

Abstract

A sensor network often consists of a large number of sensing devices of different types. Upon deployment in the field, these sensing devices form an ad hoc network using wireless links or cables to communicate with each other. Sensor networks are increasingly used to support emergency responders in the field usually requiring many *sensing tasks* to be supported at the same time. By a sensing task we mean any job that requires some amount of sensing resources to be accomplished such as localizing persons in need of help or detecting an event. Tasks might share the usage of a sensor, but more often compete to exclusively control it because of the limited number of sensors and overlapping needs with other tasks. Sensors are in fact *scarce* and in *high demand*. In such cases, it might not be possible to satisfy the requirements of all tasks using available sensors. Therefore, the fundamental question to answer is: “Which sensor should be allocated to which task?", which summarizes the *Multi-Sensor Task Allocation* (**MSTA**) problem.

We focus on a particular MSTA instance where the environment does not provide enough information to plan for future allocations constraining us to perform *instantaneous allocation*. We look at this problem in both *static setting*, where all task requests from emergency responders arrive at once, and *dynamic setting*, where tasks arrive and depart over time. We provide novel solutions based on *centralized* and *distributed* approaches. We evaluate their performance using mainly simulations on randomly generated problem instances; moreover, for the dynamic setting, we consider also feasibility of deploying part of the distributed allocation system on user mobile devices. Our solutions scale well with different number of task requests and manage to improve the *utility* of the network, prioritizing the most important tasks.

Acknowledgements

This research was partly sponsored by the U.S.Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This research was also supported by EPSRC Doctoral Training Account and for this reason I would like to acknowledge also the EPSRC for their economic support.

I would like to express my deep gratitude to my PhD supervisor Professor Alun D. Preece for his guidance and support during the past four years. Alun has always pushed me to the limit and thanks to his Jobs-ian reality distortion field he made me achieve goals that I thought were unreachable. He guided me through the PhD leaving me the freedom to work on what I was most passionate about and at the same time helping me to cut the diversions I had taken during research. He supported me through a few thesis blues and he managed to help me regain confidence. I owe a lot to him and I have learned even more from his wide research experience and his ability to manage research projects and deliverables. During our meetings he would often see already what I was trying to explain and he would get the ideas immediately, often preceding what I was going to say or saying it at the same time. Alun has always been extremely

responsive to my emails, giving the impression he was just doing that even if actually managing other million things. I also would like to thank him for having given me the opportunity to travel for both conferences and collaborations with international researchers, in particular for the confidence he had in me since the beginning that I would do a good job in presenting our research to others. Thanks to this my confidence in presenting my research has reached levels I would have never imagined. For all these reasons Alun will always be one of my greatest mentors in my research career and I would like to express once again all my gratitude for his supervision, support and his many personal advices. A sincere thanks goes also to Prof. Roger Whitaker who has been my second supervisor and although we did not meet very often he was always available to meet me and to give me good advice on both my research and future career.

I would like to especially thank Prof. Tom La Porta, and Prof. Amotz Bar-Noy for having welcomed me to their institutions when I was visiting both Penn State University and City University of New York. But also for their incredible knowledge and advices about my research and my career. I feel very honoured to have been given the possibility to work with them, their passion and insights have hugely helped me to improve both my research and my way of collaborating with other international researchers. Both Prof. La Porta and Prof. Bar-Noy have been extremely responsive to my emails, again almost as if they were only occupied answering me. Together with Alun, they are the best mentors I could have wished to have during my PhD.

A thanks goes also to Gavin Pearson (DSTL, UK MoD), Tien Pham (US ARL) and Dr. Lance Kaplan (US ARL) for their expertise in Intelligence, Surveillance and Reconnaissance operations and their feedback on my work. I would also like to thank all the ITA (International Technology Alliance) project members for their feedback during our frequent plenary meetings in both US and UK.

A huge thanks goes also to Dr. Matthew P. Johnson (City University of New York and then Penn State University), Yosef Alayev (City University of New York), Dr. Hosam Rowaihy (Penn State University and then King Fahd University of Petroleum

and Minerals) and Fangfei Chen (Penn State University) for the helpful and insightful discussions. I would also like to thank Geeth De Mel (University of Aberdeen), Dr. Murat Sensoy (University of Aberdeen), Dr. Mario Gomez (University of Aberdeen and then Institut d'Investigacio en Intelligencia Artificial), Prof. Tim Norman (University of Aberdeen), Dr. Wamberto Vasconcelos (University of Aberdeen) and Dr Stuart Chalmers for their interest in my work and the extremely helpful comments and discussions. Recently I was lucky enough to work also with Dave Braines (IBM UK Hursley Emerging Technology Services), he is one of the smartest and most interesting persons I have met and he has supported both my work and my career with an incredible passion, I would like to express him my profound gratitude. Many of these international researchers have been my coauthors in research papers and I consider myself to be very lucky to have worked with all of them. Note that this thesis is based on works which were the result of international collaborations with many of these authors and in the introduction I highlight my contribution to the collaborative work that we have achieved together.

I would like to give a personal thanks to the researchers in Cardiff University in both the School of Computer Science and Informatics, and the School of Physics and Astronomy. In particular Matthew J. Williams, Konrad Borowiecki (and his wife Marta), Dr. Lorenzo Moncelsi, Dr. Stefanie Walch, Dr. Giorgio Savini, Dr. Jin Zangh and Dr. Ahmed Alazzawi, Chris Gwilliams, Rich Coombs, Will Webberley, Dr. Alysia Skilton, Martin Chorley, Dr. Walter Colombo for their discussions on my research, their personal advices and their friendship. I would like to particularly thank Matthew J. Williams for having proofread most of this thesis. A thanks goes also to all the PhD students in the School of Computer Science and Informatics at Cardiff University who have been very friendly and helpful giving me both research and teaching advices.

A thanks goes also to all my friends in Cardiff (my adoptive hometown) for their patience and the moments we have shared in Cardiff. Some but not even all of them are: Michele De Benedetti, Lydia Lagartija, Giancarlo Russo, Ieva Bisi, Baljinder (Bal)

Bains, Pilar Clemente-Fernandez and many others. Together with them I would like to thank my friends in my hometown (Padova, Italy), they have in fact always welcomed me back home during my holidays as if I had almost never left home. I would also like to thank Fabio Maran and Filippo Zanella for the collaborative work we have done together, in order to apply our knowledge in sensor networks to a very important problem in Italy, i.e. the development of the hi!tide Venice mobile app with which citizens and tourists in Venice can now monitor the regular tides flooding the city.

Finally the greatest thanks goes to my father Gabriele, my mother Gina, my sister Chiara and my loving girlfriend Barbara. They have supported me and encouraged me to carry on my research with a lot of patience, bearing me also during the worst periods and celebrating my successes and accomplishments. I would not even know how to thank my girlfriend Barbara, she has been amazing, unbelievably supporting, she helped me so much and I owe her such a huge part of my achievements that the only thing I can hope is that we spend as much time as possible together.

Thank you all. - Diego

Contents

List of Publications

The work presented in this thesis is based on the following publications. Many of these can be downloaded at `http://users.cs.cf.ac.uk/D.Pizzocaro`.

Journal Articles

1. M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. BarNoy, S. Chalmers, T. La Porta, and A. Preece. Sensor-mission assignment in constrained environments. IEEE Transactions on Parallel and Distributed Systems, 21(11):1692–1705, November 2010.

Refereed Conference Papers

1. D. Pizzocaro, A. Preece, F. Chen, T. La Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation. In 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011), pages 1–8. IEEE, June 2011.

2. D. Pizzocaro, A. Preece, F. Chen, T. La Porta, A. Bar-Noy, System architectures for multi-sensor task allocation. In Proceedings of the 4th Annual Conference of the International Technology Alliance (ACITA 2010), Imperial College, London, UK, 2010

3. H. Rowaihy, M. P. Johnson, D. Pizzocaro, A. Bar-Noy, L. Kaplan, T. La Porta, and A. Preece. Detection and localization sensor assignment with exact and fuzzy locations. In Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2009), pages 28–43, Berlin, Heidelberg, 2009. Springer-Verlag.

4. M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. Bar-Noy, S. Chalmers, T. La Porta, and A. Preece. Frugal sensor assignment. In Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS 2008), pages 219–236, Berlin, Heidelberg, 2008. Springer-Verlag.

Refereed Conference Poster Papers

1. D. Pizzocaro, A. Preece, F. Chen, T. La Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation: demo. In 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011). Barcelona, Spain, June 2011. IEEE.

2. D. Pizzocaro & A. Preece, Towards a taxonomy of task allocation in sensor networks. In Proceedings of the 28th IEEE international conference on Computer Communications (INFOCOM 2009) Workshops, pages 413–414, Rio de Janeiro, Brazil, April 2009. IEEE.

List of Figures

List of Algorithms

Glossary

MSTA Multi-Sensor Task Allocation problem

MRTA Multi-Robot Task Allocation problem

ST Single Task robots or sensors

MT Multi Task robots or sensors

SR / SS Single Robot or Single Sensor tasks

MR / MS Multi Robot or Multi Sensor tasks

IA Instantaneous Assignment

TE Time Extended assignment

HO Homogeneous sensor network

HE Heterogeneous sensor network

ST-MR-IA Instantaneous Assignment of Single Task robots to Multi Robot tasks

MT-MR-IA Instantaneous Assignment of Multi Task robots to Multi Robot tasks

ST-MS-IA Instantaneous Assignment of Single Task sensor to Multi Sensor tasks

MT-MS-IA Instantaneous Assignment of Multi Task sensor to Multi Sensor tasks

ST-MS-IA-HE Instantaneous Assignment of Single Task sensors to Multi Sensor tasks in a Heterogeneous sensor network

MT-MS-IA-HE Instantaneous Assignment of Multi Task sensors to Multi Sensor tasks in a Heterogeneous sensor network

MRS Multi Robot System

WSN Wireless Sensor Network

Mote Miniaturized sensor composed of a low cost, low power processing unit which monitors one or more sensing peripheral devices (e.g. seismic, acoustic, magnetic, IR, video, etc.). These are usually static sensors with wireless communication antennas powered by a battery, making them very resource constrained. To be used, motes are usually scattered around a sensing field to collect information about their surroundings.

UAV An Unmanned Aerial Vehicle is an unpiloted aircraft, which can be remote controlled or fly autonomously based on pre-programmed flight plans or more complex dynamic automation systems. A single UAV can be equipped with many different types of sensing devices, like video or IR sensors.

UGV An Unmanned Ground Vehicle is an unpiloted mobile robotic platform, which can move autonomously or be remote controlled. They are very similar to UAVs, being able to carry many different types of sensing devices at once; they are specifically designed to move on the ground over a wide variety of terrain.

GAP Generalized Assignment Problem

MRGAP Multi-Round Generalized Assignment Problem algorithm

PTAS Polynomial-Time Approximation Scheme

FPTAS Fully Polynomial-Time Approximation Scheme

JUM Joint Utility Model

BT Bundle Type

KB Knowledge Base or Knowledge-based

ER Entity-Relationship

Chapter 1

Introduction & Motivation

A sensor network often consists of a large number of sensing devices of heterogeneous types. These can vary from very simple *sensors* with limited capabilities such as motes, to very complex *platforms* such as Unmanned Aerial Vehicles (UAV) on which many sensors can be mounted[1]. Upon deployment in the field, these sensing devices form an ad hoc network using wireless links or cables to communicate with each other and with data processing centres. *Simple sensors* are in general very resource constrained, mainly in terms of lifetime and computational power. For instance, sensing motes are usually equipped with a small processor, a wireless communication antenna and are battery powered. *Platforms* instead can be equipped with a battery or with a fuel/propulsion engine, but also in this case we need to be careful not to shorten their lifetime as they might be required to operate for a very long time[2]. Considering these limitations it is clear that sensor networks — for simplicity we use the term *sensors* for both simple sensors and platforms — are extremely resource constrained environments.

Sensor networks are increasingly used to support emergency responders in the field usually requiring many *sensing tasks* to be supported at the same time [6, 87, 88, 117]. By sensing task we mean any job that requires some amount of sensing resources to be accomplished, such as video monitoring a collapsing building, tracking persons in

[1] See Glossary for a full definition of both.

[2] E.g., a long endurance UAV is usually expected to fly for over 30 hours, even if the maximum flight duration varies widely based on the type of aircraft [23].

need of help or localizing an event. As an example of such usage, consider the recent case of the Haiti earthquake[3], the Hurricane Katrina disaster in New Orleans[4] and the latest Japan Nuclear crisis [5]. In all these cases military forces used sensing platforms (UAVs) and various seismic, video and acoustic sensors to scan the disaster areas and organize humanitarian relief operations.

We model sensing tasks and sensors considering domain expertise from emergency operation specialists, intelligence analysts and military advisors reported in [87, 88, 117, 118]. We represent each sensing task with a *demand* which measures its need for sensing resources, a *profit* which represents its importance or priority over other tasks and a *location* in the field. In certain cases tasks can be divided into multiple sub-tasks, so for example monitoring a large field can be divided into monitoring multiple smaller areas. Each *sensor* or group of sensors, which we call *sensor bundles*, can provide a different *utility* (or amount/quality of information) to each task depending on several factors. These include the type of sensors, their sensing ranges, their geographic locations relative to the task location and their current operational status, such as if they are already serving another task.

1.1 The Multi-Sensor Task Allocation Problem

Sensing tasks might share the usage of a sensing resource, but more often they compete to exclusively control it because of the limited number of sensors and overlapping needs with other tasks [87, 88]. Sensors are in fact *scarce* and in *high demand* by tasks which users might request in the field. In such cases, it might not be possible

[3]http://www.af.mil/news/story.asp?id=123185754 - "Global Hawk collects reconnaissance data during Haiti relief efforts", last visited December, 2011.

[4]http://www.nsf.gov/news/news_summ.jsp?cntn_id=104453 - "Small, Unmanned Aircraft Search for Survivors in Katrina Wreckage", last visited December, 2011.

[5]http://online.wsj.com/article/SB10001424052748704530204576232881053381842.html - "U.S. Military Deploys Drones Above Fukushima, Libya", last visited December, 2011.

to satisfy the requirements of all sensing tasks using available sensors. Therefore we need schemes to automatically decide which sensors should be allocated to which task, maximizing the *utility* of sensors to tasks. In general there might be many different problem settings, sensors might be assigned to one or multiple tasks and tasks might require one or multiple sensors, but the fundamental question which we need to answer is: “Which sensor should be allocated to which task?". This summarizes the problem that we call *Multi-Sensor Task Allocation* (**MSTA**).

Consider for example the search-and-rescue scenario of Figure 1.1, where the sensor network has to support an *identification task* of people in need of rescue, and at the same time a wide area *surveillance task* to detect possible threats to the life of the injured people. If only one UAV is available to be allocated to a task, the question to answer is: “Where is it better to send that particular UAV?". To answer this question, where the implied conflicting need is spatial, we should consider the *estimated utility* (or amount/quality of information) that the UAV can provide to each different task and their *profits* (or importance). Our aim is to assign the UAV to the task which can serve best (i.e. with highest utility) and which is most important (i.e. with highest profit). The problem of course becomes hard to solve in the presence on the field of hundreds of sensors supporting tens of sensing tasks, which might be competing for the usage of the same sensing resources.

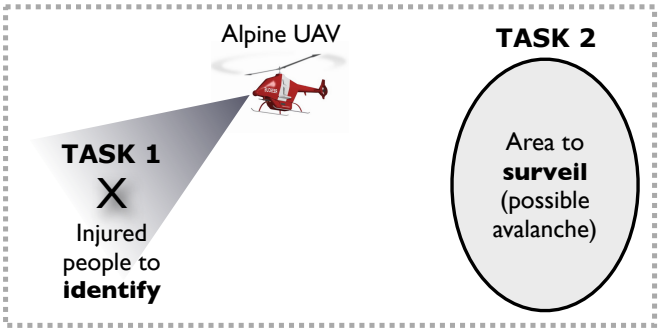

Figure 1.1: Example of Multi Sensor-Task allocation problem.

Note that from the perspective of a potential *mobile user* who is moving on the field and needing different sensing tasks, there would be no time to manually decide what

is the best set of sensors to use for each task, as highlighted in [87]. In addition, a user might not have the expertise to decide what type of sensors could best match each task's sensing requirements. Moreover, if they were to manually choose the sensors, they would consider only a subset of the sensor and task parameters. For this reason, we need to develop schemes to automatically allocate each sensor to the task they best serve.

The MSTA problem can arise not only in emergency response scenarios but also in many humanitarian relief operations requiring a multi-agency international response. In this *coalition context* the multi-sensor task allocation problem is even more evident; because users will have little idea of what sensor assets are available across the coalition, or what their ownership and access rights might be [88]. Hence the need to develop automatic mechanisms to solve the MSTA problem in such scenarios, in which it is extremely important to consider the requirements and priorities of different tasks with the capabilities and current status of each sensor, in order to decide which task should be supported.

MSTA is in general related to the *Multi-Robot Task Allocation* problem, the *sensor selection* problem and the *multi-agent coalition formation* problems, all addressing intelligent resource allocation in different environments. The Multi-Robot Task Allocation (MRTA) problem [40] aims to allocate robots to sensing and actuating tasks in a generic Multi-Robot System (MRS), here instead we focus on sensing devices with no actuator capabilities. Given that *sensors* can be seen as *resource constrained robots*, as suggested by [73, 130], MSTA can be considered a constrained version of MRTA. In fact, when supporting users in the field, sensors form ad-hoc networks which require more energy-aware approaches in the allocation mechanisms [86, 132], moreover sensors might be deployed in dangerous locations where it might be difficult to replace batteries. Given that network communication is usually the most expensive operation in terms of energy consumption, it follows that automatic allocation mechanisms need to minimize the number of messages exchanged to find a feasible solution. Another

important reason for limiting the traffic generated by the allocation mechanism is to reserve bandwidth and battery for retrieving and delivering sensor data.

MSTA is also related to the general problem of *sensor selection* in which we usually need to choose a set of "active" sensors to achieve a particular objective. This has received considerable attention lately in the wireless sensor network research community. For example, in [89, 116] the authors focus on the coverage problem with the goal to select the least number of sensors while monitoring an area and conserving overall the largest energy quantity. Another sensor selection problem, related to MSTA, is to efficiently locate and track targets such as in [59, 60, 133]. The problems considered in this thesis differ from these as we consider multiple tasks possibly competing for the exclusive usage of the same sensing resources, while instead in sensor selection the focus is usually on supporting single tasks or in some cases multiple tasks of the same type (i.e. all monitoring tasks) which can share sensor usage.

Finally, the Multi-Sensor Task Allocation problem instances that we consider can be modeled as a *multi-agent coalition formation problem* which has been extensively studied in [109, 115], where autonomous entities ("agents") can form a coalition to achieve one common goal. Agents are equivalent to sensors with regards to MSTA, which grouped into sensor bundles can satisfy a particular sensing task. In recent works [24, 70, 100], multi-agent coalition formation techniques were applied to resource allocation in sensor network. In this thesis, we also experiment on applying such a technique (among others) to solve a particular instance of MSTA. With respect to previous works in this area, we propose a novel integration of both qualitative (i.e. knowledge based) and quantitative measurements (i.e. based on numerical utility values) to evaluate the fitting of a particular sensor bundle for a certain task. We also include users' mobile devices as part of the allocation mechanism.

1.2 Research Question

In this thesis, we focus on a particular scenario motivated by emergency response operations. Such an environment does not usually provide enough information to plan for future allocations, in fact there might be frequent changes in the plan of a particular operation and usually no detailed unified plan of action is available if a multi-national coalition is involved [88, 117]. For this reason, we focus on *Instantaneous Allocation* (*IA*), as opposed to a time extended allocation of sensors to tasks, meaning that we must allocate sensing resources instantaneously without scheduling or reserving them for future allocation. In fact, if an unexpected more important and critical task needs to be supported, we should assign to it the largest number of sensing resources without saving such resources for future tasks which might be dropped (e.g. because of a change in the plan due to an unexpected event).

Considering this scenario, we mainly study two MSTA problem instances: for a *static environment*, in which all task requests from emergency responders arrive at once, and for a *dynamic environment*, in which tasks arrive and depart over time. In the static setting, the information about all tasks, including their profits and demands, is available at once when making the allocation decisions. This problem setting is useful when the system has a set of long-lived tasks such as perimeter monitoring applications. It can also be useful in systems where sensing tasks requested at different times may be batched together and start at a single point in time. In a dynamic setting, instead, the system is expected to respond promptly to incoming tasks and should be able to adapt to changes. In such setting, tasks start at different times and have different durations.

We assume that tasks are *independent* from each other by not allowing for ordering constraints among tasks. This is a strong assumption but, as noted in [40], in principle every interconnected task may be expressed as a single larger task with more complex aggregated sensing requirements. This shifts the difficulty from reasoning about task constraints, to estimating the utility of sensor bundles for a single task. Therefore, each task might have complex sensing requirements, especially if it is the result of

the merging of multiple interconnected different tasks in the field. Given that each task might require a group of sensors to satisfy its complex information requirements, as also highlighted in [87], we then consider *Multi-Sensor tasks* (*MS*) in our MSTA problem instance, as opposed to Single-Sensor tasks requiring just one sensor. Given the potentially complex task requirements, we may need to analyze which sensor types (or combination of types) could potentially satisfy each task. In addition, we might need to use both additive and non-additive functions to estimate the sensor utilities to each task, as for example the utility of each sensor might not sum-up linearly with the individual utilities of other sensors allocated to the same task.

Another important feature of the MSTA problem studied in this thesis is the focus on a *heterogeneous sensor network*; i.e., a network composed of two or more sensors with different sensing capabilities [123, 130]. Although *homogeneous sensor networks* can offer support to different operations, often users on the field request tasks with a wide variety of sensing requirements which can be better satisfied by groups of cooperating sensors with heterogeneous capabilities [87, 88]. For example, while a "vehicle tracking" task might be achieved only through acoustic sensors by triangulating their bearings and distances from the target, for an "event detection" task we might instead use both acoustic and video sensors to maximize the detection probability. Therefore our focus is on heterogeneous sensors, i.e. devices with different sensing capabilities, and heterogeneous tasks, i.e. tasks which have different sensing requirements. In this heterogeneous environment we assume that a sensor network middleware [20] allows different sensors to seamlessly exchange messages by simply specifying the recipient, taking care of the lower layers in the OSI model and allowing us to focus on application layer mechanisms. We consider a particular subset of heterogeneous sensors which can serve exclusively one task at a time, like directional sensors – we refer to these as *Single-Task sensors* (*ST*). Note that although in this work we limit sensors to performing one task at a time, this limitation is not applicable to all emergency response domains. For some sensing data types, e.g. ambient temperature, a sensor may be able to serve all tasks within its sensing range. In a given problem instance, there

may, in fact, be sensors of both types (Single-Task and Multi-Task). In this thesis, we focus on the restricted type of Single-Task sensors since this is the more restrictive problem.

Given that MSTA can be seen as a constrained version of the Multi-Robot Task Allocation problem, we can adopt the taxonomy proposed in [40] to classify it with a triple of two letter abbreviations[6]: *ST-MS-IA*, i.e. *Single-Task sensors* serving *Multi-Sensor tasks* where the environment allows only for *Instantaneous Allocation*. In this thesis, our goal is to find an optimized solution to this Multi-Sensor Task Allocation problem instance in both *static* and *dynamic* settings in a *heterogeneous sensor network*. For this reason, we further refine the label with which we identify our problem instance as *ST-MS-IA-HE*[7], where HE stands for heterogeneous sensor network opposed to homogeneous (i.e. sensors exclusively of a single type). In order to find a solution for this problem we need to decide what exactly needs to be optimized. If we were to choose to directly optimize the overall performance of a real system solving MSTA, that quantity would be very difficult to measure. To address this issue, we use *utility* as performance estimation. Utility is a widespread concept in economics, game theory, and operations research. A utility based approach allows us to evaluate the "optimality" of the algorithms proposed in this dissertation to solve MSTA problems. By "optimal" we mean that given all the available information in the system, it is impossible to construct a solution with higher utility. This is similar to the concept of "optimality" in [15, 40] and originally to [30].

In this thesis, we refer to *utility* as the expected quality of task execution, that is the quantity/quality of useful information that a sensor node (or group of them) is expected to provide to each task. This is very similar to the concept of utility which has been adopted in sensor selection works, such as [15, 106], and the choice of using the

[6]Note that differently from [40], we use the term "sensor" instead of "robot" given our problem settings.

[7]We refer to Chapter 2 for a more detailed explanation of this extended classification scheme for MSTA problems, i.e. MSTA taxonomy.

same notion of utility was motivated by the wish to align our work to the sensor network research field. Therefore, the concept of utility in this dissertation differs slightly from other formalizations, such as the one adopted in the Multi-Robot Task Allocation framework [40] in which the utility is defined as the difference between the expected quality of task execution and the expected resource cost. From this point of view, it is important to note that we also consider in our MSTA problems the notion of *cost* but we keep it distinct from the concept of utility, again to remain aligned with the terminology used in previous works on sensor selection. In particular, for the static settings, we associate to each sensor a *cost* and to each task a *budget*, e.g. in terms of energy, to constrain the usage of sensing resources by the set of active tasks on the field. For the dynamic settings, where tasks appear over time, we consider a cost associated to each *sensor bundle* which represents the operational cost associated with the reassignment of already allocated sensors to newly generated tasks; basically penalizing new tasks which require sensing resources currently supporting other tasks, following directions outlined by emergency operation specialists reported in [87, 88, 117, 118]. Note that in this setting, we assume there are no budgets associated to tasks as these may be too restrictive in a dynamic environment where we must react to new tasks and in which condition of sensors will change over time. Finally, the concept of *profit* associated to each task is of core importance in our models: it represents the importance or priority of supporting a particular task, independently from the utility which could be potentially allocated to them. This is an additional problem parameter compared to [40] which in its general framework does not consider an absolute importance or priority metric associated to each task. This notion of profit is instead again very similar to some works in the sensor selection field, such as [106, 114], but most importantly the introduction of profit in MSTA (e.g. compared to [40]) is motivated by the wish to model a typical emergency response scenario in a coalition environment following recommendations by domain experts [87, 88]. Provided that sensing resources are *scarce*, we will not be able to support all tasks on the field which implies the need to drop (i.e. let unsatisfied/unsupported) a subset of tasks. Tasks in this subset should ideally have minimal

potential sensor utilities and be the least important. Therefore, our goal is to design automatic allocation mechanisms which maximize overall sensor utilities and also prioritize the most important tasks.

Given these assumptions, our research question can be stated as following:

Research Question: *"To what extent and how is it possible to develop automatic mechanisms for Instantaneous Allocation (IA) of Single-Task (ST) sensors to Multi-Sensor (MS) tasks in a Heterogeneous sensor network (HE) which can maximize the overall sensor utilities and prioritize the most important tasks, in both static and dynamic settings?"*

This can be split into two sub-questions, describing in particular what we are going address in the rest of this thesis:

Research Question 1: *Considering a static setting, with a simplified additive sensor utility model and task requirements based only on each task's utility demand and energy budget, to what extent and how is it possible to find a scalable solution which maximizes the sensor utilities and prioritizes the most important tasks?*

Research Question 2: *Considering a dynamic setting, with general (possibly non-linear) sensor utility models and richer knowledge-based task requirements, to what extent and how is it possible to find a scalable solution which maximizes the sensor utilities and prioritizes the most important tasks?*

Note that these two questions gradually address a harder version of the *ST-MS-IA-HE* problem. We start from a relatively simplified static setting using an additive utility model to estimate "how good" is a particular sensor bundle for serving a task, and considering exclusively a *quantitative metric* based on the sensor *utility values* for a particular task. We then address *ST-MS-IA-HE* in a dynamic scenario where we allow for general non-linear utility functions and more complex sensing requirements for each task considering also *qualitative metrics* based on the fitting of sensor capabilities

with task requirements, which are *knowledge-based*.

Finally, it is worth noting that by scalable we mean a solution whose performance in terms of allocation quality does not drastically decrease in the presence of a large number of tasks in the field or a high task arrival rate in the dynamic setting. Although in this dissertation the focus is on *instantaneous allocation*, mechanisms designed for our MSTA instance need also to provide timely solutions. In particular, we will see how our MSTA instance in both static and dynamic settings contains NP-hard problems. This implies that the computational time required to find an optimal solution is always likely to grow exponentially with the size of the problem instance in terms of the number of sensors and tasks on the field (unless $P = NP$). Therefore, although our focus remains on designing automatic allocation mechanisms able to find an approximated optimized solution in terms of the sensor utility provided to the most important tasks on the field, we also consider the time aspects for each of the mechanisms presented. In particular, for the static environment we consider the computational time required to find a solution, while for the dynamic setting we study the number of important tasks on the field which are kept waiting (on average) for a decision to be reached by the dynamic allocation protocol.

1.3 Thesis Contributions

We propose a variety of *centralized* schemes for the static setting, and a *distributed* mechanism for the dynamic setting, both aiming at maximizing the overall utility of the network which we evaluate mainly using simulations on randomly generated problem instances. A centralized approach typically collects all the information in a single node, for example a *base station*, where all assignment decisions are made by an algorithm having a global view of the network. Due to its global view of the field, this approach can provide high quality solutions, but can be expensive in terms of communication overhead and introduces a single point of failure. Distributed approaches

instead require just local knowledge about the surroundings of sensors and sensing tasks, being therefore more suitable for actual implementations.

For static and dynamic problem setting, we address different constraints and design specific solutions for each. In particular for the static case, we consider tasks with a budget (e.g. monetary or energy) and a threshold on the utility demand, that is a task is allocated sensors only if the total sensor cost (monetary or energy) does not exceed its budget and a minimum utility (expressed as a threshold on its demand) is reached. In this setting, we assume additive sensor utilities, i.e. utilities of sensors contributing to the same task add up linearly. For the dynamic settings instead, we allow for non-linear sensor utilities, and we integrate these with knowledge-based metrics about sensor capabilities and task requirements in order to better deal with heterogeneous sensors and tasks. We do not constrain the budget of tasks, considering that in a dynamic environment we need more flexibility in the allocation of resources to each task, because we lack exact information about what will happen in the future. We also formalize two examples of non-linear utility models for event detection and 2D localization tasks. For both the static and dynamic MSTA problems, our practical algorithms manage to improve the utility of the network and in many cases achieve near optimal performance.

We summarize our research major contributions in the following list:

1. We formalize the *ST-MS-IA-HE* instance of the MSTA problem in *static settings*. We explore a simplified model in which sensor utilities are additive; i.e., they sum up linearly if sensors are allocated to the same task. We also constrain each task usage of sensing resources by assigning them a budget (e.g. energy or monetary). Each task is characterized by a demand of utility and a threshold on its demand to be reached in order to be satisfied. For this setting we use only a quantitative utility measurement, based only on physical attributes of the sensor instance (such as distance from a task).

2. We propose a variety of *centralized algorithms* for our *ST-MS-IA-HE* model in static settings. We evaluate the different algorithms on randomly-generated prob-

lem instances. In particular, the experiments show that the best performance are offered by our novel MRGAP algorithm, which uses a Generalized Assignment Problem (GAP) algorithm extended with a Multi-Round heuristic.

3. We formalize *ST-MS-IA-HE* in *dynamic settings*, allowing for more general sub-additive or super-additive sensor utilities; i.e., which might not sum up linearly when contributing to the same task. We assume unlimited budget for each task, and we allow for re-allocation of already assigned sensors to newly generated more important tasks. We limit the preemption of sensors from ongoing tasks through defining a re-allocation cost which we want to minimize.

4. Finally, we present a novel *distributed architecture* to solve *ST-MS-IA-HE* in dynamic settings, which uses both knowledge-based metrics and numerical utility functions to deal with knowledge rich task requests. We extend a pre-existing distributed coalition formation protocol and deploy part of the architecture on mobile devices showing that it is feasible to do knowledge based sensor-task matching on such devices. We evaluate the performance of our architecture in two ways. First, we implement a prototype mobile application which contains a knowledge-based component and we measure its performances on a real device. In addition, we run simulations and evaluate the total sensor utility over time on randomly generated scenarios with users moving on the field and requiring sensing tasks to be satisfied by the sensors composing the network.

In addition, we also list two minor contributions which are related to the above major contributions:

(A) We formalize from the literature two non-additive *sensor utility models*, with the aim of showing how it is necessary to consider non-linear utility functions. We use these as motivations to allow for non-linear sensor utilities in our dynamic MSTA model (Contribution 3); we also adopt these as examples of utility functions to evaluate the performance of the distributed mechanism for *ST-MS-IA-HE* in the

dynamic setting (Contribution 4). For *event detection*, we propose a model based on cumulative detection probabilities; for *2D localization* tasks, we propose a model based on the distance and bearings of a pair of sensors to a task.

(B) We propose a *conceptual architecture* to solve *ST-MS-IA-HE* in dynamic settings for generic sensor utility models which uses a combination of qualitative (i.e. knowledge based) and quantitative (i.e. numerical utility based) metrics to drive the allocation of sensors to tasks. We identify the necessary and sufficient components for this architecture, which we then implement as a distributed system (Contribution 4). We describe how mobile smart-devices could be used to deploy components of such an architecture and, in particular, allow a user to submit tasks to the system through a mobile app.

Below we list the papers in which we have previously published parts of these contributions. We include a brief summary of each publication, together with the role of the author of this thesis in each of them with respect to the other co-authors. We also include a reference to the chapters in which each publication is contained.

- *D. Pizzocaro & A. Preece, Towards a taxonomy of task allocation in sensor networks. In Proceedings of the 28th IEEE international conference on Computer Communications (INFOCOM 2009) Workshops, pages 413–414, Rio de Janeiro, Brazil, April 2009. IEEE.*

  In this refereed short paper, we propose an initial taxonomy of MSTA problems as an extension of a pre-existing MRTA taxonomy, in order to identify the most relevant features of the MSTA problem instances analyzed in this dissertation. The taxonomy focuses on the issue of homogeneous versus heterogeneous sensor networks. We use this taxonomy in Chapter 2 to highlight that MSTA in heterogenous sensor networks requires knowledge-based matching to deal better with sensor and task heterogeneity. In other words, to evaluate the utility of sensors we need a knowledge repository containing information about

which sensor types (and therefore sensor capabilities) or combinations of them can serve a particular task type. This work led eventually to the novel conceptual architecture described in Contribution B which integrates knowledge-based measures with numerical sensor utilities. This paper was also an important step in order to formally model *ST-MS-IA-HE* in both static and dynamic settings, it therefore spans both Contribution 1 and 3.

- *M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. BarNoy, S. Chalmers, T. La Porta, and A. Preece. Sensor-Mission assignment in constrained environments. IEEE Transactions on Parallel and Distributed Systems, 21(11):1692–1705, November 2010.*

  In this journal article, we address *ST-MS-IA-HE* in static settings with budget constraints for tasks. Note that in this publication we use the word *mission* to refer to sensing tasks. The author of this thesis contributed mainly to formally model the instance of MSTA in the static setting (Contribution 1) with linear sensor utilities and budget constraints, and in developing the centralized algortihms (Contribution 2) to solve it. This work is contained in Chapter 3.

  A preliminary version of this work appeared also as
  M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. Bar-Noy, S. Chalmers, T. La Porta, and A. Preece. Frugal sensor assignment. In Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS 2008), pages 219–236, Berlin, Heidelberg, 2008. Springer-Verlag.

- *D. Pizzocaro, A. Preece, F. Chen, T. La Porta, A. Bar-Noy, System Architectures for Multi-Sensor Task Allocation, Proceedings of the 4th Annual Conference of the International Technology Alliance (ACITA 2010), Imperial College, London, UK, 2010*

  In this refereed conference paper, we propose a conceptual architecture to solve *ST-MS-IA-HE* in dynamic settings (Contribution B), describing how we can carry out knowledge based allocation decisions and the different implementation strategies

for such architecture. We identify the necessary and sufficient components and we discuss a centralized, distributed and hybrid implementation by highlighting pros and cons of each. This work is mainly reported in Chapter 4, and inspired the distributed implementation of the system in Chapter 5.

- *H. Rowaihy, M. P. Johnson, D. Pizzocaro, A. Bar-Noy, L. Kaplan, T. La Porta, and A. Preece. Detection and localization sensor assignment with exact and fuzzy locations. In Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2009), pages 28–43, June 2009.*

  In this conference paper, we study two non-additive sensor utility models for event detection and target localization sensing tasks (Contribution A). We develop formal models for both, taking inspiration from pre-existing literature with the help of a sensor analyst advisor (Dr. Lance Kaplan). The main contribution of the author to this paper was in the formalization of the three models and in coordinating the efforts of the other co-authors. This material is mainly contained in Chapter 4.

- *D. Pizzocaro, A. Preece, F. Chen, T. La Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation. In 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011), pages 1–8. IEEE, June 2011.*

  In this refereed conference paper, we formalize *ST-MS-IA-HE* in dynamic settings (Contribution 3) allowing for more general sensor utilities, i.e. sub-additive or super-additive. We assume unlimited budget for each task, and we allow for re-allocation of already assigned sensors to newly generated more important tasks. We present a distributed implementation (Contribution 4) of our conceptual architecture (Contribution B), proposing an extension of a pre-existing protocol and developing a prototype mobile app on a real device as part of our distributed architecture. We report the content of this paper in Chapter 4 and

Chapter 5.

A live demo of the distributed system was presented at the IEEE DCOSS 2011 conference, and published as a short paper in the proceedings as:

D. Pizzocaro, A. Preece, F. Chen, T. La Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation: demo. In 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011). IEEE, June 2011.

Various versions of this demo were performed for audiences including UK Ministry of Defence and US Department of Defense directors, senior NATO (North Atlantic Treaty Organization) scientists and sensor analysts, as well as UK & US industry and academia. Finally, a video of this demo can be found at the following URL: http://www.youtube.com/watch?v=QzrpKRhGFRU

1.4 Thesis Structure

The rest of this dissertation is divided as follows. Chapter 2 provides an overview of research related to the Multi-Sensor Task Allocation problem and in particular to the *ST-MS-IA-HE* problem instance. Chapter 3 introduces *ST-MS-IA-HE* in a static environment focusing on additive sensor utilities and budget constraints for tasks, where we also propose and evaluate three novel centralized algorithms. Chapter 4 proposes a formal model for *ST-MS-IA-HE* in a dynamic environment with general non-linear sensor utilities and a re-allocation mechanism, generalized starting from two case studies for event detection and two-dimensional localization tasks. We also describe a conceptual architecture which can solve the dynamic *ST-MS-IA-HE* problem using both quantitative and qualitative measures. Chapter 5 then focuses on a distributed implementation of such architecture, which combines an allocation protocol with a knowledge-based component deployed on users' mobile devices. Chapter 6 concludes this dissertation summarizing the contributions and future research directions.

Following we provide a brief overview of the core chapters of this thesis.

Chaper 3: Static MSTA with Additive Sensor Utilities & Budget Constraints

In Chapter 3, we focus on an initial static MSTA problem instance (*ST-MS-IA-HE*) where all the tasks arrive at once. We assume that the utility of sensors to tasks is additive; i.e., utilities of sensors contributing to the same task add up linearly. This problem can be modeled as a bipartite semi-matching problem. More specifically, each task is associated with a demand value and a profit value; each sensor-task pair is associated with a utility offer. We assume that the utility provided by the sensors assigned to a task needs to be greater than a threshold for the task to be successful. To include realistic features of an emergency response scenario, we assume there is a limited number of sensors that can contribute to a task. We use a simple function to evaluate the sensor utility which models the real degradation of the sensing performances with the varying of the distance between sensor and task. We associate each task with a budget to limit the number of sensors they can use; in this case, sensors will have associated cost. We therefore consider both packing (i.e. budget) and covering (i.e. demand threshold) constraints, which make our static MSTA problem non-trivial to solve. As discussed in Chapter 2, our problem model differs from previously addressed resource allocation problems in sensor networks (e.g. [106]) as their assumption is usually to consider either packing or covering constraints separately.

For this static setting, we propose three centralized algorithms to address the problem. The first algorithm we consider is a simple centralized greedy algorithm that repeatedly attempts to satisfy the tasks with the highest potential profit. The second algorithm, Multi-Round GAP (MRGAP), is a novel algorithm that treats tasks as knapsacks that together form an instance of the Generalized Assignment Problem (GAP). The third algorithm is instead an adapted combinatorial auction algorithm. Through simulations we find that in dense networks the three algorithms perform well, with the GAP-based algorithm outperforming the greedy and the combinatorial auction based algorithm.

We also study how tasks spend their budgets and how increasing the budget affects the overall profit of the network. We find that in most cases giving tasks higher budgets does not help since what limits the amount of achievable profits is competition and not the amount of budget provided. Finally, we show that the three algorithms provide timely solutions to the static MSTA problem. We find that the combinatorial auction based algorithm requires the longest time to converge and the time grows with a steeper slope compared to MRGAP and Greedy when increasing the problem instance size. MRGAP displays a longer convergence time compared to the greedy algorithm, but a much slower increase in the computational time when the instance size grows compared to the auction based approach.

Chapter 4: Dynamic MSTA with Generic Sensor Utilities

In Chapter 4, we develop a formal model for the dynamic MSTA problem instance (*ST-MS-IA-HE*) where the tasks arrive and depart at different times. In this formulation, we allow for the utility of sensors to tasks to be non-additive; i.e., utilities of sensors contributing to the same task could be for example sub-additive or super-additive. We first formalize two particular static MSTA problems with non linear utilities, for event detection and two-dimensional localization tasks. We then generalize these two problems into a more abstract *static MSTA problem*, in which we allow for any *non-linear utility function* and for any task/sensor type to be used.

We then extend this into a *dynamic MSTA problem* including the time dimension in the problem formulation. This problem can be modeled as a tripartite semi-matching problem. As in Chapter 3, we associate each task with a demand value and a profit value; however, we now instead associate each *sensor bundle*-task pair with a joint utility offer. So we first need to group sensors into bundles, evaluate their joint utility using a (possibly non-additive) function, and finally we need to decide which one is the best assignment of sensor *bundles* to *tasks*. In this version of the problem, we assume there are no budget constraints for each task, and we allow for re-allocation of sensors

to newly created tasks. In order to avoid preemption of sensors from ongoing tasks to happen too often, we include a *re-allocation cost* in the formal model which we minimize while maximizing the total utility.

Finally, we propose a *conceptual architecture* to solve step-by-step the dynamic MSTA problem, using both qualitative (i.e. knowledge based) and quantitative (i.e. numerical utility based) metrics to evaluate how fit is a sensor bundle for a task. We also discuss the trade-offs related in choosing different implementations of this conceptual architecture, in particular as a centralized or distributed system.

Chapter 5: A Distributed System for Dynamic MSTA

In this chapter, we describe in detail how we implement the conceptual architecture as a *distributed system*. In particular, we describe how a knwoledge-based component combined with a distributed allocation mechanism could address the problem of joint utility evaluation in the case of heterogeneous sensors and tasks. We assume that sensor types and task types can be modelled using formal tools, such as ontologies, to express sensing capabilities and task requirements. We extend a pre-existing coalition formation protocol to implement our allocation mechanism and we show that it is feasible to do knowledge-based sensor-task matching on the user device. We also develop a prototype mobile app to show how a user could interact with the sensor network, requesting sensing tasks from the distributed system. Note that such "smart" devices are an integral part of our distributed architecture, thereby part of our distributed solution to the dynamic MSTA problem.

To evaluate the performance of our architecture, we first measure the performance of the knowledge-based component on a real mobile device (an Apple iPod Touch 2nd generation), as part of the mobile app which we developed to allow users to submit tasks to the system. Secondly, we measure the performance of the extended allocation protocol and we run simulations on randomly generated problem instances using the two non-linear utility models presented in Chapter 4 as an example of task types. Our

simulation results show that it is feasible to do sensor-task matching on the mobile device and that the distributed protocol is scalable, performing well in terms of overhead traffic and allocation quality when increasing the task arrival rate (i.e. the rate at which tasks are generated by users on the field).

Chapter 2

Background

Multi Sensor Task Allocation is in general related to the *Multi-Robot Task Allocation*, the *sensor selection*, the *multi-agent coalition formation* problems, all addressing intelligent resource allocation in different environments. The Multi-Robot Task Allocation (MRTA) problem [40] aims to allocate robots to sensing and actuating tasks in a generic Multi-Robot System (MRS), here instead we focus on sensing devices with no actuator capabilities. Given that *sensors* can be seen as *resource constrained robots*, as suggested by [73, 130], MSTA can be considered a constrained version of MRTA. In this chapter, we use the MRTA taxonomy proposed in [40] to identify some of the main features of the MSTA problem instance discussed in this thesis. In relation to this instance we analyze some of the formal models and techniques proposed for it. We look in particular at bundling mechanisms which have been previously used in order to allocate groups of resources to tasks; e.g., combinatorial auctions and multi-agent coalition formation problems.

MSTA is also related to the general problem of *sensor selection* in which we usually need to choose a set of "active" sensors to achieve a particular objective. This has received considerable attention lately in the wireless sensor network research community. Similar to [104], we survey some of these problems and divide them using the criteria of purpose of allocation. In particular we consider three categories: coverage schemes, target tracking/localization schemes and application-level task assignment schemes. MSTA is in particular related to this last group of sensor selection schemes and therefore can be considered a subset of the sensor selection problems where multiple-tasks

need to be satisfied simultaneously.

Finally, we propose a MSTA taxonomy, extending the MRTA taxonomy in [40], as a new way of looking at part of the literature in sensor selection. The extended taxonomy is based on the main issue of dealing with task allocation in heterogeneous and homogeneous sensor network. Looking at the papers surveyed in sensor selection, it seems almost taken for granted that matching the capabilities required by tasks and provided by sensors is straightforward, therefore the mechanisms developed consider most of the time an implicit representation of sensor capabilities and task requirements. We survey some of the work in sensor selection with focus on homogeneous vs heterogeneous sensor networks. Moreover, we analyze some of the architectures and modular approaches proposed for allocating sets of heterogeneous resources, in the field of MRTA, multi-agent and sensor selection schemes.

The rest of this chapter is organized as follows. Section 2.1 summarizes the MRTA taxonomy and discusses models and techniques for the *ST-MR-IA* problem. Section 2.2 covers some of the works in sensor selection and categorizes them based on the purpose of selection. Section 2.3 proposes an extension of the MRTA taxonomy for Multi-Sensor Task Allocation problems, mainly considering issues related to task allocation in a heterogeneous environment. Finally, Section 2.4 concludes the chapter highlighting the differences and similarities with the works discussed and this thesis.

2.1 Multi-Robot Task Allocation & Taxonomy

The Multi-Sensor Task Allocation problem is closely related to the Multi-Robot Task Allocation (MRTA) problem [40] which consists of answering the question “Which robot should execute which task?" in a generic Multi-Robot System (MRS); i.e., a system composed by autonomous robots with both sensing and actuator capabilities. Early work on MRTA has been empirical and ad hoc, experimentally evaluated and infrequently formally analyzed. Gerkey et al [40] offered a framework to categorize

different instances of the MRTA problem, proposing a domain-independent taxonomy of problems. They showed how many such problems can be viewed as instances of other well-studied optimization problems. The taxonomy of MRTA problems is organized on three axes:

- **Single-task robots (ST)** vs. **multi-task robots (MT)**: in ST each robot is capable of executing at most one task at a time; in MT a robot can execute multiple tasks simultaneously.
- **Single-robot tasks (SR)** vs. **multi-robot tasks (MR)**: SR means that each task requires exactly one robot to accomplish it; MR means that some task requires multiple robots.
- **Instantaneous assignment (IA)** vs. **time-extended assignment (TE)**: IA means that the information concerning the robots, the tasks, and the environment does not permit planning for future allocations, but only instantaneous allocation; in TE information is available to plan future allocations; e.g., a model of how tasks are expected to arrive over time.

To categorize MRTA problem instances it is enough to identify the corresponding triple of two-letter abbreviations from the list. For example, the label *ST-MR-IA* identifies Single-Task robots supporting Multi-Robot tasks in an environment which allows only for Instantaneous Allocation. Gerkey et al [40] propose to formally model this problem as a *Set Partitioning Problem* (which we also discuss in Section 2.1.1). The purpose of this formal analysis is that we could then apply techniques similar to the ones already designed for that well known optimization problem, or in general we could compare a more specific version of the model with the one proposed in the taxonomy.

The MSTA problem instance which we consider in this dissertation in both static and dynamic settings can be categorized as *ST-MR-IA*, with the caveat that instead of generic *robots* we consider *sensors*. Given our environment we use therefore the identifier

S (sensor) instead of R (robot), thus the problem instance which we consider in this dissertation can be identified by the label *ST-MS-IA*. The main difference here is that robots usually have both *sensing* and *actuator* capabilities, while instead sensors by definition are provided only with *sensing* capabilities and can therefore be seen as "resource constrained robots" as noted in [73, 130]. This has an impact on both the optimization problems to which we can reduce our MSTA problem instances, which will have necessarily more resource constrained features; e.g., budget or allocation costs. The second consequence of this is that the mechanisms developed to solve these problems should be computationally cheap and distributed, or alternatively they should move any expensive computation away from sensors to more computationally capable entities in the network; e.g., base stations or user devices. In this disseration, we consider these issues with a focus on the *ST-MS-IA* problem in its static and dynamic settings, proposing centralized and distributed algorithms and architectures. As discussed this problem is essentially a special case of the *ST-MR-IA* problem, in the next sections we present some of the related formal models and techniques proposed for it.

2.1.1 ST-MR-IA & MT-MR-IA

Below we discuss two instances of MRTA problems: the *ST-MR-IA* problem which is the one studied in this dissertation, and then as a comparison the *MT-MR-IA* problem which is related to the previous one and more general as robots might be shared among different tasks. Note that both problems involve allocating groups or bundles of robots to tasks (i.e. Multi-Robot tasks); these require bundling mechanisms which we discuss in Section 2.1.2.

ST-MR-IA considers Single-Task robots supporting Multi-Robot tasks. This problem is referred to as the *disjoint coalition formation* in the multi-agent community and it has been formally studied in [109] and [115]. Following [40], *ST-MR-IA* can be modeled as a Set Partitioning Problem (SPP) [9] in which is given a set of robots R and a family F of acceptable subsets of R and a utility function $u : F \rightarrow \mathbb{R}$. The objective is to find

a maximum-utility family X of elements in F such that X is a partition of R. Note that F represents the set of all feasible robot bundle-task pairs and the utility u represents the utility estimate for each pair. It is clear that for each different type of task we could evaluate utility in different ways, therefore using different estimation functions for example based on the distance of the robot or the energy required to accomplish the task. Finally note that casting the *ST-MR-IA* problem as an instance of SPP does not imply that all robots should be allocated to at least one task or that all the tasks should be assigned at least a bundle of robots, in fact the family F includes all the subsets of R where for some the utility might be zero.

If we restrict our attention to linear utilities, i.e. the utility of robots contributing to the same task adds up linearly with the others, then we can model *ST-MR-IA* as a Bipartite Weighted Semi-Matching problem [83]. Given a weighted bipartite graph $G = (R \cup T, E)$ where R is the set of robots, T the set of tasks and E is a set of edges. An edge (R_i, T_j, e_{ij}) indicates that robots R_i is capable of serving task T_j with weight e_{ij} (which in our case represents utility). Then we seek a max-weight semi-matching, i.e. a subset of edges of maximum combined weight under the restriction that no two chosen edges share an endpoint in R. This is similar to the formal models and algorithms presented for sensor selection problems in [106]. Clearly, the difference with SPP is that, in such a problem, the utility of a group of robots is calculated as the sum of the individual utilities of each robot assigned to that task, i.e. $\sum_{\forall R_i \to T_j} e_{ij}$. The model we present in Chapter 3 adopts a similar approach but adds also budget constraints to each task and, in addition, it considers a utility demand and a threshold on that to be surpassed in order for the task to be satisfied.

SPP and Bipartite Weighted Semi-Matching mainly consider a static setting, where the set of tasks is known at once. For many real-life MRTA problems and also sensor network deployments, however, tasks arrive at different points in time, and each time a task arrives, we must assign robots or sensors to it (dynamic settings). In some environments this happens knowing what is going to happen next, such as when we know the

plan of a particular mission (Time-Extended assignment); in others instead we have to take allocation desicions without knowledge of future tasks or energy constraints (Instantaneous Assignment). So we could have a dynamic setting where we can achieve a time-extended allocation or, alternatively, one where we can only perform instantaneous allocation due to the lack of information about the future, which is our case in Chapter 4 and 5. Note that instead in Chapter 3 we focus on the static setting, where we also assume instantaneous assignment.

Let us consider the dynamic settings, in the case in which these decisions are irrevocable, meaning that once a sensor is assigned, it cannot be reassigned to any future task. Authors studied different versions of this problem and in [58] such problems are referred to as *online assignment* problems. These problems could be modeled in general by the online versions of bipartite matching problems (like the one discussed previously) where nodes in T arrive over time. The performance of an online algorithm is evaluated by analyzing its competitive ratio, defined as the ratio between its worst performance and the performance of the best offline algorithm. In [58], the authors study weighted versions of the online *exact* matching problem (i.e. where each task is allocated exactly one robot). In particular, they study Online Min-Matching (minimizing edge weight) and Online Max-Matching (maximizing edge weight), in metric spaces, where edge weights are computed based on the metric distance between nodes from the two sets, e.g. tasks and robots/sensors. We need to match n robots to m tasks such that the total distance between them is minimized. Note that the difference between these problems and our dynamic MSTA problem in Chapter 4 is that in general we consider non-linear utilities which could be proportional or not to the distance of the sensor from the task. In addition, we allow for reallocation of sensors to newly arrived more important tasks.

In the case in which a robot can be shared among different tasks, the problem becomes the *MT-MR-IA* instance. Of course each robot will have a limited maximum number of tasks it is able to serve; in the case of sensors this might be due to sensory limitations.

For example, a camera could be able to detect up to a certain number of suspicious objects in a certain area, given that the camera might miss out-of-focus objects (as in [54]). *MT-MR-IA* has also been studied by the multi-agent community and it is referred to as the *overlapping coalition formation* problem [115]. Such a problem can be cast as a variant of the Set Covering Problem (SCP) [122] as follows: Given a set of robots R forming the network, a family F of acceptable subsets of R representing possible overlapping coalitions, and a utility function $u : F \rightarrow \mathbb{R}$ as an estimate of the utility of assigning a subset of robots to a task, the objective is to find the maximum-utility family $X \in F$ such that X is a cover of R. Note that also in this case casting the *MT-MR-IA* problem as an instance of SCP does not enforce that all robots should be allocated to at least one task and vice versa; in fact by definition the family F includes also subsets of R for which the utility is zero (representing unassigned robots or unsupported tasks). If full coverage of the tasks is required there could be several ways to include it in the problem formulation, but in our case we would allow for tasks being dropped due to the features of our environment in which sensing resources are scarce and in high demand. Note that we focus on *Single-Task sensors* which is the most restrictive scenario where sensors cannot be shared and need to either satisfy one task or the other. As discussed in Chapter 6 we propose to explore *Multi-Task sensor* problems as part of possible future work.

2.1.2 Bundling Mechanisms

Given our interest in the *ST-MR-IA* we look at various mechanisms proposed for similar problems dealing with coordination among different agents in order to form coalitions to achieve a common subgoal in a system. We can refer to this broad range of problems as *bundling problems* which basically involve Multi-Robot (or Multi-Sensor) tasks; i.e., they require a bundle of resources to be satisfied. We focus on problems which are motivated by resource allocation in either sensor networks or robot systems. Below we survey some of these in relation to *ST-MR-IA*, the MRTA taxonomy and the type of

mechanism proposed (e.g. distributed or centralized).

The mediation algorithm in [101] is an anytime algorithm that enables agents to incrementally reveal their information in order to achieve a common goal. The author describes how to apply this algorithm to a task allocation problem in a multi-sensor intruder tracking scenario, where each sensor might be assigned to one or multiple tasks and where each task might require many sensors. The mechanism assumes the presence of a central mediator agent which first collects all the *proposals* of utility that each sensor is able to provide individually to each task and then after evaluation of these (which could be achieved with any function, linear or non-linear), it tries to find a better allocation value by considering subsets (i.e. coalitions) of sensors of increasing size. The algorithm goes on until a stop signal is received. Clearly the downside of this approach is the assumption of a central mediator agent (which makes the allocation effectively centralized), but also the high communication cost connected to the hill climbing behaviour. In addition this algorithm is not designed for dealing with a set of dynamic tasks.

In [106] the authors developed an alternative proposal-based algorithm for allocating Single-Task sensors to Multi-Sensor tasks, where the utility of each sensor for a particular task sums up linearly. Compared to the algorithm in [101], the authors in this paper allow for multiple mediation agents (which they call mission leaders), making the algorithm more decentralized. Each mission leader receives the proposals of different sensors in terms of utility contributions and then selects the best set of proposals, therefore deciding a coalition of sensors able to support a particular task (which they call mission). The authors study also a distributed proposal algorithm for a dynamic environment, which avoids performing multiple rounds in order to arrive at a decision to lower the communication cost. Similarly to the approach that we adopt in Chapter 4 and 5, sensors can be *preempted* from ongoing tasks if they can better serve another task with higher profit. Although, in this work the authors do not consider explicitly the problem of matching heterogeneous sensors to different types of tasks consider-

ing qualitative measurements (e.g. semantic-based), similarly to our assumption in Chapter 3. In the same way, they consider that the utilities of sensors contributing to the same task sum up linearly, not allowing for non-linear utility functions.

As noted in [47], distributed agent-based bundling mechanisms deliver a greater portion of bidder requirements (i.e. task requests, in our scenario) to be satisfied and consistent runtimes, but potentially less optimal bundle solutions compared to centralized systems. In [47] the authors empirically compared the centralized CASS algorithm [38], which we also used in Chapter 3, with two market-based distributed mechanisms (Multiple Distributed Auction and Quote Driven Market). Both of these distributed algorithms are developed for a static setting with objective functions different from ours, in addition these do not adapt well to our dynamic setting. Interestingly, they observe that utility values in a bundling mechanism should include both *qualitative* and *quantitative* measurements about the degree at which an agent is able to perform a task. In Chapter 5, we follow this direction by considering both knowledge-based matching for sensor and task types (qualitative), and numerical utility functions considering physical attributes of sensor and task instances in the field (quantitative). Instead in Chapter 3 we take the assumption that utility values could be simply set to zero in the case a sensor cannot serve the task. Note that, in Chapter 3, similar to [47] we use combinatorial auctions as one of the ways to solve our static MSTA problem instance. A *combinatorial auction* is a silent auction in which bidders (tasks) can express preferences on bundles or *combinations* of items (sensors) [1]. Given a fixed supply of goods, the goal of the winner determination problem [103] is to maximize revenue earned from the sale of disjoint item combinations.

[78] proposes distributed algorithms for allocating resource bundles to bidders finding a solution to what they call the distributed winner determination problem. They present a set of distributed search-based algorithms for solving this problem, eliminating the need for a centralized auctioneer, differently from [47]. This work relates to ours, especially to Chapter 5, as it proposes a variation of the protocol for coalition formation in

[115], which they call individual hill climbing algorithm. The issue with this algorithm is that it considers a set of tasks in a static setting, which we address in Chapter 5 by extending the algorithm in [115] to our dynamic scenario.

The problem in [100] is probably the closest to our dynamic MSTA problem instance. In this paper, the authors propose an algorithm for coalition formation with spatial and temporal constraints motivated by a disaster scenario where a team of both responders and robots must undertake a number of tasks. Similar to the problem and distributed mechanisms described in Chapter 5, they consider a dynamic scenario where coalitions can be formed, disbanded and reformed. They also consider *deadlines* associated to each task (by when they need to be completed) and their aim is to maximize the number of tasks completed over time. The authors propose an anytime centralized heuristic that produces a schedule of mobile agents, essentially addressing a variant of the *ST-MS-TE* problem instance. The heuristic achieves a *time-extended* (TE) allocation utilizing information about tasks and agents (e.g. location, travel time and task duration) to plan for future allocations, differently from our *instantaneous allocation* (IA) approach.

Within agent-based approaches, the problem of decentralized coordination of groups of agents (i.e. bundles) is often modelled as a multi-agent distributed constraint optimization problem (DCOP). In this problem a set of agents control the value of variables in a system and they collaboratively aim at optimizing the global reward. The goal is to find the assignment of such set of variables that optimizes the aggregation of payoffs (or conversely costs) over a set of constraints defined on the values of the variables [102]. This is also our aim in Chapter 5 where we propose a distributed approximate solution to the dynamic MSTA model formalized in Chapter 4. The DCOP formalization has recently received a sizeable amount of attention from the multi-agent community and different authors have proposed both *complete algorithms* which always find an optimal solution, such as ADOPT [76] and DPOP [90], and *approximate algorithms*, such as DSA [34] or the Max-sum algorithm [102]. Given that DCOP is in general NP-hard, as for the static and dynamic MSTA problems in this disserta-

tion, complete (or exact) algorithms require an exponentially increasing coordination overhead in terms of messages exchanged and/or convergence time [33]. For example, ADOPT and DPOP arrange the constraint graph into a Depth First Search (DFS), and then exchange messages over this tree. The number of messages exchanged by these algorithms grows exponentially with the height of the DSF tree. Improvements on such algorithms have been proposed to reduce computational time and communication overhead. For example, BnB-ADOPT [131] uses a branch and bound depth first strategy instead of a best search, which improves the computational time of the ADOPT algorithm; MB-DPOP [91] instead improves the DPOP algorithm by offering a linear overhead on the number of message exchanged with polynomial message size. Also considering such improvements, complete mechanisms are still in general too computationally expensive to consider for our scenario, where sensors have limited computational resources, limited energy and constrained communication capabilities.

DCOP approximation algorithms require a less expensive local computation and keep the number of messages exchanged low, therefore being more applicable to our domain. In particular, in this dissertation we also propose approximate centralized and distributed algorithms which sacrifice the optimality of the solution in favour of computational and communication efficiency. An example of DCOP approximate algorithms is provided by the Distributed Stochastic Algorithm (DSA), where the agents randomly change the value of the variables which they control in the system. DSA adopts a local hill climbing optimization strategy, which is implemented by agents varying the value of variables based on a stochastic distribution. This algorithm was designed for achieving decentralized coordination in sensor networks and therefore is more apt for use in our environments (although considering different constraints and problem settings). A drawback of DSA and other similar stochastic based algorithms for DCOP is that in general the approximate solutions found have scarce quality and potentially long convergence time (given their probabilistic behaviour). More applicable to our domain, providing solutions closer to those of DPOP than DSA and with a lower communication overhead is the distributed approximate algorithm called Max-sum algorithm.

First presented in [33] and then refined in [102], the Max-sum algorithm is a message passing algorithm which relies on the generalized distributive law (GDL) [5]. In their work, the authors propose a novel representation of the DCOP problem as a cyclic bipartite factor graph, composed of variable and function nodes which represents agents and utilities respectively. Such representation allows them to use the GDL to break down the objective function into a set of factors which can then be calculated and optimized locally by each agent through communication with their neighbours. The drawback of this approach is that the convergence time varies with the structure of the problem, in particular with the type of utility function used. For this reason in Chapter 5 we decided to adapt the coalition formation algorithm in [115] rather than the Max-sum algorithm as although providing a greedier approach which is likely to output lower quality solutions, our extension of [115] is ensured to converge in a finite number of steps independently from the shape of the utility function. Although, considering the latest refinements of the Max-sum algorithm [102] and its recent applications to MSTA problems, e.g. in [29] for assigning UAVs to live aerial imagery collection, the Max-sum algorithm offers a very promising direction for developing more efficient techniques for MSTA problems.

In multi-robot systems the problem of forming coalitions of robots to serve different tasks has been addressed both with *emergent coordination* and *intentional cooperation* mechanisms. An example of the first approach is provided by [69], where the focus is to achieve task allocation with limited or no explicit communication among robots. In the proposed mechanism, which considers a set of homogeneous robots with the same capabilities, each robot decides which task is better to join based on observations about which tasks are being served and which other robots in their surroundings are currently performing each task. The same authors note that the downside of emergent approaches is that they can be difficult to design and usually provide sub-optimal solutions, although allowing for low communication overhead. The second approach assumes instead *intentional cooperation* among robots [84]. Here robots cooperate explicitly to achieve the common goal of maximizing the total utility of a multi-robot

system, explicitly communicating to each other information about tasks. An example of a dynamic mechanism for task allocation for the case of intentional cooperation is provided by all the bundling mechanisms discussed above (except [69]). Additionally, we can consider [120] which focuses on the MRTA problem instance identified by *ST-MR-TE*. Differently from the problem instance in this dissertation, [120] focuses on time-extended assignment which involves task scheduling and planning. Note that *intentional cooperation* is also assumed in this thesis, and it is particularly evident in the distributed mechanism proposed in Chapter 5 where sensors communicate to each other information about tasks. On a side note, in a sensor network environment it is difficult for a sensor to "observe" which tasks are being carried on by other sensors and take decisions only based on that. In fact, sensing tasks do not usually have a tangible impact on real world differently from robot tasks which include an actuating component, therefore making it difficult to use an emergent approach.

2.2 Sensor Selection

MSTA is related to the general problem of *sensor selection* in which we need to choose a set of "active" sensors to provide accurate information about a sensing field for an extended period of time. The best performance is ideally achieved by collecting measurements from as many sensors as possible. Although, as noted in the introduction, many sensors are heavily constrained by battery limitations and, in addition, are required to remain active for a prolonged operational time. The aim is therefore to keep minimum the number of selected sensors while maximizing the utility of the data collected by the network. This problem has received considerable attention in these last years by the wireless sensor network (WSN) research community, with alternate bursts of interest depending on the applications of such technology and advancements in sensor hardware.

The problem of sensor selection is defined in [104] as follows: Given a set of sensors

$S = \{S_1, \ldots, S_n\}$, we need to determine the "*best*" subset S' of k sensors to satisfy the requirements of one or multiple tasks. The "*best*" subset is defined as the one which provides a certain required quality of information to a task, while meeting the energy constraints of the sensors. The aim is therefore to collect information of high quality and, at the same time, to lower the cost of operating the sensors. This tradeoff is usually modelled with the concepts of *utility* and *cost*, similar to MRTA. The first (utility) represents the quality (or accuracy) of the gathered information and its usefulness to a task, while the second (cost) usually consists of the energy spent activating and operating the sensors (often directly proportional to the number of selected sensors k).

The aim is to select the best subset S' such that the total utility of sensors is maximized, usually while keeping the overall cost under a certain budget (e.g. a total energy which is considered acceptable for the network to consume). For many utility models, this problem is at least as hard as the Knapsack problem as noted in [52], which is NP-complete. This means that there is no polynomial-time algorithm, and for this reason realistic restrictions of the problem have received attention. For example, [52] has considered utilities with geometric structure (e.g. based on the distance of the sensors), and assumed that the cost of the selection is either zero or infinity.

This "packing" aspect is indeed highlighted in the sensor selection work, compared to MRTA, and the energy conservation issue is considered of central importance. For this reason, we model both our MSTA problem instances considering packing constraints. In particular, for MSTA in static environment, in Chapter 3, we consider budget constraints for each task; while instead in the dynamic environment in Chapter 5, we evaluate the traffic generated by our distributed approach and we consider it as an approximate measure of battery consumption, provided that the most expensive operation for sensors is to send/receive messages.

In sensor selection tasks can be general and related to the function of the network such as monitoring the whole field by ensuring complete coverage, or more specific and application-oriented, such as tracking and target's movement. At a given time, the

system might be required to do multiple tasks such as monitoring an event and, at the same time, track a single or multiple moving objects. In general sensor selection has been used in previous work to achieve three major purposes, as noted by [104]:

1. *Coverage schemes:* where sensor selection is used to ensure coverage of areas (or targets) of interest.

2. *Target tracking and localization schemes:* where sensors are selected for achieving the best performance in target tracking and localization.

3. *Application-level task assignment schemes:* where sensors are selected for achieving *single* or *multiple* application-level tasks, i.e. not at a basic functionality level like the previous two categories (e.g. identify a target).

Below we briefly survey these three *sensor selection* categories and we highlight differences and similarities with our work. We show that the MSTA problems considered in this thesis differ from these as we consider multiple tasks *competing* for the exclusive usage of the same sensing resources, while instead in sensor selection the focus is usually on supporting single tasks. Although in some cases multiple tasks have been addressed, these often do not explicitly compete for exclusive usage of sensing resources, or alternatively use additive functions to evaluate the utilities of sensors contributing to the same task.

2.2.1 Coverage Schemes

Coverage schemes have probably received the largest attention in the sensor selection literature. They aim at selecting sensors to ensure complete coverage of the field, where every point in the field must be covered by at least one sensor. The works surveyed here can be divided into sensor selection for *static sensors* and for *mobile sensors*.

In [17, 74, 89, 116] the authors focus on the coverage problem using *static sensors*, with the goal of selecting the least number of sensors and conserving overall the largest

energy quantity to monitor the field. Specifically, [89] divides the sensors into sets, such that each set is capable of providing complete coverage of the field and only one set is active at a time. Their objective is to obtain an optimal scheduling of these sets such that the lifetime of the sensor network is maximized and a minimum level of quality of service is achieved. In [17] sensors are divided into disjoint sets, so that every set completely covers all the targets and only one set is active at a time. Differently from [89], the authors schedule the sets in a round-robin order and the focus is to find the maximum number of disjoint sets, which they prove to be NP-complete. Both [89] and [17] focus on centralized schemes. Differently, [116] proposes to achieve full coverage with a minimum number of sensors by identifying redundant sensors and turning them off, in a distributed fashion. Voronoi diagrams [8] are used to identify such sensors. The proposed algorithms plan for future allocations by considering the residual energy of sensors to balance the energy consumption and extend the network lifetime. Another distributed protocol is given in [74], where the authors aim to provide k-coverage, that is every location on the field is covered by at least k sensors. This improves the fault-tolerance of the network and often also the quality of the final sensor readings (e.g. through sensor data fusion). Their goal is to determine a schedule which activates the least number of sensors.

Differently from the work surveyed above, [113, 124] assume that sensors are *mobile*; i.e., they can be moved in order to cover a particular area. In [113] the authors propose different schemes (which could be executed distributely) for Dynamic Coverage Manteinance (DCM), i.e. schemes for deciding which sensors should be moved in order to fill a hole in the coverage. The heuristics used to achieve this vary from selecting the sensor with maximum residual energy, to minimizing its travel distance and combinations of both. Similarly, in [124] the authors propose a bidding protocol for deciding which sensors should be moved to cover holes in the coverage. They consider a mixed network of mobile and static sensors, where mobile sensors are initially redundant. When static sensors discover holes in the coverage of the field (using Voronoi diagrams), they bid for mobile sensors based on the size of the detected hole, and

also on the size of the hole that the sensor will leave after having moved. The mobile sensors choose the bids which minimize the size of the hole left uncovered, and maximize the size of the holes they could cover.

2.2.2 Target Tracking and Localization Schemes

In this section we present a brief overview of selection schemes for the purpose of target tracking and localization. Similar to [104], we categorize some of these works with the type of solution proposed. In particular, we provide an example for each of the following categories, (i) Entropy-based solutions, where the selection schemes aim to minimize the entropy of measurement, (iii) Dynamic information-driven solution, where the aim is to maximize the information gain based on the dynamic information gathered, and (iii) Mean Squared Error-based solutions, where the aim is to minimize the mean squared error of measurement. We denote that the common feature among these works is that they usually focus on single target tracking or localizing problems. Thus basically the number of tasks (i.e. track or localize) is essentially equal to 1, and therefore there is no competition for resources among different tasks.

In [125], the authors consider an entropy-based scheme for sensor selection to achieve target localization. *Entropy* is a measure of uncertainty, and a low entropy entails a high accuracy in the target localization. The authors propose a heuristic which aims at minimizing the entropy. The centralized scheme proposed, greedily selects one sensor in each step, without retrieving any sensor observation. Given the a prior probability of the target location and the location and sensing model of candidate sensors, the heuristic selects the most informative sensor such that the collection of previously gathered measurements of the selected sensors and the prior target location distribution would yield the lowest entropy in the target location distribution.

In [133], the authors propose a dynamic information-driven scheme to select sensors for target tracking. Their aim is to select at each step of the tracking the sensor which

provides the best improvement on the estimate of a target location at the lowest cost. The improvement on the estimate is formalized as *information gain* defined based on distance measure. The proposed protocol starts by selecting a single sensor (leader), considering the predicted starting position of the target. After having collected information about the target, the sensor then elects a new leader among the sensor candidates ranked using the concept of information gain.

Finally, in [59] the author proposes a sensor selection scheme with the aim of minimizing the *mean squared error* (MSE) on the target estimated location. An assumption here is that each sensor can estimate the direction of arrival (DOA) of the target by using acoustic properties. The centralized sensor selection scheme proposed here, starts by selecting (and activating) two sensors which are not collinear with the target. This initial set of two sensors is chosen by using global information about the locations of all the sensors on the field, and by evaluating all the possible pairs of these (i.e. of complexity n^2). Sensors are then added one at a time using a greedy approach, with the goal of minimizing the MSE of the target position. This work, similarly to the two previously considered, focuses on a single target localization task, and therefore does not take into account any competition among different tasks.

2.2.3 Application-level Task Assignment Schemes

In this section we present a brief overview of selection schemes for achieving both *single* and *multiple tasks* which have to be supported collectively. Differently from the sensor selection schemes covered before, we discuss works which consider tasks at an "application-level", and not at a basic functionality level such as coverage or single-target tracking. In this more general framework, approaches discussed usually propose more generic measurements to drive allocation such as a "utility" value. This is more similar to the MRTA and MSTA problems, which consider a generic utility to drive the allocation of robots/sensors to tasks. Although some of these works address the issue of tasks *competing* for resources (i.e. Single-Task sensors), our static and

dynamic scenarios have still not been addressed. For the static setting, tasks with both a threshold on utility demand and budget constraints have still to be considered; while in the dynamic setting, addressing tasks and sensors with different capabilities (i.e. heterogeneous sensors and tasks) and allowing resource re-allocation still require exploration.

Single Task Assignment

In [15] the authors consider the problem of selecting sensors with the aim of satisfying a single generic application-level task. They provide as an example the general task of aggregating sensory information from multiple sources and more in particular they point at "beamforming" algorithms which combine a set of acoustic signals into a single signal without loss of relevant information. They focus on satisfying a single task repeatedly over time in the most efficient manner. The aim in this case is to select sensors using the general notion of "usefulness" which is quantified through a utility value. In this work, the authors propose a scheme to select sensors based on a utility function and a cost model for energy consumption (due to sensing or data dissemination). The utility of a set of sensors is computed based on the sensor locations and on the cardinality of the sensor set.

Another example of application-specific single task assignment which uses utility-based metrics is discussed in [12]. Also here the authors focus on satisfying a single task repeatedly over time, but differently from [15] they provide a more general framework in which the application can specify the utility values of the sensors. They argue, in fact, that which sensor collected data is useful depends heavily on the specific application aim. For example, in some cases reading from a group of sensors may be considered redundant while in others it could be considered as necessary (e.g. in the beamforming algorithms). The objective in their schemes is to select a sequence of sensor sets such that the total utility overtime is maximized, while not exceeding the available energy.

Finally, in [2] the authors focus on the single application-level task of monitoring the entire field by selecting a subset of the sensors using the concept of relationship between sensor data. They, in fact, highlight that sensor readings are often related, for example in the case of redundant sensors covering the same area or that are close to each other. In many cases, these relationships among sensors can be predicted and used to select the sensors to approximate the stream of others. The idea is based on the concept of information networks, through which each sensor is seen as a data object and the relationships among them modelled as logical links. The authors propose an integer programming formulation and a greedy approximation algorithm to select the best subset of sensors with the objective to minimize the error in the sensor data collected by using such sensors rather than all of them.

Multiple Task Assignment

Below we give some examples of multiple-task assignment schemes, in which different tasks need to be accomplished simultaneously. The presence of multiple tasks often entails competition for sensing resources and, since resources are usually scarce (e.g. in terms of quantity of sensors or energy constraints), completion of one task means that others will not be accomplished. Note that we always refer to specific application-level tasks, and that these are actually MSTA problems.

In [54], the authors aim at providing multiple target coverage configuring cameras. They formally introduce the pan and scan problem where each camera can observe multiple targets. They consider a set of static cameras (fixed positioned), where for each one they are allowed to choose its orientation and its zoom factor or field of view (in two dimensions). The aim is to maximize the quality of target's reading by the set of cameras. The sensors here are directional and covering each target can be viewed as a different task. They consider two variants of the problem: an easier one in which each target accumulates measurement qualities additively from all cameras observing it, and a more difficult one where each target obtains only the best measurement. They

propose an optimal and an approximation algorithm for a geometrically constrained and an unconstrained version of the problem.

Similarly to [54] in [4] the authors study a more restricted version of a similar problem where, always given a set of deployed directional cameras, the aim changes to maximize the number of targets covered with the minimum number of active sensors (while instead in [54] the objective is to maximize the quality of the sensor readings). In both [4] and [54], sensors are directional and therefore covering each target can be viewed as a different task. Note that covering one task might mean leaving unmonitored another one, and from this point of view they are multiple task assignment schemes which are clearly instances of the MSTA problem.

In [77] the multiple-task assignment problem is modelled as a market, and the authors discuss the advantages of incorporating e-commerce concepts to sensor management. The basic intuition here is to introduce in sensor management the competition aspect typical of markets, where a customer usually competes with others for buying "goods". The authors argue that so far sensor management have been approached as a relatively uncompetitive problem. The model proposed is composed of two main components: the task manager and the sensor manager, which are implemented using genetic algorithms (GA). The task manager allocates budgets to the application (consumer), based on the different tasks involved in the application and their requirements. Using these budget values, the consumer places bids to the sensor manager. Based on these bids, the sensor manager allocates sensors to tasks. The scheme proposed here is a centralized system and it is not trivial to implement in a distributed way because of the complexity of the model and the GA computation involved.

In [106], the authors consider the problem of assigning directional sensors to multiple generic tasks (which they call missions) using the concept of profit as a function of the utility that each sensor can provide, therefore identifiable as a utility-based selection scheme. They call this problem Semi-Matching with Demands (SMD), and they model it as a bipartite semi-matching problem. More specifically, each task is associated with

a demand value and a profit value; each sensor-task pair is associated with a utility offer. The utility of sensors allocated to the same task adds up linearly. This is similar to our assumption in Chapter 3, although this problem differs from the one we analyze in the static settings because here tasks are not given a particular budget. The goal is to decide which sensors should be assigned to which task in order to maximize the profit of the satisfied tasks. They prove that this problem is NP-hard and investigate other two versions of SMD. One where the number of sensors which can contribute to a task is bounded (δ-SMD), and another where sensors and tasks are deployed on a 2D field (GeoSMD). They also introduce a generalization of the problem using a threshold (Threshold-SMD), where they assume that the utility provided by sensors to a task needs to be greater than a threshold on the task demand for it to be successful, similar to what we do in Chapter 3. They propose solutions to these problems in both static and dynamic settings, always considering additive sensor utilities. In particular, for the static setting, they discuss a centralized greedy algorithm for δ-SMD and a polynomial-time approximation scheme (PTAS) for GeoSMD. They then propose a distributed solution which consists in a novel multi-round proposal algorithm, for both static and dynamic settings. They also provide an energy-aware extension to the distributed algorithm to prolong the network lifetime, which we could classify as time-extended assignment. These algorithms performed near-optimally in their experiments for both the static and dynamic settings. Furthermore, they prove that distributed algorithms are competitive with the centralized algorithms, especially in networks with high node density. Clearly, this work is the one that is the closest to this thesis, and we will note in the next chapters that the models presented for both static and dynamic settings are extensions of the SMD model, and therefore still NP-hard.

In our previous work [94], we also addressed assignment of directional sensors to multiple tasks (which we also used to call missions). In this paper, we characterized tasks with *uncertain demands* for sensing resource capabilities, differently from [106]. We modelled this assignment problem by introducing the Sensor Utility Maximization (SUM) model, where each sensor-task pair is associated with a utility offer and each

task with a profit and an uncertain demand of utility. Utilities of sensors contributing to the same task adds up linearly. The goal is to find a sensor assignment that maximizes the total profit, while ensuring that the total utility cumulated by each task does not exceed its uncertain demand. SUM is a special case of the well known Generalized Assignment Problem (GAP) [22].

Finally, in [82], multi-modal sensors (radars) are used and multiple areas need to be monitored simultaneously. The authors create a bid for each task considering the possible sensor configurations (which sensor operates in which mode) and the task utility value. They then solve the winner determination problem using a modified combinatorial auction algorithm and thus a sensor is allocated to the task, and an operation mode is selected for the sensor. Note that we can also consider the bidding scheme presented in [124] as a multiple-task assignment scheme, if we consider that each hole to be covered represents a task. In fact, in [124], the task competition aspect is included in the objective function of the problem which considers also the task (or areas to be covered, in this case) which will be left unsatisfied (i.e. uncovered) when the sensor moves to the new hole to be covered.

2.3 Multi Sensor Task Allocation & Taxonomy

The Multi-Sensor Task Allocation problem can be seen as a subclass of the sensor selection schemes previously surveyed. It, in fact, focuses on *multiple-task assignment schemes* where tasks are usually considered at an application-level. Similarly to MRTA [40], we assume that each task can be either discrete (e.g. localize vehicle) or continuous (e.g. monitor an area), and that it is independent from any other task. *Task independence* is a strong assumption, as in MRTA, we in fact require that each task can be considered and assigned resources independently from the others. By imposing this limitation, we can avoid to consider at the allocation stage the dependencies between tasks which generally increase the problem difficulty. This would oblige us

to impose additional constraints on the problem instance which in general might not ensure a good solution quality and a sufficient flexibility/scalability of the system [40]. In principle, as also noted in the introduction, each set of tasks with sequential or parallel constraints could be merged and presented as input to the problem as a single monolithic task, although this shifts the difficulty from the problem of reasoning about task constraints to the problem of reasoning about aggregated task requirements and estimating a joint utility value.

As in [40], we assume that the objective of the MSTA problem is in general to maximize the "usefulness" of sensors to tasks by choosing which sensors should execute which tasks. From this perspective MSTA considers a subset of MRTA problems in *sensor network environments*. In the MRTA taxonomy [40] the distinction between a MRS composed by homogeneous devices versus one composed by heterogeneous devices does not represent one of the main axes. This distinction is instead relevant when considering solutions for the sensor selection problem. For example, in [77] the authors suggest that the heterogeneity could be used to improve the flexibility of the sensor selection, by choosing alternative resources in case the ones requested are unavailable. A similar argument is given in [130], although the focus is on link and energy heterogeneity in the sensors, rather then heterogeneity in the type of sensing capabilities provided. In Section 2.3.2, we highlight the need of a formal representation of task requirements and sensor capabilities when we are in the presence of a heterogeneous sensor network supporting different types of sensing tasks. In Section 2.3.3, we discuss different architectures proposed for dealing with heterogeneous resource allocation. Following (Section 2.3.1), we introduce the MSTA taxonomy which provides a new perspective inspired by the MRTA problem on sensor selection problems.

2.3.1 MSTA Taxonomy

Given the differences highlighted between MSTA and MRTA, we are aware that these two general problems still remain strongly connected. Nonetheless, our aim is to in-

crease the awareness of the MRTA framework advertising it to the Wireless Sensor Network research community, by referring to a particular subset of the sensor selection problems as MSTA. In particular, provided that the WSN community has started to address both homogeneous and heterogeneous environments, we propose a preliminary MSTA taxonomy as an extension of the MRTA taxonomy with a new axis which addresses this distinction. MSTA is therefore an alternative way of looking at the literature in the sensor selection field, adopting the same point of view taken in MRTA by Gerkey et al [40]. In addition, this hopefully could help to identify new opportunities for problem instances which have not yet been explored in the sensor selection field.

Looking at the papers surveyed above proposing sensor selection schemes, it seems almost taken for granted that matching the capabilities required by tasks and provided by sensors is straightforward. This is due to the focus on cheap wireless sensors with low processing power, limited battery and sensing capabilities. As highlighted in the introduction, this is now changing especially in military and emergency response operations, which increasingly make use of heterogeneous sensing platforms able to mount multiple different sensors, providing aggregated complex sensing capabilities and a wide variety of computational power and autonomy levels throughout the network. From a certain point of view MRSs, and in particular MRTA problems, include already this aspect, because of the assumed heterogeneity of robots in the environment and their wide range of capabilities.

In order to provide an alternative framework to look at sensor selection works, we suggest therefore to extend the MRTA taxonomy as follows and to adopt the name MSTA for the particular subset of sensor selection focussing on application-level multiple task assignment schemes. Moreover, we consider also problems related to matching heterogeneous sensor capabilities to different task types. We basically extend the MRTA taxonomy with an additional dimension through which we can classify MSTA problems also based on the type of sensor network considered. The MSTA taxonomy is therefore identified by the following four axes:

- **Single-task sensors (ST)** vs. **multi-task sensors (MT)**
- **Single-sensor tasks (SS)** vs. **multi-sensor tasks (MS)**
- **Instantaneous assignment (IA)** vs. **time-extended assignment (TE)**
- **Homogeneous sensor network (HO)** vs **heterogeneous sensor network (HE)**: HO means that the network is composed only by sensors with the same sensing capabilities and usually provides support to a set of tasks of the same type (e.g. all monitoring tasks). HE represents the case in which the sensor network consists of sensing devices of different types each with different sensing capabilities, often providing support to different task types (e.g. both monitoring and target identification tasks). Note that, in the second case, we need an either an *implicit* or *explicit* representation of the sensor capabilities and task requirements in order to find a matching among those.

To categorize MSTA problem instances by using this taxonomy, we need to identify the corresponding quadruple of two-letter abbreviations. In this thesis we study the *ST-MS-IA-HE* problem in both static and dynamic settings. As previously highlighted, in our work we address first a simple quantitative based framework to explore initial matching schemes for a static problem without explicitly considering the problem of fitting the capabilities of sensors to task requirements, and simply setting the utility to zero if resource types are not compatible with a task. We then explicitly introduce a semantic-based component which shows how we could integrate sensor-task capability fitting in the problem of allocating heterogeneous sensor resources to different types of tasks in a dynamic setting. We note, in fact, that usually a heterogeneous environment implies some kind of representation of the capabilities provided by sensors and required by tasks.

It is important to note that utility is a very flexible measure of fitness that, in particular in heterogeneous environments, should include both quantitative measurements about the sensor performance on the field (based on location, signal quality, etc.), but also

qualitative measurements such as the fitness of a particular sensor type for a particular task type, as also highlighted by [47]. The only constraint on utility estimators is that they must produce a single scalar value that can be compared for the purpose of ordering sensors (or group of sensor canditates) for the task, as noted in [40].

2.3.2 Homogeneous vs Heterogeneous Sensor Networks

Following we provide a brief survey of sensor selection schemes which we divide into schemes for homogeneous (HO) and heterogeneous sensor networks (HE). Our aim is to highlight that the works in HE adopts either an *implicit* or *explicit* representation of sensing capabilities and task requirements, differently from HO which do not consider this problem. An *implicit* representation means, for example, that we can simply set utility values to zero when the sensor types are not compatible with a task. An *explicit* representation instead is for example achieved using formal tools to represent knowledge, such as ontologies.

For the case of *heterogeneous sensor networks*, examples are provided by [77, 110, 126]. In [77], previously surveyed, the authors assume that each user could create a task with different requirements for resources such as a "search for target" or "monitor person in danger". They therefore consider the problem of matching different types of sensors to various task types. In order to match different task requirements with different sensing capabilities, in the task manager component of their architecture they maintain a knowledge base which contains the type of resources required for a particular task, together with the utility that each of these resources can provide to the task. The knowledge base is represented using a simple description language which *explicitly* describes the requirement of each task in terms of plan, and the capabilities provided by sensors.

The same process is described in more detail in a paper by the same authors in [126], where they extend the sensor selection problem to a more general scenario in which

any type of task (not only sensing task) competes for the exclusive usage of scarce resources. In Figure 2.1, we show an example of how the authors *explicitly* describe a task for an emergency response scenario in which resources are represented also by vehicles and human operators. They propose a multi-agent approach in which each agent contains a knowledge base of the task required to accomplish a general mission and the resources required by each task (as in Figure 2.1). Differently from [77], the authors propose a combinatorial auction approach in which each agent bids for a *bundle* of resources *compatible* with the task, then the winner determination problem is solved by an auctioneer agent running the CABOB algorithm [108] in a centralized fashion.

```
(plan deliver_to(?dest ?obj)
     (res-requirement (helicopter 2)(pilot
2)
      alternatives (1 (truck 3)(driver 3))
     )
     (termcondition (current_loc ?dest =))
     (utility 500)
     (process
            (seq
               (load ?obj)
               (move_to ?dest)
               (unload ?obj)
             )
     )
 )
```

Figure 2.1: Example of a task resource requirements used in [77, 126].

Finally, [110] proposes an allocation scheme for heterogeneous tasks to heterogeneous team members. They focus on providing support in emergency response scenarios with "extreme teams", including mobile sensors but also human operators. Each team member has a capability to perform a particular task, which they represent as a numerical value in the range $[0, 1]$. If this value is greater than 0 this means that the team member has enough resources to perform the task and they assume that in extreme teams this is true for a really large number of tasks. They model this problem extending the generalized assignment problem and they propose a distributed constraint optimization (DCOP) algorithm to maximize the team reward. If we constrain the team members only to sensing devices, this problem considers *heterogeneous sensor*

networks. Although the difference of this work with [77, 126], is that they do not consider explicitly the issue of matching different capabilities with different requirements. Their assumption is that extreme teams could achieve any task type, and therefore they use an *implicit representation* of sensor capabilities which simply sets the utility value to 0 when the resources are not compatible with the task, similar to our approach in Chapter 3.

For the case of *homogeneous sensor networks*, examples are provided by [24, 54, 82]. In particular, in [24] the authors study the problem of forming coalitions of mobile autonomous sensors to optimize the total utility of the ad-hoc network formed by these mobile sensing entities while performing exclusively wide area surveillance tasks. The authors focus on overlapping coalitions in which sensors can sometimes perform multiple tasks. They assume that all the sensors in the network are able to satisfy wide area surveillance tasks with different degree of utility, and for this reason they do not consider the problem of representing different sensor capabilities with task requirements. In [54], the focus is on selecting camera sensors to monitor multiple targets, and [82] focuses only on sensor networks composed by radar arrays accomplishing exclusively weather monitoring tasks, thus again not needing to represent different capabilities and requirements.

2.3.3 Architectures for Heterogeneous Resource Allocation

Different modular approaches and architectures have been proposed for allocating heterogeneous resource bundles to different tasks. In this section we present how many of these split the allocation process in two parts, by first considering the type(s) of resources required (i.e. qualitative), and then the actual resource instances matching the type(s) required (i.e. quantitative). We highlight similarities and differences with the architecture in Chapter 5, which in general adopts the same approach to divide the allocation process in qualitative-based matching, through explicit representation of sensing capabilties and task requirements, followed by quantitative-based matching

using the features of sensor instances (e.g. distance or energy remained).

In [81], the authors propose an agent-based architecture for dynamic formation of *virtual organizations*, which are composed of a number of individuals, departments or organizations each having a range of capabilities and resources at their disposal. This problem is related to our dynamic MSTA problem in heterogeneous environments, where also each sensor has a different range of capabilities. The mechanism delegates each task request to a Requirements Agent (RA), similar to the mission leader concept in [106], that then has the role to form the virtual organization by choosing the best Service Providers (SP), i.e. sensors in our case. The RA uses a Yellow Pages (YP) agent to match the *type of service* provided by each SP, e.g. movies or text messaging; it then sends a request for *proposals* to this subset of SPs, which then answer with the actual cost of the service provided (e.g. $\pounds 25$ per month). The SPs could decide also to partner together and send a *joint proposal*. The RA then decides which one is the best bundle of SPs to accept. This is similar to the problem and conceptual architecture proposed in Chapter 5, in fact we also use a knowledge-based component in order to find which type of sensor can potentially match the requirements of a certain task type. Differently although, given our distributed sensor network environment, we do not assume a central yellow pages agent like the authors do, and instead deploy this component in each of the users' mobile devices. In the already cited [77], the architecture proposed is essentially centralized and considers also a "yellow pages" central component similar to [81].

Closer to our approach, [32] proposes an ontology-based mechanisms for coalition formation in heterogeneous multi-robot systems. Their aim is to solve the *MT-MR-IA* problem for a multi-robot system for the USAR (Urban Search and Rescue) scenario [64]. The scheme they propose is similar to our conceptual architecture, and it is composed of three main components: an ontology-based reasoner, a search for "useful coalitions" and a coalition instantiation algorithm. The first is implemented using the Pellet reasoner which uses the task and robot ontology in [19] to find which kinds of

robots are able to accomplish a certain task type. Then, given its recommendations, the mechanism searches for useful coalitions of robots which could potentially satisfy each tasks requirements. Finally, the coalition instantiation algorithm finds a feasible allocation of robot coalitions to tasks, extending the algorithm in [85]. Although this algorithm could be implemented in a distributed fashion, the authors in [32] propose an essentially centralized architecture. In addition, they do not explore the dynamic setting considering, instead, a static set of tasks on the field. Their approach confirms that in heterogeneous environments it is convenient to split the allocation process in at least two steps: one based on capability matching followed by one based on physical features of the robots in the field.

Similarly to [32], in one of our previous works [44], we investigated how to integrate an ontology-based reasoning component with an allocation algorithm supporting sensing tasks in a heterogeneous sensor network. We described a preliminary version of the architecture proposed in Chapter 5, discussing how the components might be integrated and work together. As in the different architectures surveyed above, we distinguish mainly two operations: first, sensor-task fitting based on capabilities provided and required, second, the actual allocation of sensor instances to tasks. Our distributed architecture extends the ontology-based reasoner with a bundle generator component which given the recommendation of the reasoner looks for feasible sensor coalitions, this idea is similar to the "search for useful coalitions" step in [32]. Note that differently from our distributed mechanism in Chapter 5, the architecture in [44] does not offer the possibility of using different utility models (e.g. non-additive) to evaluate different sensor bundle utilities, in general considering only individual sensor utility offers to tasks. From this point of view, the architecture discussed in Chapter 5 can be seen as an evolution of the one proposed in [44].

Finally, [114] is closely related to the architecture proposed in this thesis for the dynamic settings. The authors consider both the *ST-MS-IA-HE* and *MT-MS-IA-HE* problems for which they suggest a distributed agent-based approach. They develop an

extension of the semantic reasoner in [44] and integrate that with an extended distributed version of our MRGAP algorithm, presented in Chapter 3. The extended reasoner is able to consider links among different tasks while instead the allocation algorithm considers both budget and threshold constraints being an extension of the MRGAP algorithm. The downside of this allocation system is that it considers only individual sensor utility offers, and combines those additively for sensors contributing to the same task. The distributed mechanism which we propose for the dynamic MSTA problem, instead, allows for both linear and non-linear utilities, although being limited to Single-Task sensors and not considering budget and threshold on demand constraints (which we instead consider only in the static setting in Chapter 3).

2.4 Conclusion

Sensor selection and MRTA problems have received a sizeable amount of attention lately, although they have been considered separately. In this chapter, we proposed to look at a subset of the *sensor selection* work as problems similar to *MRTA*, which we call *Multi-Sensor Task Allocation* problems to highlight the link with MRTA [40]. We proposed to extend the MRTA taxonomy in order to classify different MSTA problem instances based on features of sensors, tasks, assignment and sensor network.

In sensor selection, the issue of handling multiple tasks that might compete for the usage of sensing resources and with different priorities has started to receive increasing attention. We think that this is due to sensor networks now being composed of heterogeneous sensors, able to support a larger number of application-level tasks simultaneously. We believe that going in this direction, sensor selection problems will face the same issues that the Multi-Robot System community has already faced in MRTA, although with a different perspective. Sensors, in fact, can be seen as resource constrained robots and therefore still require more conservative approaches, for example in terms of energy, processing power or bandwidth.

For example, an issue common in sensor selection is that there are few works addressing estimation of sensor utilities taking both quantitative and qualitative metrics into account (similar to [77]). Currently, most sensor selection schemes determine the utility of a sensor by its physical attributes, using numerical utility functions based on sensor location. The need here is to develop utility estimator mechanisms which consider not only physical attributes, but also some sort of semantics. For example, a video sensor which is close to a target might not perform well during night because it cannot provide nightvision. In general, the problem becomes even more complex when considering groups of sensors where we have to consider aggregated sensor capabilities. This has been addressed in some of the MRTA literature [32] but it still needs to be explored in the sensor selection field. In particular, we address this in Chapter 4 and 5 where we propose a distributed architecture which allows to include both numerical utility values and semantic based metrics to evaluate the performance of a group of sensors to a particular task.

The idea of allocating *sensor bundles* to tasks, instead of individually selecting sensors, has not yet received much attention in sensor selection schemes (e.g. [24, 54]). Differently, the work in MRTA has already considered tasks requiring multiple robots and referred to these as *Multi-Robot tasks*; in addition, formal models together with well known bundling techniques were proposed to address these [40]. This relates to the issue of conditional utilities which has been embedded in some information-gain sensor selection schemes [133]. By conditional utilities we mean how would the utility of one sensor for a task change if other sensors are allocated simultaneously to the same task (similar to [104]). For example, if we consider the task of localizing a particular object on a two-dimensional field using directional acoustic sensors, then we will require at least two of them in order to obtain the target location, therefore the utility of allocating just one sensor will be zero while instead allocating two or more might fully satisfy the task requirements. We refer to these as *non-linear utility models* and we explore them in Chapter 4, where we develop sensor allocation schemes in which we allow for evaluation of *joint* utilities rather then considering each *individual* sensor utilities as

in Chapter 3, which are commonly assumed by many other works in sensor selection (e.g. [37, 106]). Finally, in Chapter 5, we apply techniques already proposed in the MRTA field (i.e. multi-agent coalition formation protocols [109, 115]) in the context of allocating groups of sensors with non-linear utilities to tasks.

Another issue in current sensor selection schemes is lack of schemes taking advantage of node re-assignment (like in [106, 124]). Especially in a dynamic environment, with different tasks appearing on the field over time, we need to be able to re-allocate sensors to higher priority tasks or to the task to which they could contribute the most. This entails *pre-empting* resources from ongoing tasks, and therefore might have a negative effect on others. Effects of such node reassignment and its associated cost need to be studied more deeply. In Chapter 5, we study the effects of preemption and we show how this is actually beneficial towards the overall satisfaction of the tasks over time.

Finally, in sensor selection there has been some work in defining frameworks for single task assignment problems defining models using the notions of utility and cost, in which the goal is to find a solution that maximizes the utility while staying under a predefined budget (e.g. [12, 15]). In these works, however, the sensors are working cooperatively on a single large task, such as monitoring toxicity levels in an area, and therefore do not consider multiple tasks in competition for the same resources. In Chapter 3, we address this issue by considering multiple tasks *competing* for the same resources and associated to different utility demands and budgets. In that work, we aim at maximizing the profit of the satisfied tasks while staying under the budget constraints, therefore addressing the lack in the literature of sensor selection schemes considering multiple tasks in competition under demand and budget constraints. We believe instead that budget constraints in a dynamic setting are too restrictive (especially considering instantaneous allocation) and therefore in Chapter 4 and 5 we only consider utility demands for each task.

Chapter 3

Static MSTA with Additive Sensor Utilities & Budget Constraints

In this chapter, we examine the MSTA problem in a *static* setting where all the tasks arrive at once. The static setting is motivated by situations in which different users are granted control over the sensor network at different times. During each time period, the current user may have many simultaneous tasks. While the current user will want to satisfy as many of these as possible, sensing resources may be limited and expensive, both in terms of equipment and operational cost. In some environments, replacing batteries may be difficult, expensive, or dangerous. Furthermore, a sensor operating in active mode (i.e. assigned to a task) may be more visible than a dormant sensor, and so in greater danger of tampering or damage. Therefore, we give each task in the static problem a *budget* so that no single user may overtax the network and deprive future users of resources. This budget serves as a constraint in terms of the amount of resources that can be allocated to a task regardless of profit; in this case, sensors will have associated cost.

As noted in Chaper 1, we focus on *Single-Task* sensors supporting *Multi-Sensor* tasks, in an environment where we do not have enough information to plan for future allocation allowing only for *instantaneous allocation* in *heterogeneous sensor networks* (i.e. *ST-MS-IA-HE* according to the MSTA taxonomy in Chapter 2). Although some sensors, such as seismic sensors, can receive data from their surroundings as a whole, other sensor types, such as cameras, are directional and therefore Single-Task. In these

cases, the direction of each sensor, and thus the task it serves, must be chosen appropriately.

A given sensor may offer different tasks varying amounts of information (because of geometry, obstructions, or utility requirements), or none at all. We assume that the utility of sensors to tasks is *additive*, i.e. utilities of sensors contributing to the same task add up linearly. We use a simple function to evaluate each sensor utility which models the real degradation of the sensing performances with the distance between sensor and task. Tasks, on the other hand, may vary in importance (or *profit*), and amount of resources they require (or *demand*). More specifically, each task is associated with a demand value and a profit value; each *individual sensor-task* pair is associated with a utility offer. In some but not all applications, it may be more beneficial to do one thing well than to do many things badly; that is, to fully satisfy one task rather than give a small amount of utility to several. We model this with a *threshold* on the utility demand which needs to be surpassed in order for the task to be successful. Note that we therefore consider both packing (i.e. budget) and covering (i.e. demand threshold) constraints which make the problem non-trivial to solve.

Contributions. We model the static *ST-MS-IA-HE* problem with additive utilities, demand threshold and budget constraints as a bipartite semi-matching problem. We propose three centralized algorithms to address the problem. We give an efficient greedy algorithm and a multi-round proposal algorithm whose subroutine solves a Generalized Assignment Problem (GAP), as well as an adapted Combinatorial Auction algorithm. Through simulations we find that in dense networks the three algorithms perform well, with the GAP-based algorithm outperforming the greedy and the combinatorial auction based algorithm. We also studied how tasks spend their budgets and how increasing the budget affects the overall profit of the network. We find that in most cases giving tasks higher budgets does not help since what limits the amount of achievable profits is competition and not the amount of budget provided.

The rest of this chapter is organized as follows. Section 3.1 presents some related

work. In Section 3.2 we present our network model and then propose the formal model for *ST-MS-IA-HE* in static settings. In Section 3.3 we propose schemes for this static problem, whose performance we evaluate in Section 3.4. Finally, Section 3.5 concludes the chapter.

3.1 Related Work

Sensor networks. Assignment and selection problems have received significant attention, in both experimental and theoretical communities. The authors of [74, 89, 116], for example, solve the coverage problem, which is related to our static MSTA problem. They try to use the smallest number of sensors in order to conserve energy. The techniques used range from dividing nodes[1] in the network into a number of sets and rotating through them, to activating one set at a time [89], to using Voronoi diagram properties to ensure that the area of a node's region is as close to its sensing area as possible [116]. Sensor selection schemes have also been proposed to efficiently locate and track targets. For example, [133] uses the concept of information gain to select the most informative sensor to track a target. [59] attempts target localization using acoustic sensors. The goal is to minimize the mean squared error of the target location as perceived by the active nodes. In these works, however, sensors are typically being chosen to work together to perform a single large task, such as covering an area or tracking a target, whereas our work here is essentially concerned with competition among many simultaneous missions.

In wireless sensor networks, there has been some work in defining frameworks for single and multiple task assignment problems. For example, [15, 12] defines a framework for modeling the assignment problem by using the notions of utility and cost. The goal is to find a solution that maximizes the utility while staying under a predefined budget. Here again, though, the sensors are working cooperatively on a single large

[1] We use the terms *node* and *sensor* interchangeably.

task, such as monitoring toxicity levels in an area. In [77], a market-based modeling approach is used, with sensors providing information or "goods". In this case, however, the motivation is more about testing auction mechanisms than in optimization algorithms.

Algorithms. The Generalized Assignment Problem (GAP) [22] is a generalization of the Multiple Knapsack Problem, in which the weight and value of an item may vary depending on the bin in which it is placed. There is a classical Fully Polynomial-Time Approximation Scheme (FPTAS) [122] for the core Knapsack problem which performs a dynamic programming (DP) procedure on a discretization of the problem input. If the knapsack budget value is not too large, then the DP can find the optimal solution in polynomial time. A stricter version of the static MSTA problem was formalized as Semi-Matching with Demands (SMD) in [106]. In that formulation, profits are awarded only if a certain utility threshold is met but no budgets are considered. The static problem we study here is a common generalization of this previous problem, incorporating both budgets and a profit threshold.

Although we use the terminology of sensors and tasks for concreteness, the problem we are considering can be viewed as a more general problem of resource allocation. An alternative interpretation regards scheduling jobs on unrelated parallel machines. As in other (maximization) scheduling problems [119], the goal is a schedule that maximizes profit earned from jobs completed, subject to certain constraints. The twist is that each job specifies not the set of *machines* that can perform it, but the set of *families of machines* that can perform it. (A job may be too difficult to be performed by any single machine.) The feasibility constraint is that no machine can be assigned more than one "sub-job".

Our problem also relates to other optimization problems, such as the Bin Covering problem, in which the goal is to use a set of items to fill completely as many bins as possible. The static MSTA instance that we study is a generalization of (weighted) Bin

Covering in that an "item" may take up different amounts of space in different "bins". In this way, an analogy can be seen between our static MSTA instance, Bin Covering, Multiple Knapsack and the Generalized Assignment [35] problems.

Many weaker and often fractional models of sensor-task assignment can be reduced to maximum matching or network flow problems, and thus can be solved optimally in polynomial time [11].

Combinatorial Auctions. Another way of understanding our problem is as a combinatorial auction, that is a silent auction in which bidders (tasks) can express preferences on bundles or *combinations* of items (sensors) [1]. Given a fixed supply of goods, the goal of the winner determination problem [103] is to maximize revenue earned from the sale of disjoint item combinations. Given that there may be exponentially many bids, this is a difficult problem to solve and therefore much of the research focus has been on AI or algorithm-engineering approaches [28].

A view expressed in the literature is that combinatorial auctions can in practice provide good approximate solutions within reasonable time for problem instances of reasonable size (in terms of number of bids) [28]. To test if this was the case for the static MSTA problem, we modeled it using the combinatorial auction formalism and we used a well-known existing algorithm for general combinatorial auctions called CASS (Combinatorial Auctions Structured Search) [38] to solve it.

3.2 Static MSTA Problem Definition

With multiple sensors and multiple tasks, sensors should be assigned in the "best" way. This goal is shared by all the both static and dynamic settings we consider in the rest of this thesis. There are a number of attributes, however, that characterize the nature and difficulty of the problem. In this section, we start by defining our network model.

Then, we propose a formal model for the MSTA problem identified by *ST-MS-IA-HE* in static settings.

3.2.1 Network Model

In our network model, we assume a set of static sensors pre-deployed in a field in a uniformly random manner. Task requests arrive all at once. By a task, we mean a sensing task that requires information of a certain type, which may be contributed by one or more deployed sensors. Each task is defined by a specific geographic location. (While task locations could be chosen from any distribution in principle, in the simulation described below task locations are chosen uniformly at random.) An example of a task is video monitoring an area of interest. General tasks that cover large areas, such as perimeter monitoring, can be decomposed into multiple subtasks, each with its own (discrete) location. The deployed sensors are directional in nature, and hence can be assigned to tasks individually. Sensors have limited sensing range R_s; a sensor provides no utility to any task that is located beyond this distance. The sensors are equipped with wireless communication equipment and have limited communication range R_c. The communication range is defined as the maximum distance over which two sensors can directly communicate. Note that in this chapter we consider a simple utility function based on signal strength, and we do not consider qualitative semantic-based measures such as if a particular type of sensor can match the sensing requirements of a particular task. These will instead be included and discussed in Chapter 4 and 5.

3.2.2 Static MSTA Problem Model

The static MSTA problem on which we first focus can be represented as a complete weighted bipartite graph, shown in Figure 3.1, where the vertex sets are composed of a set of sensors $\mathbf{S} = \{S_1, ..., S_n\}$ and a set of tasks $\mathbf{T} = \{T_1, ..., T_m\}$. Each task T_j is

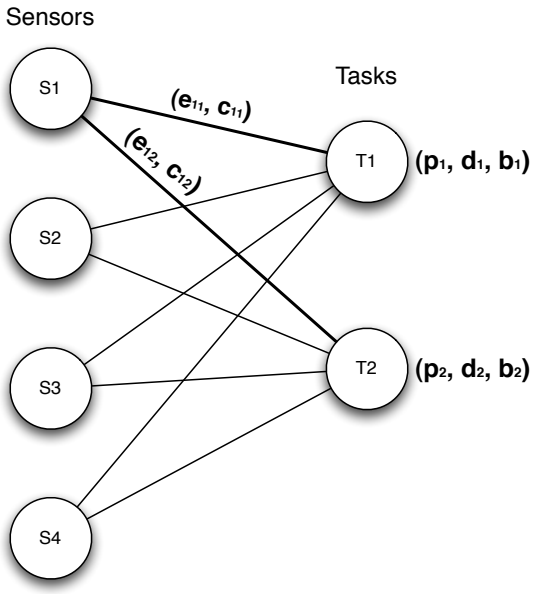

Figure 3.1: Static MSTA problem as a bipartite graph.

associated with a utility demand d_j, indicating the amount of sensing resources needed, and a profit p_j, indicating the importance of the task. Each sensor-task pair is associated with a utility value e_{ij} that task T_j will receive if sensor S_i is assigned to it. This can be a measure of the quality of information that a sensor can provide to a particular task. To simplify the problem, we assume that the utility values e_{ij} received by a task T_j are additive. While this may be realistic in some settings, in others it is not. For example, in a localization application, the utility provided by sensors is not additive as it depends on the relative angles by which they view the target. For our purpose of comparing the different centralized algorithms for the static setting, however, this utility model is sufficient for now. In Chapter 4 we will extensively address this issue, exploring more sophisticated utility models. Finally, a budgetary restriction is given in some form, either constraining the entire problem solution or constraining individual tasks as follows: each task has a budget b_j, and each potential sensor allocation has cost c_{ij}. All the aforementioned values are positive reals, except for costs and utility, which could be zero. The most general problem is defined by the following mathematical program (MP) **P**:

$$
\begin{aligned}
&\textit{maximize:} && \textstyle\sum_{j=1}^{m} p_j(y_j) \\
&\textit{such that:} && \textstyle\sum_{i=1}^{n} x_{ij} e_{ij} \geq d_j y_j, \forall\, T_j \in \mathbf{T}, \\
& && \textstyle\sum_{i=1}^{n} x_{ij} c_{ij} \leq b_j, \forall\, T_j \in \mathbf{T}, \\
& && \textstyle\sum_{j=1}^{m} x_{ij} \leq 1, \forall\, S_i \in \mathbf{S}, \\
& && x_{ij} \in \{0,1\}, \forall\, (S_i, T_j) \in \mathbf{S} \times \mathbf{T}, \\
& && y_j \in [0,1], \forall\, T_j \in \mathbf{T},
\end{aligned}
$$

A sensor can be assigned at most once ($\sum_{j=1}^{m} x_{ij} \leq 1$). Profits are received per task, based on its satisfaction level y_j. Note that y_j corresponds to the fraction of utility demand met by the sensors assigned to it, i.e. u_j/d_j within the range [0,1] where u_j is the utility cumulated by the task. With *strict* profits, a task receives exactly profit p_j iff $u_j \geq d_j$, that is iff $y_j = 1$ (provided that $y_j \in [0,1]$). With *fractional* profits, a task receives a fraction of p_j proportional to its satisfaction level y_j and at most p_j. More generally, profits can be awarded fractionally, after reaching a fractional satisfaction threshold T:

$$
p_j(y_j) = \begin{cases} p_j \cdot y_j, & \text{if } y_j \geq T \\ 0, & \text{otherwise} \end{cases}
$$

For simplicity in the later description of our algorithms and to make more explicit the dependency of this profit function on u_j, we express $p_j(\cdot)$ as a function of u_j which maps directly to the values of $p_j(y_j)$. As a matter of fact, what really varies in y_j is the absolute utility cumulated by the task (u_j) while instead the demand of utility (d_j) does not vary per task in our model. Therefore using $p_j(y_j)$ is effectively the same as adopting $p_j(u_j)$, but the second definition allows us to explicitly indicate that allocating an amount of utility which exceeds the task demands, i.e. $u_j > d_j$, does not benefit the task as it would obtain the same amount of profit p_j. We therefore express the function $p_j(\cdot)$ with respect to the variable u_j:

$$
p_j(u_j) = \begin{cases} p_j, & \text{if } u_j \geq d_j \\ p_j \cdot u_j/d_j, & \text{if } T d_j \leq u_j < d_j \\ 0, & \text{otherwise} \end{cases}
$$

When $T = 1$, program **P** is an integer program; when $T = 0$, it is a mixed integer program with the decision variables x_{ij} still integral. Note that if we impose no restriction on T, an alternative way of expressing our mathematical program **P** as a mixed-integer linear program is to use the method in [112] which allows the inclusion of step and ramp functions into a linear programming model. However, it is non-trivial to express our model through the use of that technique, in particular considering that our objective function is a sum of ramp and step functions. This would imply including $3 \times m$ extra constraints and $2 \times m$ extra variables, therefore increasing the cost associated to implementing such a model. Since our focus in this thesis is on timely solutions, this approach has not been pursued further.

The edge values e_{ij} may be arbitrary non-negative values, or may have additional structure. If sensors and tasks lie in a metric space, such as the line or plane, then edge values may be based in some way on the distance D_{ij} between sensor S_i and task T_j. In the *binary sensing* model, e_{ij} is equal to 1 if distance D_{ij} is at most the sensing range R_S, and 0 otherwise. In another geometric setting, e_{ij} may vary smoothly based on distance, according to a function such as $1/(1 + D_{ij})$.

Similarly, the cost values c_{ij} could be arbitrary or could exhibit some structure: the cost could depend on the sensor involved, or could e.g. correlate directly with distance D_{ij} to represent the difficulty of moving a sensor to a certain position. It could also be unit, in which case the budget would simply constrain the number of sensors.

Even if profits are unit, demands are integers, edge values are 0/1 (though not necessarily depending on distance), and budgets are infinite, then this problem is NP-hard and as hard to approximate as Maximum Independent Set [106]. If we also add the restriction that sensors and tasks lie in the plane and that 0/1 edge utility depends on distance (i.e., the edges form a bipartite unit-disk graph), then solving the problem optimally remains NP-hard [106].

Algorithm 3.1: Greedy

1: **while** true **do**
2: **for** each available task T_j **do**
3: $u_j \leftarrow \sum_{S_i \text{unused}} e_{ij}$
4: $j \leftarrow \arg\max_j p_j(u_j)$
5: **if** $p_j(u_j) = 0$ **then**
6: **break**
7: $u_j \leftarrow 0$
8: $c_j \leftarrow 0$
9: **for** each unused S_i in decreasing order of e_{ij}/c_{ij} **do**
10: **if** $u_j \geq d_j$ or $e_{ij} = 0$ **then**
11: **break**
12: **if** $c_j + c_{ij} \leq b_j$ **then**
13: assign S_i to T_j
14: $u_j \leftarrow u_j + e_{ij}$
15: $c_j \leftarrow c_j + c_{ij}$

3.3 Algorithms for the Static Setting

In this section we describe two algorithms to solve the static MSTA problem instance: *Greedy* and *Multi-round Generalized Assignment Problem (MRGAP)*. The former requires global knowledge of all tasks to run and hence is considered centralized, whereas the latter can be implemented in both centralized and distributed environments, a benefit in the sensor network domain. We also consider solving the problem as a *combinatorial auction*.

3.3.1 Greedy

The first algorithm we consider (Algorithm 3.1) is a greedy algorithm that repeatedly attempts the highest-potential-profit untried tasks. Because fractional profits are awarded only beyond the threshold percentage T, this need not be the task with maximum p_j. For each such task, sensors are assigned to it, as long the task budget is not yet violated, in decreasing order of cost-effectiveness; i.e., the ratio of edge utility for that task and the sensor cost. The running time of the algorithm is $O(mn(m + \log n))$. No approximation factor is given for this efficiency-motivated algorithm since, even for the first task selected, there is no guarantee that a feasible solution will be found. This by itself is, after all, an NP-hard 0/1 Knapsack problem.

3.3.2 Multi-Round GAP (MRGAP)

The idea of the second algorithm (shown in Algorithm 3.2) is to treat the tasks as knapsacks that together form an instance of the Generalized Assignment Problem (GAP). The strategy of this algorithm is to find a good solution for the problem instance *when treated as a GAP instance*, and then to do postprocessing to enforce the lower-bound constraint of the profit threshold, by removing tasks whose satisfaction percentage is too low. Releasing these sensors may make it possible to satisfy other tasks, which suggests a series of rounds. In effect, tasks not making good progress towards satisfying their demands are precluded from competing for sensors in later rounds.

Cohen et al. [22] give an approximation algorithm for GAP which takes a knapsack algorithm as a parameter. If the knapsack subroutine has approximation guarantee $\alpha \geq 1$, then the Cohen GAP algorithm offers an approximation guarantee of $1 + \alpha$. We use the standard knapsack FPTAS [122], which yields a GAP approximation guarantee of $2 + \epsilon$. Because GAP does not consider lower bounds on profits for the individual knapsacks, which is an essential feature of our sensor allocation problem, we enforce it using the postprocessing step.

Algorithm 3.2: Multi-Round GAP (MRGAP)

while true **do**
  initialize set of remaining tasks $\mathbf{T} \leftarrow \{T_1, \ldots, T_m\}$
  initialize global threshold T (in our case $T \leftarrow 0.5$)
  for $t = s$ to T, step s (in our case $s = 0.05$) **do**
    run the GAP algorithm of [22] on $\mathbf{T}$ and the *unassigned sensors*
    in the resulting solution, release any superfluous sensors
    update the tasks' demands and budgets accordingly
    if T_j's satisfaction level is $< t$ (temporary threshold), for any j **then**
      release all sensors assigned to T_j
      $\mathbf{T} \leftarrow \mathbf{T} - \{T_j\}$
    if T_j is completely satisfied OR has no remaining budget, for any j **then**
      $\mathbf{T} \leftarrow \mathbf{T} - \{T_j\}$
    if $\mathbf{T} = \emptyset$ or all sensors have been assigned **then**
      break

Both the knapasck FPTAS and GAP alogithms are reported in Appendix A for completeness. In the same section, we also describe in detail the specific implementation improvements we used, which allowed our implementation to run in reasonable time on the static MSTA instances of the size we will consider in our experiments (e.g. 500 sensors and 100 tasks).

The algorithm works as follows. A temporary threshold is initialized to a small value; e.g. 5%. In each round, a GAP solution is found based on the remaining sensors and tasks. After each round, tasks not meeting the threshold are removed, and their sensors are released. Any sensors assigned to a task that has greater than 100% satisfaction, and which can be released without reducing the percentage below 100%, are released. Such sensors are in fact *superfluous* and different strategies might be adopted to release those; in our implementation of MRGAP sensors are released in order of decreasing utility value. Sensors assigned to tasks meeting the threshold remain assigned to those

tasks. These sensors will not be considered in the next round, in which the new demands and budgets of each task will become the remaining demand and the remaining budget of each one of them. Finally, the temporary threshold is incremented, with rounds continuing until all sensors are used, all tasks have succeeded or been removed. The GAP instance solved (to within $1 + \alpha$ of the optimal) at each round is defined by the following linear program:

$$\begin{aligned} &\textit{maximize:} && \textstyle\sum_j \sum_i p_{ij} x_{ij} \text{ (with } p_{ij} = p_j \cdot e_{ij}/\hat{d}_j) \\ &\textit{such that:} && \textstyle\sum_{S_i \text{unused}} x_{ij} c_{ij} \leq \hat{b_j}, \text{ for each remaining } T_j, \\ & && \textstyle\sum_{T_j \text{remaining}} x_{ij} \leq 1, \text{ for each unused } S_i, \text{ and} \\ & && x_{ij} \in \{0, 1\}\ \forall i, j \end{aligned}$$

Here $\hat{d}_j$ is the remaining demand of T_j, that is, the demand minus utility received from sensors assigned to it during previous rounds. Similarly, $\hat{b_j}$ is the remaining budget of T_j. The concepts of demand and profit are encoded in the GAP model as $p_{ij} = p_j \cdot e_{ij}/\hat{d}_j$. This parameter represents the fraction of demand satisfied by the sensor, scaled by the priority of the task. In each GAP computation, we seek an assignment of sensors that maximizes the total benefit brought to the demands of the remaining task.

One advantage of MRGAP is that it can be implemented in a distributed fashion. For each task there can be a sensor, close to the location of the task, that is responsible for running the assignment algorithm. Tasks that do not contend for the same sensors can run the knapsack algorithm simultaneously. If two or more tasks contend for the same sensors, i.e. they are within distance $2R_s$ of one other (where R_s is the sensing range of a sensor), then synchronization of rounds is required to prevent them from running the knapsack algorithm at the same time. To do this, one of the tasks (e.g. the one with the lowest id) can be responsible for broadcasting a synchronization message at the beginning of each new round. However, since R_s is typically small compared to the size of the field, we can expect many tasks to be able to do their computations simultaneously. As noted in Chaper 2, the authors in [114] implemented in a distributed fashion this algorithm extending it to their problem settings.

The total running time of the algorithm depends on the threshold T and the step value chosen, as well as on the density of the problem instance, which will determine to what degree the knapsack computations in each round can be parallelized.

3.3.3 Combinatorial Auctions

The sensor allocation problem in its static settings can be formulated as a combinatorial auction in which the bidders are tasks $T_1, ..., T_m$, the items are sensors $S_1, ..., S_n$, and each task places a bid, equal to its own profit, for any set of sensors which satisfy the task's demand and respect the task's budget constraint. Formally, we compute the *value* that bidder T_j obtains if he receives the bundle B of sensors using the following *valuation function*:

$$v_j(B) = \begin{cases} p_j, & \text{if } u_j \geq d_j \text{ , } w_j \leq b_j \\ p_j \cdot u_j/d_j, & \text{if } T \leq u_j/d_j \text{ , } w_j \leq b_j \\ 0, & \text{otherwise} \end{cases}$$

where $u_j = \sum_{S_i \in B} e_{ij}$ represents the joint utility of the sensor bundle B, and $w_j = \sum_{S_i \in B} c_{ij}$ its total cost to the task T_j.

We list all the bids for each task T_j as pairs $(B, v_j(B))$ and we use the OR* bidding language [80] to explicitly introduce mutual exclusion between bids placed by the same task. The CASS anytime algorithm then conducts a structured depth-first search on the list of all bids trying to find the subset of bids which maximize the total revenue under the constraint that each item can be allocated to at most one bidder, i.e. a solution to the classic *winner-determination problem* [103].

The number of bids is potentially exponential if we adopt the naive approach of enumerating all nonzero-value bids containing bundles respecting the tasks' demand and budget constraints ($O(m \cdot 2^n)$ bids in the worst case, but lower in practice). In the experiments in Section 3.4, we adopt this naive approach to test if the number of bids

generated within our scenario is computationally tractable. As a comparison, we also consider a constrained version in which the number of bids is not exponential. In this particular case we assume that tasks behave as *single-minded bidders*, i.e. each task bids only for the bundle of sensors which evaluates to be the best bundle it can obtain (using the valuation function $v_j(B)$), therefore in total there will be at most m bids. Each "optimal" bid is generated using the standard knapsack FPTAS (adopted also to implement MRGAP in Section 3.3.2): for each task we solve the knapsack problem of finding the subset of sensors which maximizes the total task profit using the profit function $p_j(u_j)$ (defined in Section 3.3), while respecting the task's budget constraint. Finally we post-process the sensor bundle generated by this algorithm by removing any superfluous sensors, obtaining what can be considered the best sensor bundle for each particular task.

3.4 Performance Evaluation

In this section we present the results of the simulations we performed. We implemented the schemes in Java and tested them on randomly generated problem instances. We evaluate the algorithms proposed for the static setting, which we implemented in a centralized fashion. For clarity's sake, we present a sampling from our comprehensive series of experiments, evaluating the behavior of the schemes on a variety of problem input types.

3.4.1 Simulation Setup

Recall the model outlined in Section 3.2. Each task T_j has a demand, which is an abstract value of the amount of sensing resources it requires, and is awarded profit $p_j(u_j)$, based on its importance and its satisfaction level. Each sensor can only be assigned to a single task. Once assigned, the full utility of the sensor is allocated to

support the task. We consider a task to be successful if it receives at least 50% of its demanded utility from allocated sensors (i.e. $T = 0.5$).

Task demands are chosen from an exponential distribution with an expected value of 2 and a minimum of 0.5. Profits for the different tasks are also exponentially distributed with an expected value of 10, a minimum of 1 and a maximum of 100. To enforce a minimum on the distribution we simply sum the minimum value to the numbers sampled from the exponential distribution. Instead to enforce the maximum value, we flatten the tail of the distribution (towards infinity), i.e. for any generated number which is greater than the maximum we impose it to be equal to the maximum value. Using an exponential distribution to generate task demands simulates realistic scenarios in which many tasks demand few sensing resources and a smaller number demand more resources. The same applies to profit. The simulator filters out any task that is not individually satisfiable; i.e., satisfiable in the absence of all other tasks. For a sufficiently dense network, however, we can expect there to be few such impossible tasks.

The utility of a sensor S_i to a task T_j is defined as a function of the distance D_{ij} between them. Many sensor types exhibit some kind of quality deterioration or signal attenuation based on distance. In order to evaluate their utilities to tasks, we assume that all sensors know their geographical locations. Formally, the potential utility contribution is:

$$e_{ij} = \begin{cases} \frac{1}{1+D_{ij}^2/c}, & \text{if } D_{ij} \leq R_s \\ 0, & \text{otherwise} \end{cases}$$

where R_s is the global sensing range. This follows typical signal attenuation models in which signal strength decays inversely with distance squared [66]. Note that this utility function is only used for testing in our experiments and is not a property of our algorithms; it is not meant to model the exact behaviour of any sensor. In our experiments, we set $c = 60$ and $R_s = 30m$.

Nodes are deployed in uniformly random locations in a $400m \times 400m$ field. Tasks are created in uniformly random locations in the field. The communication range of

all sensors is $R_c = 40m$, and we ensure that the sensor network is connected. (If a randomly created instance in our simulation is not connected, it is discarded and we repeat, until a connected instance is produced.) We do not consider communication overhead other than noting the fact that in a static setting, a centralized approach will typically collect all task requests and all information about sensors (in our case, sensor locations) in a base station and then communicate all the allocation decisions back to the sensors. The communication overhead remains therefore constant with the static number of sensors and tasks on the field, differently from the dynamic scenario where tasks appear over time and in which the communication overhead depends therefore on the task arrival rate. For this reason, while in static settings the centralized approach is feasible, as all tasks arrive at once, in a dynamic environment a distributed approach (as in Chapter 5) is more desirable, as resilient to loss of connectivity with a central node due to spikes in the number of task requests over time (as discussed in Chapter 4).

3.4.2 Static Setting Results

In this experiment, all tasks occur simultaneously. We fix the number of sensors in the field and vary the number of tasks from 10 to 100. Each sensor has a cost, chosen uniformly at random from $[0, 1]$, which does not depend on the task it is assigned to. This can represent the sensor's actual cost in real money, the energy spent to operate the sensor or, e.g., a value indicating the risk of discovery if the sensor is activated. Each task has a budget, drawn from a uniform distribution with an average of 3 in the first experiment and varying from 1 to 10 in the second. As mentioned in Section 3.3.3, the static problem can be modeled as a combinatorial auction. The general CASS algorithm is not computationally tractable in our scenarios as the number of bids grows exponentially. Even for smaller scenarios its performance was inferior to the greedy and MRGAP algorithms. So, in the following results we only show the CASS algorithm with single-minded bidders which is more tractable. In the following

results, we show the average of 20 runs, which we found experimentally were necessary in order to get significative comparable results. In particular, by running each of the experiments 20 times we ensured a low standard deviation error (i.e. around $\pm 2\%$).

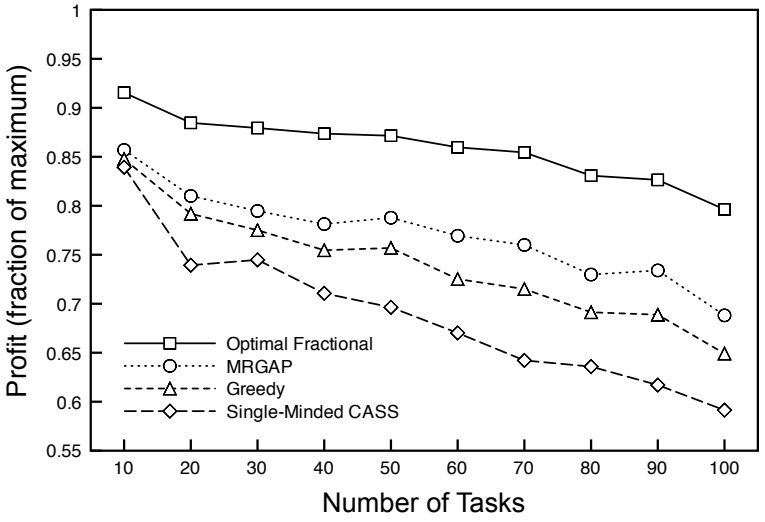

Figure 3.2: Fraction of maximum profit achieved (250 nodes)

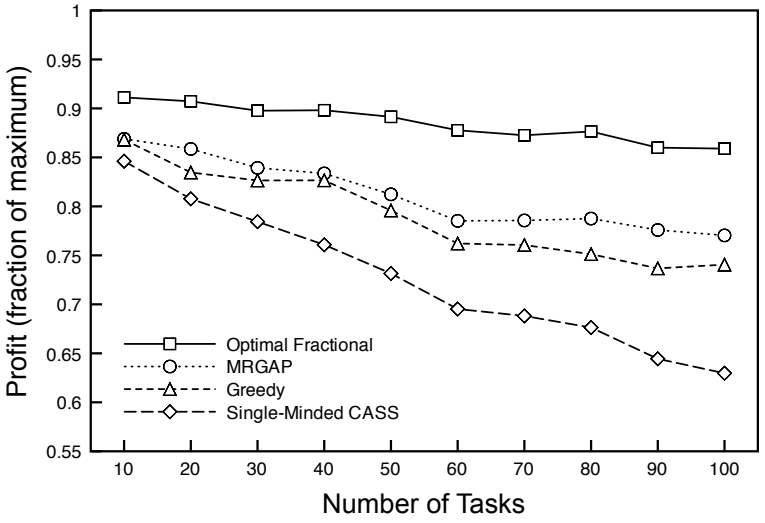

Figure 3.3: Fraction of maximum profit achieved (500 nodes)

The first series of results show the fraction of the maximum task profits (i.e., the sum of all task profits) achieved by the different schemes. We show the profits for the greedy algorithm, MRGAP, single-minded CASS, and an upper bound on the optimal value,

running on two classes of sensor networks, sparse (250 nodes) and dense (500 nodes) (Figures 3.2 and 3.3, respectively). The upper bound on the optimal is obtained by solving the LP relaxation of program **P**, in which all decision variables are allowed to take on fractional values in the range $[0, 1]$, and the profit is simply fractional based on satisfaction fraction; i.e., $p_j y_j$ for task T_j with no attention paid to the threshold T. The actual optimal value will be lower than the fractional one.

The MRGAP scheme, which again can be implemented in a distributed fashion, achieves higher profits in all cases than does the greedy scheme, which is strictly centralized (because tasks have to be ordered in terms of profit). The difference, however, is not very large. The single-minded CASS algorithm achieves lower profits in both settings. We note that with 500 nodes the network achieves higher profits, which is to be expected.

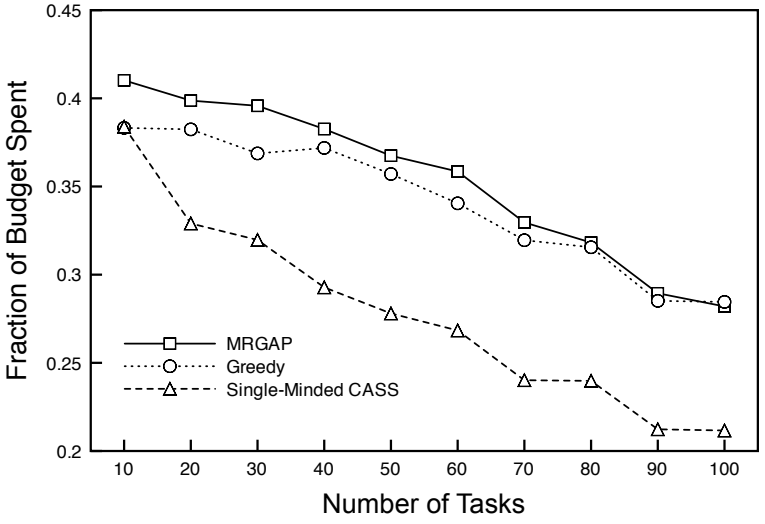

Figure 3.4: Fraction of spent budget (250 nodes)

Figure 3.4 shows the fraction of the total budget each scheme spent to acquire the sensing resources assigned to it, in a network with 250 nodes. The MRGAP scheme achieves more profit than the greedy algorithm and spends a modest amount of additional resources. The single-minded CASS algorithm spends the smallest fraction of the budget. The fraction of remaining budget is significant (more than 60% in almost all cases), which suggests either that successful tasks had higher budgets than they

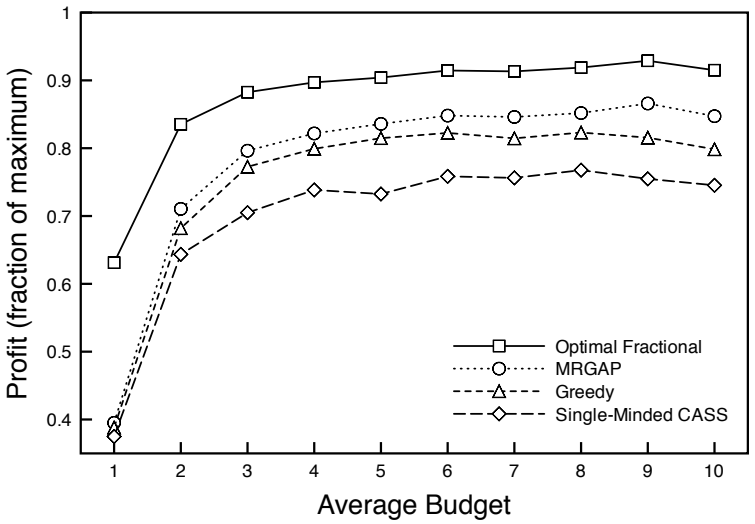

Figure 3.5: Varying the average budget (250 nodes)

could spend on available sensors or that unsuccessful tasks had lower budgets than necessary, preventing them from reaching the success threshold, and so their budgets were not spent. When the number of tasks is large, the unspent budget can be attributed to the fact that there simply were not enough sensors due to high competition between tasks. We found that with 250 sensors and 30 tasks, on average about 82% of the tasks succeed and the remaining 18% fail. The successful tasks spend only 50% of their budget. Among the failed tasks, 50% fail because there are not enough resources for the tasks to reach the success threshold. In this case, on average, the available sensors provide only 24% of the demand, whereas 50% is needed for a task to be successful. The other 50% of the tasks fail because their budgets do not allow them to allocate enough resources to reach the success threshold. When the number of tasks is increased to 60, we found that on average 74% of the tasks succeed and the remaining 26% fail. Again, the successful tasks spend only 50% of their budget but we found that a higher percentage of the failed tasks, 67%, fail because there is not enough resources, i.e. due to competition.

We performed another set of experiments, in which the number of tasks was fixed at 50 and the average task budget varied from 1 to 10. Figure 3.5 shows the results for

a network with 250 sensors. We observe that the achieved profit initially increases rapidly with budget size but slows as the influence shifts from budget limitations to competition between tasks. We observe the same pattern: MRGAP achieves highest profits followed closely by the greedy algorithm. The single-minded CASS algorithm comes in third.

Figures 3.6 and 3.7 show the same results when 500 nodes are deployed in the same area. We notice that more of the budget is spent in this case. Since there are more sensors, tasks are able to spend more of their budget and reach higher profits than when 250 sensors are deployed. We also note that the achieved profits are higher when we vary the amount of budget given to tasks. Again, this is because there are more sensors on which the tasks' budgets could be spent.

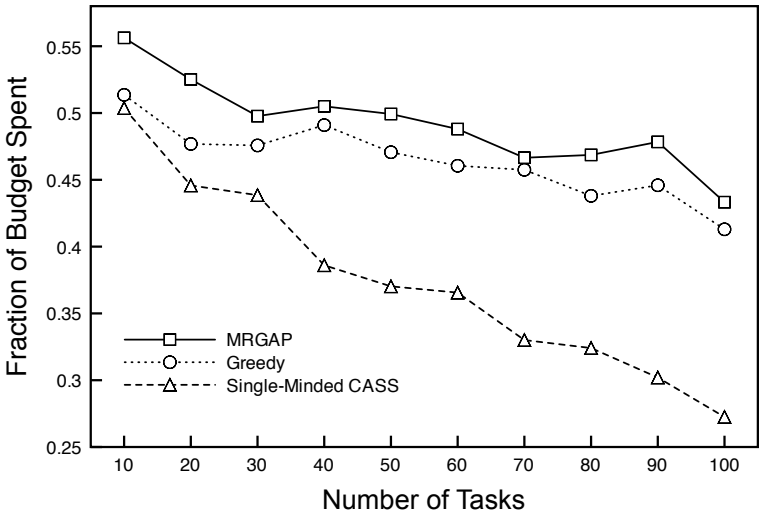

Figure 3.6: Fraction of spent budget (500 nodes)

As noted in Section 3.3, the static MSTA problem is NP-Hard and therefore it is worth considering also the performance of the allocation mechanisms in terms of computational time. In Figure 3.8, we show that both MRGAP and Greedy provide timely solutions to our static MSTA problem model, while instead the combinatorial auction approach (Single-Minded CASS) displays the worst computational-time performance. These benchmarks are consistent with the theoretical computational time of the al-

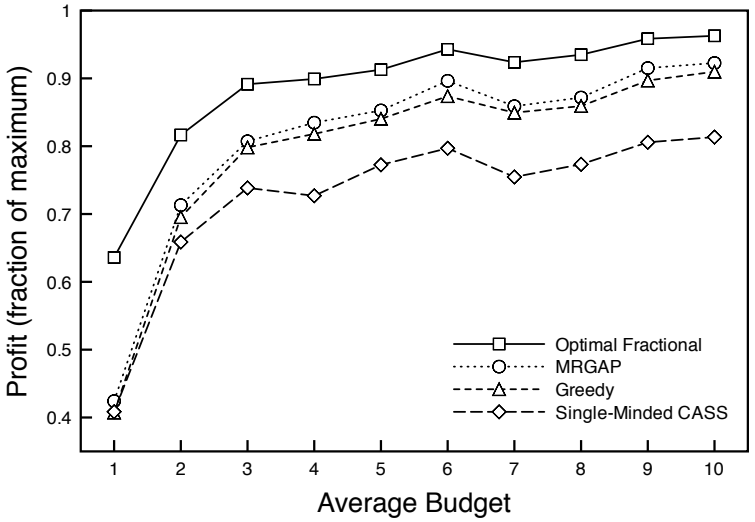

Figure 3.7: Varying the average budget (500 nodes)

gorithms presented and were measured on a Macbook Pro with a 2.4 GHz Intel Core 2 Duo processor and 4GB 667 MHz DDR2 SDRAM, running Mac OS X Snow Leopard (version 10.6.8). It is important to note that in the implementation of these algorithms we did not focus on optimizing the code in terms of computational runtime, and therefore implementation improvements may be possible although the general trend should not change. In these experiments, we fix the number of sensors on the field to 250 and we increase gradually the number of tasks from 10 to 100; locations of sensors and tasks on the field, profits, utility demands and budgets were randomly generated with the same parameters and distributions used in the first set of experiments (shown in Figure 3.2 and Figure 3.4). The time performance displayed by the greedy algorithm is the most desirable, being almost constant in the range of 10-20 ms. Instead, the computational time for MRGAP grows sub-linearly with the number of tasks on the field and falls in the range from 300 ms (10 tasks) to 1300 ms (100 tasks). Although MRGAP takes longer than Greedy to converge to a solution, its performance is still very desirable compared to the Single-Minded CASS approach, which stays within the range from 400 ms (10 tasks) to 3400 ms (100 tasks) and grows with a steeper inclination. Note that the standard deviation for the time measurements taken is around ± 7 ms

for the Greedy, among ± 150 and ± 220 ms for MRGAP and finally for Single-Minded CASS varies from ± 150 to ± 400 ms. These results indicate that MRGAP, although showing a higher computational time than Greedy, scales well with the number of tasks on the field, still providing the highest profit and making the best use of the assigned budget.

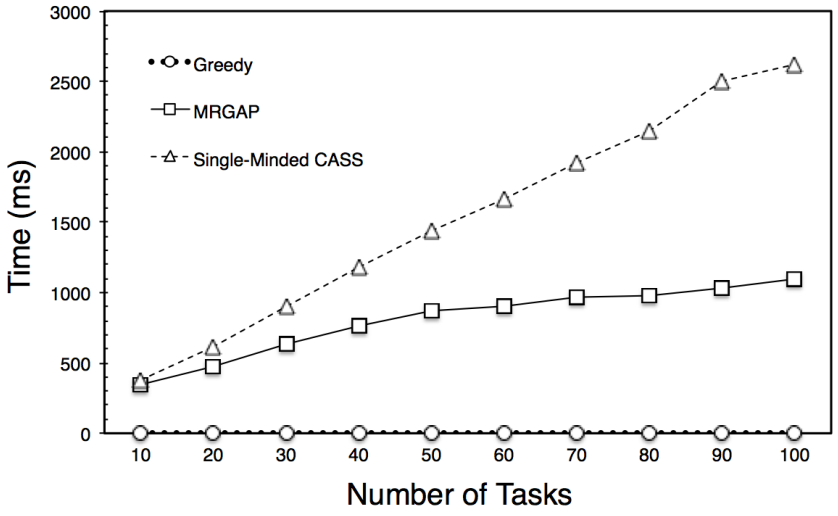

Figure 3.8: Computational time (250 nodes)

3.5 Discussion & Conclusion

In this chapter, we defined the MSTA problem instance *ST-MS-IA-HE* in a static setting where all tasks arrive at once. This scenario is motivated for example by situations in which different users are granted control over the sensor network at different times. Sensing resources may be limited and expensive, both in terms of equipment and operational cost. To limit the request for sensing resources of the current user who will want to satisfy as many of these tasks as possible, we assign a budget to each task. This serves as a constraint in terms of the amount of resources that can be allocated to it regardless of profit, each sensor is therefore associated to a cost when serving a particular task. We assume that the utility of sensors to tasks is additive; i.e., utilities of sensors

contributing to the same task add up linearly. We use a simple function to evaluate the sensor utility which models the real degradation of the sensing performances with the varying of the distance between sensor and task. This problem can be modeled as a bipartite semi-matching problem and we provide a formal mathetical model clearly defining our optimization objective.

We proposed three algorithms to solve this problem and evaluated these through simulations. We found that in dense networks the three algorithms perform well, with our novel Multi Round GAP algorithm outperforming the greedy and the combinatorial auction based algorithm in terms of profit achieved. We also study how tasks spend their budgets and how increasing the budget affects the overall profit of the network. We found that in most cases giving tasks higher budgets does not help since what limits the amount of achievable profits is competition and not the amount of budget provided. We showed that the three algorithms provide timely solutions to the static MSTA problem. In particular, MRGAP takes longer to converge to a solution compared to the greedy algorithm, but the quasi-logarithmic increase in runtime when the instance size grows makes of it a very desirable alternative to the combinatorial auction algorithm which displays the worst computational time performance.

Moreover, we observe that Multi Round GAP is a novel algorithm that treats tasks as knapsacks that together form an instance of the Generalized Assignment Problem (GAP) and it could be implemented in a distributed fashion. We therefore answered *Research Question 1*, modelling the static *ST-MS-IA-HE* problem and exploring different scalable algorithms able to find a solution which is reasonably close to the optimal in both sparse and dense sensor networks considering different amounts of task requests. It is worth reminding that, as mentioned in Chapter 1, by scalable we mean a solution whose performance in terms of allocation quality (i.e. overall sensor utility allocated to most important tasks) does not drastically decrease in the presence of a large number of tasks in the field, and that is able to provide timely solutions considering that the static MSTA problem is NP-Hard.

Although we do not consider them in this chapter, there are many other profit functions that may warrant study. Obviously zero utility should yield zero profit, and full utility full profit. For the shape of the profit curve lying between these two extremes, natural options include: linear, full profit only for fully met demands, linear only after crossing a threshold, smooth convex (often difficult to maximize) and smooth concave (often easier to maximize) [83]. In the next chapter (Chapter 4), for the dynamic setting, we choose to use a profit model which includes both linear and non-linear utility functions between zero profit and full profit. In the static MSTA model we consider a simple additive utility function and therefore we limit the heterogeneity of tasks able to be supported by these mechanisms to the ones that do not require non-additive utility evaluations.

More sophisticated utility models, taking into consideration not just physical attributes such as distance but also some sort of semantics, could be developed. Recent work by Bisidikian [13], for example, considered the effects of sensor sampling models and quality of information. In this chapter, we based the potential utility of a sensor-task pairing on their separating distance. However, in the case of video sensors, for example, the closest sensor to an event might not be the best candidate for selection because its view of the field is obstructed or it cannot provide sufficient frame rate.

In this chapter, we take the simplistic assumption that all sensors provide the same type of information, possibly with varying quality. This is reasonable when we do not have enough knowledge about the task types which need to be supported and sensors which are deployed on the field. In this case, a simple utility function based for example on the degradation of the sensing capabilities with the distance could provide a fair approximation to evaluate the goodness of a sensor compared to others. Although, in more realistic scenarios tasks may have complex requirements which might require sensors with specific capabilities (e.g. infrared or acoustic sensor). In these cases, we might consider task requirements and sensor capabilities be defined using explicit representation tools, such as ontologies (e.g. [19] for the robotics field). In the next chapters, we

include therefore in the utility estimation both a quantitative measurement, e.g. based on the distance separating sensors and tasks, but also qualitative measurements, e.g. based on semantic matchmaking of heterogeneous sensor capabilities provided and required.

Finally, in this chapter, we focused on MSTA in a static setting for which we propose centralized solutions (although MRGAP could be implemented distributely). In a more realistic emergency response scenario, however, we cannot expect that all task requests will be generated at the same time and collected in a single node, e.g. a base station. The natural organization of such operations is indeed distributed and usually the information is collected in multiple distributed nodes. In the next chapter (Chapter 4), we therefore focus on the dynamic MSTA problem instance *ST-MS-IA-HE* for which a distributed solution (described in Chapter 5) is more appropriate. The sensor network should, in fact, support tasks also in the case of no connectivity with a base station, e.g. in the case of heavy network traffic or events which might cause the collapse of communication with the central node. Moreover, in Chapter 4, we observe that moving to MSTA in a dynamic setting, obliges us to include the time dimension in the problem formulation, to consider the duration over time of each task. Therefore, differently from the static setting, the goal of the dynamic model in the next chapter is to maximize the support of tasks over time.

Chapter 4

Dynamic MSTA with Generic Sensor Utilities

In this chapter, we propose a formal model for the dynamic MSTA problem instance *ST-MS-IA-HE* where the tasks arrive and depart at different times, generalising it from two specific sensor allocation problems. We allow for the utility of sensors to tasks to be non-additive; i.e., utilities of sensors contributing to the same task could be for example sub-additive or super-additive.

In Chapter 3 we studied a version of *ST-MS-IA-HE* with a simplified model in which utility from multiple sensors is assumed to combine additively. Here we study more complex utility models, starting from two particular applications: event detection and two-dimensional localization tasks. We formalize these two problems as static MSTA instances with *non-additive utilities*, considering restricted scenarios in which we know in advance that all the tasks consist of just a single type; i.e., only event detection or only 2D localization tasks.

We then generalize these two problems into a more abstract *dynamic MSTA problem*, in which we allow for any *non-additive utility function* and include the time dimension in the problem formulation. We now first need to group sensors into bundles, and then evaluate their *joint utilities* using either an additive or a non-additive utility function. Finally, we must decide which is the best assignment of *sensor bundles* to *tasks*. We model this problem as a tripartite semi-matching problem. As in Chapter 3, we asso-

ciate each task with a demand value and a profit value; but in this case we associate each *sensor bundle*-task pair with a joint utility offer, whereas in the previous chapter we associated each *sensor*-task pair with an individual utility offer. Given our dynamic setting, we assume there are no budget constraints for each task as these may be too restrictive because we must react to new tasks given our current operating environment; that is, the condition of the sensors will change over time. For this reason, we allow also for *re-allocation* of sensors to newly-created more important tasks which means sensors could be *preempted* from ongoing tasks. We also include a re-allocation cost in the formal model to avoid preemption happening too often.

The dynamic setting is useful in an emergency response scenario where we need to adapt to changes and respond promptly to tasks, while instead the static one was motivated by scenarios with a set of long-lived tasks. The emergency response environment is, in fact, more often *highly dynamic* with users moving and generating tasks at different rates. The model presented in this chapter provides more flexibility for heterogeneous sensors and tasks allowing for both additive and non-additive utility functions. Users in the field typically lack the time and expertise to manually decide which are the best sensors for their tasks, therefore solving the Multi-Sensor Task Allocation problem is even more relevant. A dynamic allocation mechanism is presented in the next chapter (Chapter 5) where we also discuss how we use the two models developed in this chapter for event detection and 2D localization as examples to evaluate our distributed system performance.

As already noted, the dynamic MSTA problem studied in this chapter considers *Single-Task sensors* supporting *Multi-Sensor tasks*. Although we focus on a dynamic setting, we assume that we still do not have enough information about the future, given that the operation plan might change as unexpected events may happen. For this reason we do not use, for example, information about the task duration to drive our allocation mechanisms; this therefore constrains us to perform *instantaneous allocation*, without the possibility to schedule sensor allocation for future tasks. Therefore our focus still

remains on the *ST-MS-IA-HE* problem instance. The difference here, compared to the previous chapter, is that we allow for non-additive utilities and dynamic re-allocation of sensing resorces. In Chapter 5, we show that the first helps us to deal better with heterogeneous sensors and tasks, while the second improves the quality of support to high priority tasks over time.

Finally, in this chapter we propose a *conceptual architecture* to solve step-by-step the dynamic MSTA problem, using both qualitative (i.e., semantic based) and quantitative (i.e., numerical utility based) metrics to evaluate sensor bundle utilities. We describe how our conceptual architecture combines a knowledge-based component with an allocation mechanism to address the complex problem of evaluating heterogeneous sensor bundle utilities. Given the flexibility of the dynamic MSTA model, we allow for any heterogeneous sensor types and task types in the field, with the only assumption that we can model explicitly their sensing capabilities and requirements; e.g., using the ontologies in [44]. In fact, as highlighted in Chapter 2, our approach follows the example of other modular solutions for heterogeneous resource allocation (e.g. [77, 32]), which explicitly model resource capabilities and task requirements in order to split the allocation process in two parts. First, it considers the type of resources required by each task, using explicit capabilities/requirements representations (e.g. ontologies), and then it decides the allocation of the actual resource instances (of that particular type) to each task which might be competing for their exclusive usage.

The rest of this chapter is organized as follows. In Section 4.1, we cover some related problems to event detection, 2-dimensional localization problems and then the more abstract dynamic MSTA model. In Section 4.3, we then study in detail event detection and 2D-localization MSTA instances with non-additive sensor utilities in a static setting. We generalize them in Section 4.4 as a dynamic MSTA problem which allows for both additive and non-additive sensor utilities. In Section 4.5 we provide an overview of our conceptual architecture, and discuss central and distributed implementations of such architecture, highlighting how the centralized is unrealistic in our dynamic set-

ting. Finally, Section 4.6 concludes the chapter with additional discussion about the formalized problems.

4.1 Related Work

In the past, sensor-task assignment problems in wireless sensor networks have been studied mainly using simplified models in which utility from multiple sensors is assumed to combine additively. For example, [37] uses distributed approaches to assign individual sensors to tasks, assuming additive utility and no competition for the same sensing resources between tasks. [106] analyzes a variant of the problem, considering competition between different tasks for the exclusive usage of sensors. A richer problem motivated by conservation of resources in the static setting is addressed in Chapter 3. In this chapter, we consider more complex models to evaluate the utility of a bundle of sensors and describe a conceptual architecture able to address this issue.

As noted in Chapter 2, directional sensors with tunable orientations have recently been addressed for coverage [4] and target tracking [16] problems separately. For non-directional sensors, both [3] and [49] propose algorithms to provide a certain level of (cumulative) detection probability over a certain area. Target localization problems have also been previously considered (e.g., in [125]) which develops a centralized solution using a prior distribution of target location and sensor locations. A distributed solution for the localization problem is proposed in [60], but it does not consider competition on resources between multiple simultaneous tasks.

Similarly to our hypothesis of non-additive utility of sensor bundles, in the MRTA framework [40] where the authors study the ST-MR-IA problem instance, they make no assumptions on the combined utilities of a group of robots. They model this problem as a Set Partitioning Problem, proposing to solve it using pre-existent approximation algorithms [7]. These algorithms do not scale well to large numbers of sensors and tasks, which are instead features of our application domain, and they are not easily

extensible to a dynamic environment, as noted in [40]. The algorithms proposed refer to an entire sub-class of problems all having the same high level features, thus not taking advantage of the specific characteristics of the problem instance. In Chapter 5, we exploit features of the dynamic MSTA problem defined in this chapter to design a scalable distributed system.

Event Detection. Our MAXCDP problem (defined below) lies within a family of submodular combinatorial auctions. Guaranteed approximation algorithms are known for this class of problems (see for example [68] and references therein). Our focus here, however, is in the design of a model for the specific Multi-Sensor Task Allocation problem where all tasks are exclusively of the "event detection" type. The aim of our model is in fact to give an example of how non-linear utility functions are necessary to evaluate the usefulness of sensors for particular task types. Some related problems involving cumulative probabilities are considered in [36], but those problems involve the product of task success probabilities instead of the sum. MAXCDP is a simplified version of a much more complicated event detection problem defined in [121]. In both problems, the underlying detection probabilities, which depend on the strength of the event and the relative geomemtry, are given part of the problem input, and so the calculation of them is beyond the scope of this thesis. A typical model, however, is that detection likilhood falls like $1/r^{\alpha}$ based on distance r, for some $\alpha > 2$ [31].

2D Localization. Problems related to the localization problem we study below, in which pairs of sensors are assigned to targets, have recently received attention. [42] studied what have been called Focus of Attention problems, with the objective functions of (1) minimizing the sums of aspect ratios (i.e., the ratio of distance of sensors to target and distance between sensors) or (2) minimizing (the sum of) the deviation of the angle formed by sensor-target-sensor from ninety degrees. (These problems were previously considered by [53] and [41], respectively. [46] gives hardness results for a generalization of the formulation from [53].) [42] proposes a solution for similar problems, but only for restricted settings, such as a deployment in which all sensors

lie on a base line. Our problem differs in that both angle and distance from target are incorporated into a single objective function, and in that both sensors and targets lie in the plane.

The MSTA problems studied here, in which sensors are partitioned among multiple tasks, are distinct from problems of *sensor selection*, in which a subset of sensors is chosen typically to optimize for a single task, e.g. [12, 26], or where if multiple tasks are considered then often individual utilities are assumed to combine additively, as noted in Chapter 2. Finally, distributed coordination among different entities (agents) in order to form coalitions to achieve a common subgoal in a system has been extensively studied by the multi-agent community and is referred to as *coalition formation* problem; e.g., in [109, 115]. Different distributed and centralized architectures have also been proposed in order to allocate heterogeneous resources to tasks in both sensor networks and multi-robot system environments, e.g. [32, 114]. See references in Chapter 2 for an in depth discussion of these.

4.2 Network Model

We consider a network consisting of static sensors of different types. The deployed sensors are *directional* in nature. Examples of such sensors include imaging sensors, which can be used for event detection, and directional acoustic sensor arrays. The latter can be used both for event detection and, when directed in pairs towards an estimated location, to accurately localize a target. Such sensors cannot be used to perform multiple tasks unless these tasks are all in the same locality. Thus, we assume that a sensor or a bundle of sensors can be assigned to at most one task at a time. We also assume that sensors know their location.

In our model, a task is specified by a geographic location and a task type, for example, detecting events occurring at location (x, y) or accurately localizing a target within a small area known to contain the target's estimated location. A larger-scale task, such as

field coverage or perimeter monitoring, can be divided into a set of tasks, each having its own location. Examples of such larger-scale tasks we envision include coverage of a strategically important intersection, say, or of major crossing points along a border. In such a case, the larger task would be divided into a not-unreasonable number of independent (sub)tasks.

In the network, there may be multiple types of tasks, each having different sensing requirements. In many scenarios, a task will require more than one sensor in order to satisfy its requirements. Some task types may only require that the assigned sensors are close to the task location, others may require that the collection of sensors form a specific shape. For example, a larger number of sensors given to a detection task will increase the detection probability, but even a single sensor may provide some help. For other tasks, such as localization, there may be a minimum number of sensors required to obtain any benefit, and the collection of chosen sensors must form a specific shape.

Task requirements are therefore complex and usually require a non-additive function to evaluate the utility of a particular sensor bundle. The specific characteristics of a given task's requirements naturally allow us to restrict our attention to just the applicable sensors. These characteristics are: (1) the type of sensor compatible with the task, i.e. *can* a sensor type (or collection of sensor types) potentially satisfy the information requirements of a task, (2) the utility value of a sensor or group of sensors to a particular task, i.e. *how well* a sensor bundle might perform. The utility estimate can depend for example on the distance from the task location, or relative angles between sensors and in general can be either additive or non-additive.

As an example, we consider below two types of tasks incorporating these two requirements. The first task we consider is an *event detection* task in which the goal is to detect activity at a specific location. This task can be accomplished using one or more sensors. Each sensor has a detection probability that depends on its type and distance from the target. A collection of sensors can be combined together to improve the detection probability. Usually such tasks can use any sensor type that can detect activity;

e.g., acoustic or seismic [129] and imaging sensors [75].

The second task type we consider is target localization or in general *two-dimensional localization*, where the goal is to precisely localize a target within the small area where it is expected, perhaps prompted by the detection of an event in this area or by some prior knowledge. This type of task requires at least two sensors to triangulate the location of the target on the field. An interesting property of this task type is that assignment quality depends not only on sensor type and separating distance but also on the angle between the selected sensors. In the model of [63], for example, two sensors perform optimally if they are separated by a 90° angle and are as close to the target as possible. Usually such tasks can be satisfied by acoustic or radar sensors [50], or more in general phased (acoustic) arrays [48].

Note again that we first formalize both event detection and 2D localization MSTA instances in a *static setting* in which all task requests arrive at the same time. We then generalize these into a more abstract sensor-task allocation problem in a *dynamic setting* which allows for both additive and non-additive sensor utilities in Section 4.4.

4.3 Two Non-Additive Sensor Utility Models

In this section, we study two particular MSTA problems, for *event detection* and *two-dimensional localization* tasks in a static setting. We assume that all the tasks in each problem instance are going to be of a single type, and we use this information to refine the formulation of each problem. We explore these as examples of non-additive sensor utility models to show that considering such utility schemes is necessary when supporting sensing tasks in heterogeneous sensor networks, such as in an emergency response operation. Below we provide a formal definition of both event detection and 2D-localization MSTA problems in a static setting.

4.3.1 Event Detection

The goal of the detection task is to detect events in a specific location with the highest probability. [121] gives a complex model of sensor assignment, with an objective function based on the probability of detecting certain kinds of events, conditioned on the events occurring and the number of sensors assigned to detect the event in a given location.

We extract the kernel of this problem as follows. Given are collections of sensors and tasks. Each task is to monitor and detect events, if they occur, in a certain location. The utility of a sensor to a task is the detection probability when the event occurs. Let $S_i \rightarrow T_j$ indicate that sensor i is assigned to task j and let p_j indicate T_j's profit. Sensor detection events are assumed, for simplicity and concreteness, to be independent, under the assumption that false negatives are the result of random failure of the individual sensor components. The objective is then to maximize the sum of cumulative detection probabilities for tasks (weighted by task profits), given the probability e_{ij} that a single sensor S_i detects an event for T_j:

$$\sum_{T_j \in \mathbf{T}} p_j (1 - \prod_{S_i \rightarrow T_j} (1 - e_{ij})) \tag{4.1}$$

Equation 4.1 is based upon the model that the responses of sensors are statistically independent of one other. This is based upon the reasonable assumption that the random fluctuations measured by the sensors are statistically independent given that the sensors are either disparate over space and/or modality. We call the resulting problem the Cumulative Detection Probability maximization problem (MAXCDP). Here the utilities are monotonic increasing but nonlinear, as sensors are assigned. [107] proves that MAXCDP is strongly NP-Hard and that the hardness result remains even for geometric instances, even if sensors and tasks lie on a line.

4.3.2 Two Dimensional Localization

For target localization through triangulation of the bearing measurements, two or more sensors that are not collinear with the target are necessary to ensure full observability of the target's location. The expected mean squared error when incorporating imperfect bearing measurements is well understood [57, 61]. Specifically, it can be shown that when the bearing measurements are modeled as the true bearings embedded in additive white Gaussian noise (AWGN) of mean zero and variance σ^2, then the error covariance of the (x, y) location of the target is approximately:

$$\mathbf{R} = \left[\sum_{i=1}^{n} \frac{1}{\sigma^2 d_i^2} \begin{pmatrix} \cos^2\theta_i & -\cos\theta_i \sin\theta_i \\ -\cos\theta_i \sin\theta_i & \sin^2\theta_i \end{pmatrix} \right]^{-1}$$

where d_i and θ_i are the distance and bearing, respectively, from the target event to the i-th sensor. We choose to model the uncertainty in the calculated target location, U, as a function of the expected mean squared error (MSE), which is $U = \text{trace}\{\mathbf{R}\}$. Alternatively, the uncertainty could be $U = \det\{\mathbf{R}\}$ as described in [63]. We prefer the trace because of its physical interpretation as the MSE and because it bounds the determinant. As noted in [61], the error covariance $\mathbf{R}$ is related to the inverse of the Fisher Information Matrix. Therefore, clearly this approach relates to the ones covered in Section 4.1 and Chapter 2 as it uses the inverse of maximum-likelihood estimates of target locations as an uncertainty measurement. Such uncertainty estimates are, in fact, the common trait of most of the localization schemes, which as discussed are based on entropy, mean squared error and information-gain approaches.

We consider the case in which only two sensors are used for each localization task, which in most cases provides enough accuracy. More sensors lead to better accuracy but for the purpose of analyzing a non-additive utility problem two sensors are sufficient. As before, a sensor may not be assigned simultaneously to more than one task. For the case of sensors 1 and 2 assigned to task j, the uncertainty of the localization

computation is given by:

$$U_j = \sigma \frac{\sqrt{D_{1,j}^2 + D_{2,j}^2}}{|\sin(\theta_{1,j} - \theta_{2,j})|} \tag{4.2}$$

We refer here to the localization problem as **L**. In this problem, (disjoint) pairs of sensors are assigned to tasks, in order to minimize the sum of all tasks' uncertainty values; i.e., to minimize $\sum_j U_j$. (Note that σ is simply a scaling constant that without loss of generality we ignore by setting to 1.) We may also define the quality Q_j of a task's assignment as $Q_j = 1/U_j$, in which case we may give a dual formulation **L'** of the problem where the goal is to maximize the total assignment quality; i.e., to maximize $\sum_j Q_j$. With the objective thus defined, task j's quality Q_j is maximized (and its uncertainty U_j is minimized) when the separating angle is 90° and the distances are minimal.

In the expression above, $D_{1,j}$ and $D_{2,j}$ are the distances between the target and two sensors and $\theta = \theta_{1,j} - \theta_{2,j}$ is the smaller angle formed by the three points. Note that expression 4.2 incorporates two incentives. First, the sensors should ideally be as near to the target as possible; second, the ideal angle formed by the points is 90 degrees. If there happens to be a minimum separating distance between any sensor-target pair (say $\sqrt{1/2}$), then there is a minimum possible uncertainty value (in this case, 1).

The problem with the weights described is a special case of the NP-hard MAXIMUM 3D MATCHING problem [39]. In a weighted version of that problem, we are given finite element sets X, Y, and Z and weights for the triples in a set $T = X \times Y \times Z$, and the task is to choose a maximum-weight subset of T such that no element is used more than once.

4.4 Dynamic MSTA Problem Definition

We now generalize the two examples of non-additive utility MSTA problems into a more generic environment in which we do not know a priori which type of sensing tasks will need to be supported. Therefore, we need to allow for both additive and non-additive functions to be used in order to estimate utilities of groups of sensors (i.e. bundles) for each task. Note that in the static MSTA model in Chapter 3, sensors were individually allocated and the utility of a bundle was calculated by simply adding up the contributions of each single sensor allocated to a task. Therefore no benefit would have come from explicit inclusion of the "bundle" concept in the static model. Now instead, given that in the dynamic MSTA problem settings we allow for non-additive sensor utility functions, including explicitly the bundle construct within the model formulation gives the advantage of being able to associate non-linear utility values directly to groups of sensors. An alternative approach to including explicitly bundles in the model could have been to adopt the cyclic factor graph representation proposed in [33, 102], where bundles are implicitly represented in the formalization through the concept of factor. Our choice to explicitly include bundles in the model has also driven the architecture conceptualization in Section 4.5. Such architecture proposes to solve the dynamic MSTA model in two separate steps, where the first is substantially based on the concept of bundles and in particular on the bundle utility computation. This eventually led to the novel distributed allocation mechanism in Chapter 5.

We first model this problem in a static setting and then we extend this to a dynamic setting, by adding the time dimension in the problem formulation. We consider the same network model in Section 4.2 with the variant that we now allow for tasks to arrive and depart overtime and have different utility demands which need to be met by the sensor bundle in order to satisfy a task. Unlike the MSTA problem presented in Chapter 3 we do not consider a threshold on the utility demand, we in fact assume that if the utility demand is not met then the task remains unsatisfied.

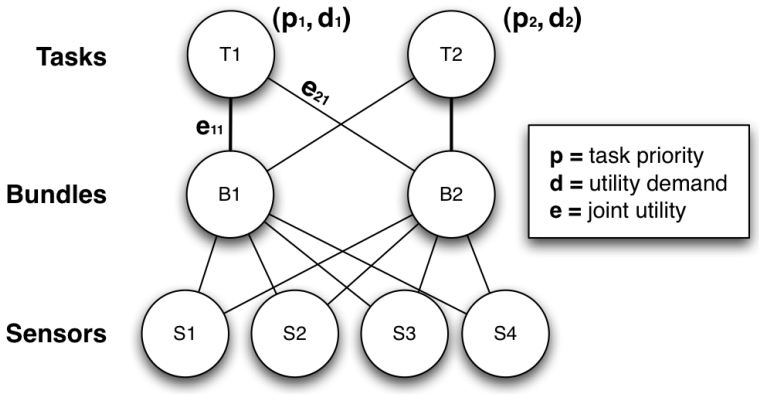

Figure 4.1: Dynamic MSTA problem as a tripartite graph.

Given that tasks can vary in importance, in a dynamic setting we allow for a sensor to be reassigned from a task with lower profit (which represents importance) to a task with a higher profit. However, since some tasks are more sensitive to interruption in service, preemption should be limited to tasks that can tolerate such interruption. For example, localization is very sensitive to interruption whereas event detection is less so. In the following problem formulation we allow for this behaviour by forcing a large reassignment cost in case resources cannot be preempted from an ongoing task.

4.4.1 Dynamic MSTA Problem Model

Our dynamic MSTA problem can be modelled as a tripartite graph whose vertices consist of a set of sensors $\mathbf{S} = \{S_1, \ldots, S_n\}$, a set of bundles $\mathbf{B} = \{B_1, \ldots, B_l\}$ and a set of tasks $\mathbf{T} = \{T_1, \ldots, T_m\}$ as represented in Fig. 4.1.

Each task T_j is associated with p_j, representing the priority and/or importance of the task. For each task T_j, we are given a set of sensor bundles, each of which would at least minimally satisfy the task's utility demand d_j. Each possible assignment of a bundle B_k to a task T_j is associated with a joint utility value e_{kj}, which is an estimate of how well a bundle of sensors is able to satisfy the sensing requirements of a task. We make no assumptions on the utility function: it could be for instance subadditive,

superadditive or linear. We assume that each of the bundle utilities would at least minimally satisfy the task's utility demand; i.e., $e_{kj} \geq d_j$. Instead, if bundle B_k has a joint utility smaller than the demand or it cannot serve task T_j at all, we set $e_{kj} = -1$. We describe more in detail how these values are computed in Section 4.4.2.

We define the *profit* as the utility e_{kj} of a sensor bundle B_k allocated to the task T_j scaled by the priority p_j. Therefore the goal in this problem is to maximize the profits of the satisfied tasks, subject to the constraint that no sensor or task is used more than once. This problem is essentially the NP-hard SET PACKING problem [39], and an offline[1] version of it can be formulated as the following integer program (IP):

$$\textbf{\textit{Maximize:}} \quad \sum_{k=1}^{l} \sum_{j=1}^{m} p_j e_{kj} \cdot y_{kj}$$

$$\textbf{\textit{Such that:}} \quad \sum_{k=1}^{l} y_{kj} \leq 1, \text{ for all } T_j \in \mathbf{T},$$

$$\sum_{k=1}^{l} \sum_{j=1}^{m} I_{ik} y_{kj} \leq 1, \text{ for all } S_i \in \mathbf{S}$$

$$y_{kj} \in \{0, 1\}, \text{ for all } (B_k, T_j) \in \mathbf{B} \times \mathbf{T}$$

The decision variables y_{kj} indicate whether bundle B_k is assigned to task T_j. The first set of constraints prevents more than one bundle from being assigned to any one task. The second set of constraints prevents any sensor from being used more than once, in multiple chosen bundles. Matrix I specifies the membership relationship between sensors and bundles; i.e., $I_{ik} = 1$ iff sensor S_i appears in bundle B_k. We assume that we are given this matrix as an input to the problem, this could for example be set using knowledge of which sensor type is compatible with which task. In Chapter 5, we analyze a method to compute both the elements of matrix I and the e_{kj} joint utility values, using knowledge based metrics combined with numerical utility functions.

Here our main interest lies in a *dynamic* (i.e., online) scenario. We include the time dimension in the previous static IP formulation, in order to model tasks arriving over

[1]Note that with "offline" we refer to the static settings, where all the tasks arrive at the same time.

time and having different durations. We adopt a discrete model of time in which we have discrete timeslots t. Tasks might arrive at the start of any timeslot and may last for any discrete duration. We assume that the *profit* for a task that lasts for multiple timeslots is the sum of the profits earned over all timeslots during the task's lifetime. In such a dynamic scenario, we consider the possibility of *preempting* sensors already serving other tasks in the case these might be more useful for satisfying newly arrived higher priority tasks. Of course we want to avoid preempting sensors too often from ongoing tasks otherwise the allocation strategy could interrupt the support of a task too frequently, quite literally making sensors undecided as to which task they should contribute. We express this by including in the objective function a cost $c_{ijj'}$ subtracted from the profits for each individual preemption event; i.e., a sensor assigned to an ongoing task T_j at time $t-1$ which at time t is reassigned to a different task $T_{j'}$. Thus the goal in this problem is to maximize the profits of the tasks and minimize the cost of the sensor-task allocation over all the timesteps. We formalize the dynamic MSTA problem as the following IP:

Maximize: $\sum_t(\sum_{kj} p_j e_{kj} y_{kjt} - \sum_{ijj'} c_{ijj'} z_{ijj't})$

Such that: $\sum_k y_{kjt} \leq 1$, for all $T_j \in \mathbf{T}, t \geq 0$

$\sum_{kj} I_{ik} y_{kjt} \leq 1$, for all $S_i \in \mathbf{S}, t \geq 0$

$y_{kjt} \leq C_{jt}$, for all $B_k \in \mathbf{B}, T_j \in \mathbf{T}, t \geq 0$

$z_{ijj't} \geq y_{k'j't} + y_{k,j,t-1} - 1,$

for all $S_i \in \mathbf{S}, t > 1, T_j \neq T_{j'}, B_k \wedge B_{k'} \in \mathbf{B}$

such that $I_{ik} = I_{ik'} = C_{j,t-1} = C_{jt} = 1$

$y_{kjt}, z_{ijj't} \in \{0, 1\}$ for all $T_j \wedge T_{j'} \in \mathbf{T}, B_k \in \mathbf{B}, t \geq 0$

Similar to the offline formulation above, the decision variables y_{kjt} indicate whether bundle B_k is assigned to task T_j at time t. The first set of constraints prevents more than

one bundle from being assigned to any one task at each timestep. The second set of constraints prevents any sensor from being used more than once at the same timestep, in multiple chosen bundles. We introduce a third set of constraints which prevents a bundle from being assigned to an inactive task; for this we define a matrix C as $C_{jt} = 1$ if task T_j is active at time t and 0 otherwise. Additionally, we use a decision variable $z_{ijj't}$ which is 1 if sensor S_i was assigned (within some bundle B_k) to one ongoing task T_j at time $t-1$ but is then reassigned (within some bundle $B_{k'}$) to another task $T_{j'}$ at time t. The fourth set of constraints forces $z_{ijj't}$ to be 1 if both $y_{k',j',t}$ and $y_{k,j,t-1}$ are equal to 1. These constraints apply only if sensor S_i appears in bundles B_k and $B_{k'}$ (i.e. $I_{ik} = I_{ik'} = 1$), and task T_j is active at both times $t-1$ and t (i.e. $C_{j,t-1} = C_{jt} = 1$). Note that no reassignment cost is charged when a sensor switches from a task that is ending.

As a generalization of the static MSTA problem in Chapter 3, the offline version of this dynamic problem is again NP-hard. For small instances of the dynamic MSTA problem, optimal solutions can be obtained by solving this IP; given the problem hardness, larger instances require to be solved with heuristics, which in our case are implemented by the distributed system architecture in Chapter 5.

4.4.2 Computing Sensor Bundle Utilities (e_{kj})

In the IP formulation for the dynamic MSTA problem, we do not explicitly include each task's demand d_j. The reason is that, differently from Chapter 3, we assume that all the joint utilities e_{kj} are already expressed with respect to each task's utility demand. We formalize the utility e_{kj} with the following expression:

$$e_{kj} = \begin{cases} u_{kj}, & \text{if } u_{kj} \geq d_j \\ -1, & \text{otherwise} \end{cases} \tag{4.3}$$

It is clear that we cannot compare directly raw utilities computed with different func-

tions, as these usually have different meanings. For example, one might represent a probability of detection and another a localization uncertainty, based on the task being served and the types of sensors. Therefore we need to normalize those values (between 0 and 1) to represent the percentage of satisfaction of each task with a particular sensor bundle assigned. We call this normalized value u_{kj}, and since this represents the percentage of satisfaction for each task, this is also comparable among different task types.

The u_{kj} values are computed using a joint utility function compatible with a particular task type (e.g., event detection) and bundle type (e.g., acoustic arrays). The matching process which is necessary to find which utility function is compatible with which task type and bundle type is described in detail in Chapter 5. In particular in Section 5.3, we describe as an example two joint utility models to compute the u_{kj} bundle utilities for both event detection and 2D-localization tasks. We extract the core of the static MSTA problems described in Section 4.3 by restricting the evaluation of the utility to a group of sensors (i.e., sensor bundle B_k) for just a *single* task T_j, without taking into account competition. In fact, the issue of solving the competition among tasks requiring the same sensors is included and resolved in the IP formulation of the dynamic MSTA problem.

Each of the bundle utilities e_{kj} ranges between d_j and 1 if bundle B_k *can* serve task T_j (with $d_j \in [0, 1]$); i.e., the sensor type or the combination of sensor types in the bundle can potentially satisfy the information requirements of that task. If the bundle normalized utility u_{kj} is smaller than the task's demand d_j or it cannot serve the task T_j, i.e. the combined sensor capabilities do not match the task requirements, then we set $e_{kj} = -1$. This is to avoid that the IP may assign a bundle to a task that does not need it, as the maximization optimization will force it not to use negative utilities.

By including the demand constraints in the calculation of the constant values e_{kj}, the IP problem formulation is easier to understand and provides a more flexible definition of e_{kj}. In fact, if necessary, we may remove the demand constraint completely or add more constraints in the calculation of the utility (such as a minimum threshold as in

Chapter 3). The alternative to this approach would have been to include the demand constraints directly in the IP. We can achieve this by defining a 0-1 matrix G where each element G_{kj} is 1 if and only if bundle B_k can serve task T_j with enough utility ($u_{kj} \geq d_j$). In addition it would be necessary to add a new set of constraints to the IP, so that each bundle-task pair respects the demand constraints, i.e. $y_{kjt} \leq G_{kj}$. It is clear from the alternative IP briefly outlined that this formulation would have deeply tied the problem to the demand of each task. Our approach instead provides more flexibility in dealing with the demand and an easier to understand IP formulation.

4.5 Conceptual Architecture

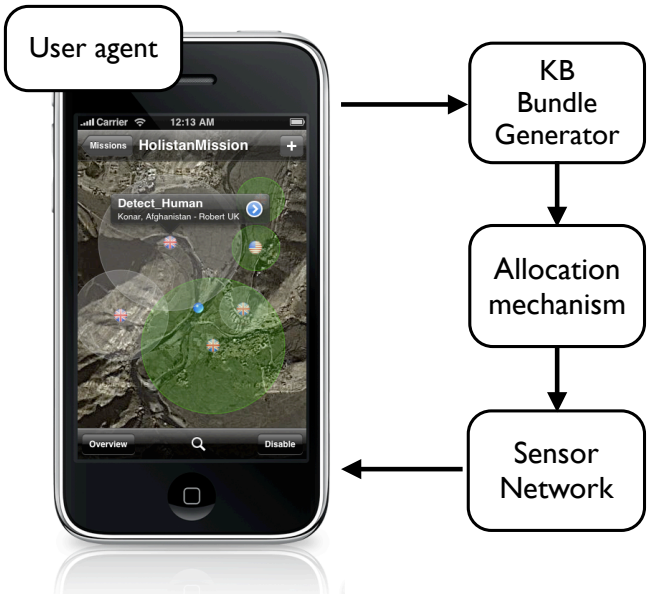

Figure 4.2: Conceptual architecture for the dynamic MSTA problem.

In this section, we give an overview of a conceptual architecture which highlights the steps necessary to find a solution to the dynamic MSTA problem. Such architecture is comprised of four main components as shown in Figure 4.2: a mobile user in the field (user agent), a Knowledge based (KB) bundle generator, an allocation mechanism, and the sensor network. The *user agent* represents the point of entry of tasks into the

system: it allows a mobile user to express their sensing requirements through a mobile app interface, defining the task location in the field and the type of task required (e.g. “localize vehicle”). The user also specifies its priority (p_j), together with the utility demand (d_j) and the required duration of the task. Note that we assume the users on the field are provided with mobile devices (such as smartphones, PDAs or tablets) to access the system, and that a task might end before or last longer than its expected duration.

The second component is the *Knowledge-based bundle generator* which recommends bundles of sensors that are known to be “fit-for-purpose" for each particular task specified by the user agent. In terms of the dynamic MSTA problem it basically computes the values of matrix I, generating all the possible bundles which could satisfy the requirements of the task and computing the sensor bundle utilities e_{kj}. This component uses *explicit* representations of sensing capabilities and task requirements, such as ontologies, which allows for an expressive and logically-sound way to represent knowledge and reason with it. This is similar to the work which addresses allocation in heterogeneous environments surveyed in Chapter 2 (Section 2.3.2). For example, [32] uses the ontology in [19] to explicitly represent the capabilities of robots, while instead [77, 126] use a knowledge-base implemented with a simple description language where they include both information about resource type and quantity required for each. Also our KB bundle generator uses both qualitative (i.e. knowledge-based) and quantitative (i.e. numerical utility based) measurements in order to generate bundles compatible with the task and evaluate their joint utilities. We describe in detail how we integrate both of these metrics in Chapter 5 (Section 5.3).

The third component is the *allocation mechanism* which finds a solution to the dynamic MSTA problem, considering the output of the KB bundle generator for each task generated. This consists of a set of coalitions of sensors that, if allocated to the task, would satisfy the sensing requirements demanded by the user (i.e. $e_{kj} \neq -1$). Note that a single sensor bundle recommended by the KB bundle generator is enough

to satisfy the sensing requirements of the task; therefore we will have a one-to-one relationship between sensor bundles and tasks, as in the dynamic MSTA model in the previous section. Finally, the *sensor network*, i.e. the last component of the conceptual architecture, is configured according to the output of the allocation mechanism, and begins serving the tasks by delivering sensor data to the user. Note that in our system, this last component has the only purpose of allowing each sensor to present itself to the other components, providing a description of its capabilities, physical attributes and current status (e.g. currently serving a task).

The conceptual architecture described above formalizes the steps adopted in many other works to achieve heterogeneous resource allocation. In Chapter 2 (Section 2.3.3), we in fact highlighted how it is common to split the allocation process in two parts (e.g. [32, 77]), by first considering the type(s) of resources required (i.e. qualitative matching), and then the actual allocation decisions on resource instances matching the type(s) required for each task (i.e. quantitative matching). The KB bundle generator and the allocation mechanism components basically implement these two steps which are necessary and sufficient to automatically allocate sensors to tasks in our heterogeneous environment.

For example, consider [32] where, first, an ontology-based reasoner is used to find which kinds of robots are able to accomplish a certain task. This component then provides such recommendations to a mechanism called "search for useful coalitions", which searches for groups of robots matching the "kind" recommended, able to potentially satisfy the task through collaboration. These two mechanisms are included in our *KB bundle generator* which similarly uses an explicit knowledge-based matching process (e.g. ontology-based) followed by the computation of all the possible sensor bundles matching the type recommended by the KB (i.e. implementing the "search for useful coalitions"). Moreover similarly to the "coalition instantiation algorithm" in [32], our conceptual architecture uses an *allocation mechanism* to solve the dynamic MSTA problem, sorting out the contention among different tasks with overlap-

ping requests (i.e. requesting the same sensing resources). The solutions proposed in [77, 126], also assume a component similar to the *KB bundle generator* which they call "task manager"; such component contains a knowledge-base of resource types compatible with each different task, including information about the cardinality of each resource. Both works then implement an *allocation mechanism* for finding a solution to the competing needs of different tasks with the aim to maximize the "welfare" of the set of tasks (one using a genetic algorithm and the other a combinatorial auction). For other examples we refer to Section 2.3.3, which highlights that our conceptual architecture includes the necessary and sufficient steps for solving our dynamic MSTA problem.

The novelty of our conceptual architecture consists mainly in describing explicitly a modular approach to solve the *ST-MS-IA-HE* dynamic problem step-by-step, formalizing what had been adopted in previous approaches for both MRTA and multi-agent coalition formation problems. Such architecture integrates, a knowledge based module with an allocation mechanism and includes the user agent in the mechanism as the point of entry of tasks in the system. These components offer flexibility in the automatic choice of sensors while also hiding the complexity of this choice from the user. Below we also describe how mobile smart-devices could be used to deploy components of such an architecture and, in particular, can allow a user to submit tasks to the system through a mobile app (i.e. implementing the user agent). This conceptual architecture could be implemented as either a centralized or a distributed system. In a centralized system we collect all the task requests in a central node such as a base station, while in a distributed approach we process the task requests distributedly. In the next sections, we consider the fundamental issues with a centralized approach in our dynamic environment; this pushes us in Chapter 5 to focus on the evaluation of a distributed implementation of our conceptual architecture.

Note that the *user agent* and the *sensor network* components are always "distributed": in fact, the first resides on the user mobile device where its only role is to allow users

to state sensing tasks and submit them to the system; while the second allows sensors (which are intrinsically distributed) to present their features and status to the allocation system. Below we therefore analyze the possible choices that we could make for the remaining two components, i.e. KB bundle generator and allocation mechanism. We describe two types of architectures, one where the *allocation mechanism* is *centralized* and one where it is *distributed*.

4.5.1 Centralized Allocation Mechanism

In many emergency response and military operations, we might assume the presence of one or more base stations collecting information about the environment, thereby having a global view of the ongoing tasks. Moreover the mobile devices used in the field are usually provided with cellular/GPRS/3G connection and therefore could use these technologies to remotely communicate with a base station. As a consequence we might be able to implement both the *KB bundle generator* and the *allocation mechanism* on a central server (as shown in Figure 4.3) and use mobile devices only to create new tasks and submit them to the system; i.e., each mobile device exclusively contains the user agent. The base station could then continuously execute a centralized online allocation algorithm receiving task requests over time and providing feasible allocation decision for each task, sorting out the competition for sensing resources with other tasks and maximizing the utility.

In [47], surveyed in Chapter 2, the authors showed that centralized bundling mechanisms in static settings usually deliver the best performance in terms of solution quality given their global view of tasks and resources. However in a dynamic scenario using a centralized allocation mechanism is unrealistic, first of all because information flow from the edges of the network to the central base station is prone to disruption. For example, if a certain area becomes “hot”, due to an explosion and/or a collapsed building, and suddenly many mobile users (military or emergency responders) occupy the same area, this often leads to an overload of the mobile telecommunication network

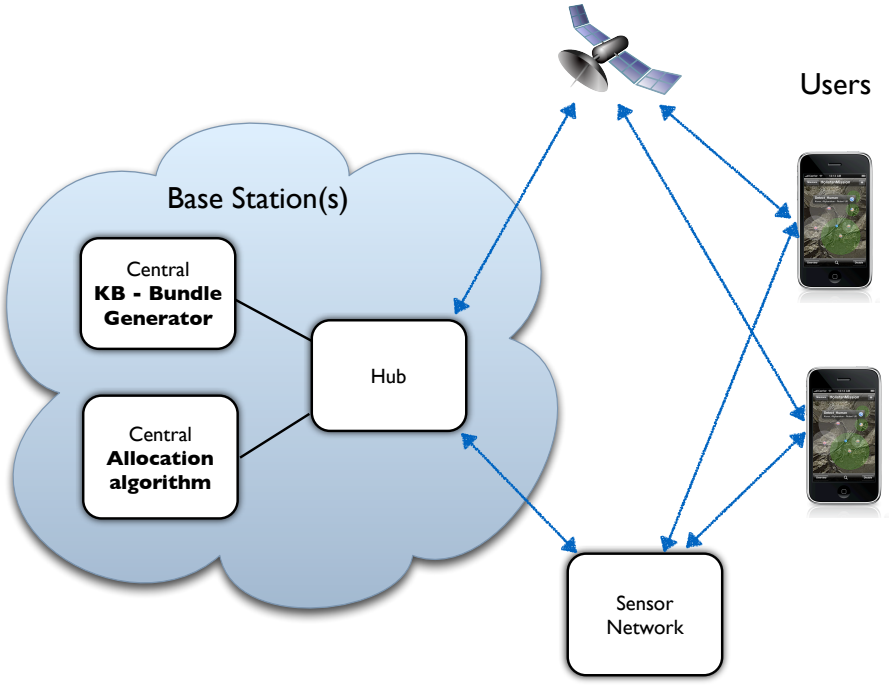

Figure 4.3: Centralized system architecture.

and therefore to a loss of connectivity with the base station. Given that we need to ensure support to sensing tasks requested by mobile users also when there is no satellite connectivity with the base station, as a backup in such situations we could redirect the communications to/from the central base station via the sensor network. However, this would reduce the lifetime and capacity of the sensor network and therefore it is not desirable.

Moreover, in a dynamic setting, a centralized system will not have a complete picture of what is happening at the edges of the network, considering that once the information arrives at the base station it might already provide an out-of-date picture of the field. Therefore, trying to accumulate data on task requests and sensor statuses in a central node in order to optimize the allocation decisions is flawed from the start. Finally, note that another option is to deploy the KB bundle generator on the user mobile devices together with the user agent, while still keeping the allocation mechanism centralized; however, also this variant of the centralized architecture would have the same issues discussed above.

4.5.2 Distributed Allocation Mechanism

Given that using a centralized allocation mechanism is unrealistic in our dynamic settings, in this section we briefly describe a distributed approach, shown in Figure 4.4, which we then implement and evaluate in Chapter 5. This system is able to work autonomously without the presence of a base station, thus addressing the main problem with the centralized approach. This architecture moves the *KB bundle generator* on the users' mobile devices and the *allocation mechanism* on both mobile devices and the sensors composing the network.

Note that the user devices communicate directly (e.g. using WiFi or Bluetooth) the requests for bundles generated by the *KB bundle generator* to the sensors composing the network; then the sensors are able to autonomously negotiate through a distributed allocation mechanism which is the best task to serve as part of a bundle and communicate back the decision directly to the user agents on the mobile devices. This addresses the main concern of the centralized architecture, which is that we can avoid collecting all of the information in a central base station by taking decisions using only local knowledge, thus always using an up-to-date picture of what is happening in the vicinity of each node.

Note that another option could have been to move the *KB bundle generator* to the sensors, leaving only the *user agent* on the mobile device. This architectural choice is not ideal given the possibly limited computational power and the valuable battery lifetime of sensors. Finally, the downside of the distributed architecture discussed above, compared to the centralized, is that it does not offer a global view of what is happening on the field. Although this might be desirable, e.g. for command and control purposes, it is not as important as providing support to task requests also in the case of loss of connectivity with the base station, and therefore it does not justify the use of a central allocation mechanism.

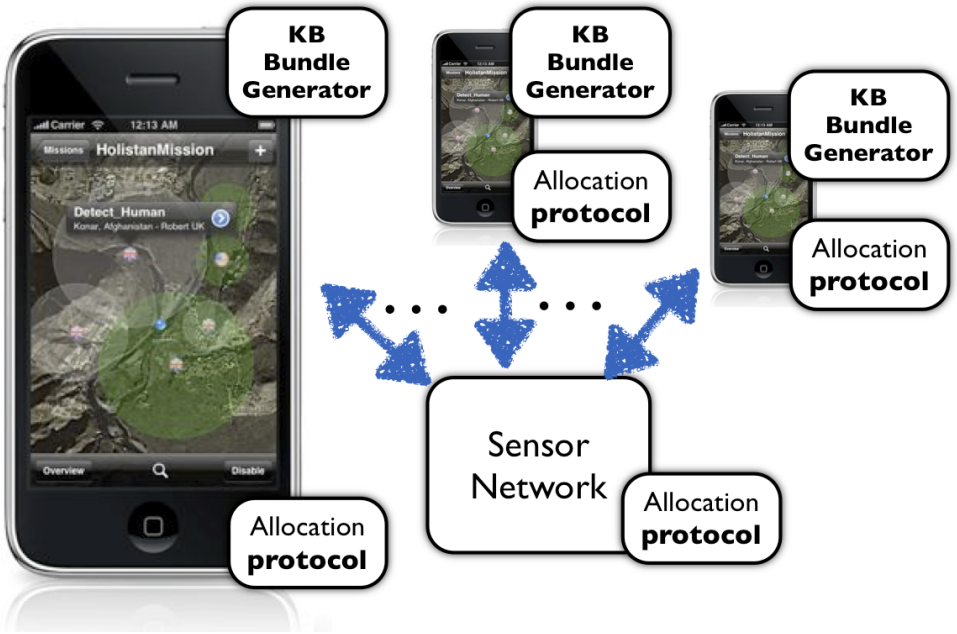

Figure 4.4: Fully distributed system architecture.

4.6 Conclusion

In this chapter, we presented a formal model for a dynamic MSTA problem (*ST-MS-IA-HE*) supporting tasks which might require non-additive functions to evaluate the utilities of sensors. We extended this problem to include a reallocation cost in order to minimize as much as possible the preemption of sensors from ongoing tasks. This model was developed by generalizing two specific MSTA static problems for event detection and 2D localization tasks, which we used to show the need to consider non-additive sensor utility models when dealing with task allocation in heterogeneous sensor networks. This led us to include support for both additive and non-additive utility functions in our more general dynamic MSTA problem formulation. In addition, in Chapter 5, we use the core of the two static problems, for event detection and target localization, as examples of non-linear sensor utility functions in order to evaluate our distributed solution.

As already mentioned in Chapter 1, the dynamic setting is a better fit for emergency

response scenarios which typically require to respond promptly to tasks and to react to changes in the environment. In addition allowing for non-additive sensor utilities in our model adapts well to the complex task sensing requirements and the many different sensor types which are available in such environments. For these reasons, we considered the problem of assigning groups of sensors (i.e. bundles) to tasks, allowing that different combination of sensors might provide different usefulness to each task. Moreover, each task type (e.g. detect vehicle) is allowed to use a different utility function to evaluate such groups of sensors, which makes the model flexible enough to be used with potentially any kind of tasks and sensors (although limited to *Single-Task* sensors).

On a side note, the two static MSTA problems formalized for event detection and 2D-localization tasks can be helpful also in developing ad-hoc solutions. In [107] we proposed a variety of algorithms for these specific problems[2]. From those algorithms and their evaluation, we note that formalizing MSTA problems in such restricted scenarios, where we know in advance that we need to provide support exclusively to a single type of task, can be used to develop ad-hoc solutions with good approximation results. However, modelling a sensor-task allocation problem for support to a single type of task necessarily ties us to that specific scenario, and the solutions developed for such problems are usable only in those environments. The dynamic MSTA problem formulation, instead, allows for potentially any task and sensor types. In fact, it allows for both additive and non-additive utility functions to be used to evaluate "how good" is a sensor bundle. Mechanisms designed to solve this more general problem can therefore better support scenarios in which there are multiple task types and heterogeneous sensors. It follows that mechanisms developed for solving the dynamic MSTA problem can be designed without apriori knowledge of which task and sensor types will appear in the field.

Finally, in the dynamic MSTA problem presented, one of the most challenging parts

[2] As explained in Chapter 1, these algorithms were not included in this dissertation because the core contribution of the thesis author is in the formulation of these problems.

is the actual computation of the inputs to the model; in particular, the joint utility e_{kj} of a sensor bundle and the computation of the membership matrix I, which defines the allowed bundle members. In fact, as observed already, sensing tasks usually require complex sensing capabilities which can be only provided by a combination of sensors, like, for example, an infrared camera working together with an acoustic array. Our model assumes that for each particular bundle we can compute these parameters. In the last part of this chapter, we proposed a conceptual architecture which is able to tackle the complexity of the dynamic MSTA problem, addressing separately the bundle formation and the actual allocation issues. The approach adopted by this architecture is similar to others in the literature, it is mainly composed of a knowledge-based component and an allocation mechanism. The first one integrates qualitative measurements (e.g. semantic based) by establishing which sensor types could potentially serve each task type, with qualitative measurements, such as both linear and non-linear utility functions to calculate the joint utility of feasible sensor bundles. The allocation mechanism then sorts out the competition among tasks requiring the same sensing resources. We discussed the pros and cons of choosing different strategies for implementing the conceptual architecture, in particular as a centralized and distributed system, highlighting how the distributed implementation is a more desirable choice.

Chapter 5

A Distributed System for Dynamic MSTA

In this chapter we propose a distributed system to solve the dynamic MSTA problem presented in Chapter 4. As already noted, we focus on a more realistic scenario compared to the one in Chapter 3 where we deal better with heterogeneous sensors, providing support to richer knowledge-based task requests expressed by emergency responders in the field.

We focus on the implementation of our conceptual architecture as a *distributed system*. To implement such a system, we develop a protocol (*allocation mechanism*) for both sensors and user devices by extending a pre-existing coalition formation mechanim [115]. Through this protocol sensors and user devices can autonomously negotiate which tasks should be served. We show that it is feasible to do part of the sensor-task matching on the user device by developing a knowledge-based component (*KB bundle generator*) as part of a prototype mobile app. The purpose of such a prototype is also to demonstrate how a user could interact directly with the sensor network, in particular submitting sensing tasks to our distributed system (*used agent*). Note that the "smart" devices are an integral part of our distributed architecture, thereby essential to find a solution to the dynamic MSTA problem.

To evaluate our distributed architecture, we first measure the performance of the knowledge based component on a real mobile device (an Apple iPod Touch 2nd generation)

implemented as part of our prototype mobile app which provides a user interface to the system. Then, we measure the performance of the allocation protocol by running simulations on randomly generated problem instances using the two non-linear utility models presented in Chapter 4 as a way to evaluate sensor bundles for two particular task types (i.e. event detection and 2D-localization). Our simulation results show that it is feasible to do part of the sensor-task matching on the mobile device and that our architecture is scalable, performing well in terms of overhead traffic and allocation quality when increasing the task arrival rate over time.

The main contribution of this chapter is the *distributed system* which implements our conceptual architecture, addressing the dynamic MSTA problem using both qualitative (i.e. knowledge based) and quantitative (i.e. numerical utility based) metrics. Its novelty consists in the interaction between mobile devices and sensor nodes formalized in both our distributed protocol and knowledge based component. We therefore answer to *Research Question 2* by developing a scalable solution for the MSTA problem (*ST-MS-IA-HE*) in a dynamic setting which maximizes sensor utilities and prioritizes the most important tasks, allowing for both additive and non-additive sensor utility models and richer knowledge-based requirements for each of the user generated tasks.

The remainder of this chapter is organized as follows. In Section 5.1 we analyze which features we need to add to the allocation protocol chosen as a skeleton for our architecture and discuss some related works. In Section 5.2 we give an overview of the distributed system based on the conceptual architecture solving the dynamic MSTA problem in Chapter 4. We describe the *Knowledge-based bundle generator* component of our system in Section 5.3 which supports the *allocation protocol* described in Section 5.4. In Section 5.5 we show the performance of our distributed system implemented in a discrete time event simulator. We also include the benchmarks of a prototype mobile app integrating the KB bundle generator, and showing the user interface which allows a user to submit tasks to the system (user agent). Finally, Section 5.6 concludes the chapter outlining some future work.

5.1 Background

The MSTA problem instance in this chapter is again identified by *ST-MS-IA-HE* and specifically is the dynamic MSTA problem formulated in Section 4.4. A version of the *ST-MS-IA-HE* is referred to as the *disjoint coalition formation* problem in the multi-agent community in [40] and it has been formally studied in [109] and [115]. A well known efficient allocation protocol has been proposed in [115] for the *static setting* in generic multi-agent systems. We therefore adapt it to deal with our problem, in particular by extending it with four main features to deal with a dynamic setting and to take advantage of the mobile user devices in the field.

5.1.1 Protocol Extensions

First, the original protocol assumed the presence of a central "matchmaker" agent to which the other agents would have asked if their capabilities were a fit for the task's requirements — we instead take advantage of the user's mobile devices which can host such a "matchmaker" which we call *Knowledge-based bundle generator*. We also completely move the calculation of the joint utility values of sensor bundles to the user devices, whereas in [115] it was carried out distributedly on the sensors with a preliminary protocol stage aimed at the calculation of the "coalitional value".

The second feature which we introduce is the ability for a task to request a different bundle of sensors in case the resources requested the first time were not available – we call this the *rebid mechanism*. There are in fact multiple ways of satisfying a task, as stated in [126] which considers alternative coalitions for satisfying a rescue task if the first attempt to get those resources fails. In general this extension is inspired by the concept of *substitute goods* widely used in economics [10].

The third extension consists of making more explicit the *preemption mechanism* which is mentioned in [115] when the authors describe the protocol implementation in the RETSINA agent system. We allow preemption of sensors which are already serving

a task, whereas the original protocol allows only preemption of sensors which have already committed to a task but have not already started to serve it. Our approach is similar to the multi-robot system architecture proposed in [18] which assigns tasks to robots with first-price auctions but allows (in some circumstances) later reassignment.

Finally, in a dynamic setting where each task lasts for a different time interval, providing support to the task too late might be useless for a user. We therefore extend the protocol by considering deadlines or *expiration time* before which we need to decide if we can serve the task. Although we consider this feature of the dynamic problem, we keep our focus on maximizing the overall user/task "happiness", without trying to reduce the tasks' waiting time. We achieve this by using the *rebid mechanism* mentioned above, which in the case of failure of the first request for sensors allows the protocol to request another bundle if there is still time to satisfy the task.

5.1.2 Architectures & Techniques

Various modular approaches and architectures have been proposed for allocating heterogeneous resource bundles to different tasks. These often adopt a similar approach to ours where they split the allocation process into two parts: resource-task capability fitting, followed by the actual evaluation of utility based metrics using information about the actual resource instances. Some examples include [81] which considers the problem of forming virtual organizations by aggregating multiple service providers, [77] proposing an economic based approach to solve sensor management and finally [32, 114] performing ontology-based allocation of heterogeneous robots/sensors to heterogeneous tasks. We refer to Chapter 2 for a more detailed overview of other architectures, we note that many of these architectures assume a central knowledge-based matchmaker to recommend which type of resources should be used. Our system differs from those as instead we deploy the matchmaker component on the user "smart" devices, making thus the system fully distributed.

Similarly many techniques have been proposed for coordination among different entities (agents) in order to form coalitions to achieve a common subgoal in a system. For example, the mediation algorithm [101] which allows agents to achieve a common goal while gradually revealing their information (e.g. utility). Such an algorithm considers the presence of a central mediator agent and therefore it is not fully distributed. Another example is provided by [100] which proposes a centralized heuristic to achieve coalition formation with spatial and temporal constraints. Finally, [106] provides a proposal-based algorithm for allocating directional sensors to tasks but assumes additive utilities. We chose to adapt the *disjoint coalition formation* protocol in [115] because of its fully distributed features, given that it does not assume a central mediator agent, and it is prone to be easily extensible to our dynamic settings. For a more extended discussion of different techniques and architectures we refer to Chapter 2.

5.1.3 Mobile Device & Simulation Environment

Recent work has addressed how mobile devices could be used as a repository of large quantities of semantic data, such as triple stores expressing relationships between objects, and how retrieving this data can be made efficient. In [127], for example, the authors propose a conference guide implemented as a mobile app on iPhone and iPod Touch. They show that by using the Hexastore indexing technique [128], it is feasible to efficiently query and store large amount of semantic data on mobile devices. Another example is provided by [25] which studies how much semantic data can be stored and processed on small devices (in their case the ASUS EEE PC 700 netbook), so that they could also evaluate how much of the data have to be processed remotely on a server. The authors developed a set of benchmarks to evaluate the performance of existing semantic web tools (e.g., query engines, triple stores and reasoners) in terms of disk space and query response. They argue that current semantic web tools are developed mainly for large-scale applications and that more work is needed to develop semantic software infrastructures dedicated to small-scale applications running with

limited hardware resources. Similarly to [127, 25], in our distributed architecture we store semantic data about which different set of sensor types can fit different task types. In order to efficiently store this data, in Section 5.3, we adopt a much more trivial approach by simply flattening the knowledge base into a look-up table, and we then study the performance of querying such data and storing it in the optimized database engine in iOS (SQLite). Note that it is out of the scope of this dissertation to develop efficient semantic web based techniques to store and query data on a mobile device, thus our choice of using a standard look-up table.

We use the REPAST Simphony agent-based simulation environment[1] to evaluate the distributed allocation protocol component of the architecture proposed in this chapter. This tool has been used in recent works [72, 111] to simulate protocols for wireless sensor networks and emergency response scenarios. For example, [72] describes a simulation platform specifically designed to support the development of sensor network middleware services. The authors use REPAST as a base for developing this platform, but unfortunately this tool was not publicly released. In Section 5.5, to evaluate our distributed allocation protocol, we implement a REPAST simulation whose design was largely inspired by [72]. Finally, note that we chose REPAST over other simulators commonly used in WSN research such as NS2[2] or TOSSIM [71], because REPAST provides a fast-development infrastructure for implementing and evaluating distributed collaborative behaviour of sensor nodes. As noted in [72], NS2 provides support for many low level communication protocols (e.g. TCP, routing, multicast) but lacks tools for fast implementation and evaluation of node collaboration protocols at the OSI application layer (which is our case); TOSSIM instead is focused on emulating low-level operations of TinyOS-based WSNs.

[1] http://repast.sourceforge.net/

[2] http://isi.edu/nsnam/ns/

5.2 Distributed System Overview

The distributed system moves the *KB bundle generator* and the *allocation mechanism* onto the users' mobile devices and the sensors comprising the network. This is based on the idea that nowadays the computational power and storage on a mobile device, such as a smartphone, is comparable to that of an average personal computer, albeit with more limited running time due to the limited battery life. It is therefore more convenient to deploy as many components as possible on those devices rather than on sensors, which might also be difficult to access and even more energy constrained.

Below we propose how we may adapt the protocol for the *disjoint coalition formation problem* [115] to solve our dynamic MSTA problem, through which the sensors are able to autonomously negotiate which is the best task to serve as part of a bundle. Note that this protocol was designed for generic multi-agent systems, so we need to adapt it in order to include the mobile device as a "task entry point" and as the only entity able to generate a list of sensor bundle candidates for each task. In addition, this protocol was designed for static settings while instead here we deal with a dynamic setting. The adapted protocol runs on the two main entities composing our distributed system: sensors and user devices. When the user creates a task (through the user agent implemented as the mobile app in Figure 5.1) the *KB bundle generator* on the same device then computes *feasible sensor bundles* (setting the values of matrix I) and their *joint utilities* (e_{kj}), we call these pairs *bids*. These are then sent to the sensors which, using the extended disjoint coalition formation protocol, choose greedily the task to serve based on the average contribution of utility per sensor until there are no more bids.

Thanks to the layered approach derived from the conceptual architecture in Chapter 4 which integrates a knowledge-based component with an allocation protocol, we provide flexibility in the choice of sensors to use in order to satisfy the users' task requirements. Providing a step-by-step solution to the dynamic MSTA problem, using both qualitative (knowledge based) and quantitative (numerical utility based) meas-

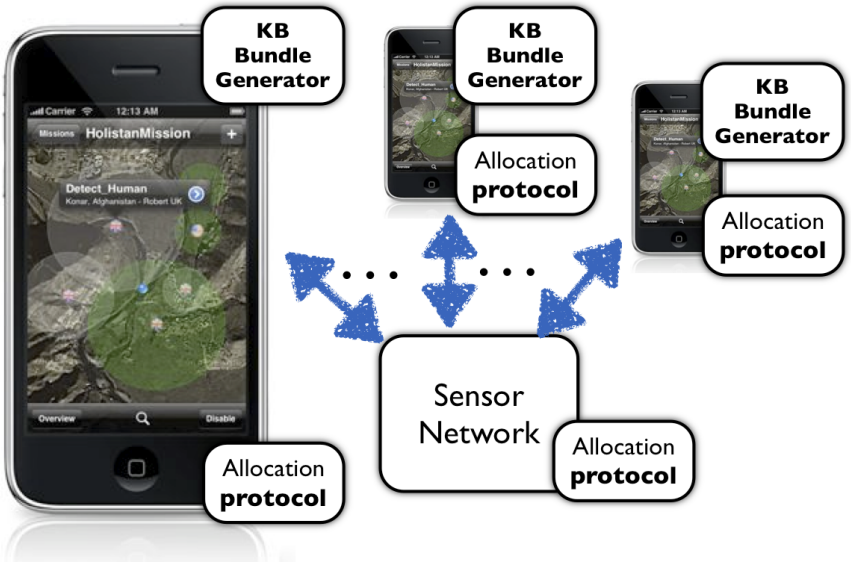

Figure 5.1: Fully distributed system architecture.

ures. In the next sections, we describe in detail the *Knowledge-based bundle generator* and the *allocation protocol*.

5.3 Knowledge-Based Bundle Generator

Task allocation in heterogeneous sensor networks requires knowledge of which sensor types are applicable to which kinds of task. We separate two issues: whether a type (or combination of types) of sensor *can* potentially satisfy a task, and *how well* might particular sensor instances perform on a given task. We encode this knowledge in a knowledge base (KB). The KB stores, for each kind of task, each type (or combination of types) of sensor that can theoretically achieve that task, and a *joint utility model* that allows us to compute the utility of particular sensor instances for that task. We refer to the types of applicable sensors as *bundle types* (because they determine the types of sensors that form our allocated bundles). These types may be defined using a *sensor ontology*, such as the one described in [44].

Figure 5.2 shows the reasoning process enabled by the KB in more detail. For each

newly created task T_j at time step t the KB recommends a Joint Utility Model (JUM) to compute the sensor bundle utility and a Bundle Type (BT) to select the sensor types compatible with the task. The identification of Bundle Types was originally implemented as described in [44], by dynamically matching the sensing capabilities provided by each single sensor and the capabilities required by each task type. Sensor and task features were defined using ontologies expressed with the Web Ontology Language (OWL); the matching process was implemented using the Pellet open source OWL reasoner[3]. The output of this step is a set of Bundle Types, where each of the entries is composed of a set of sensor types which all together would satisfy the information requirements of the sensing task. The BTs recommended at this stage are just sets of sensor types, and they do not contain any information regarding the number of instances to choose for each sensor type in the BT.

To determine the number of instances of each sensor type assigned to a task we use a *Joint Utility Model*. Such a model consists of a suggested maximum, minimum or exact number of sensor instances for each of the sensor types forming a BT (e.g., exactly two sensors for 2D localization tasks using acoustic arrays). Moreover it includes a utility function which is used to compute the estimated value e_{kj} for a group of sensor instances implementing the recommended BT. The KB indicates which JUMs are appropriate for each task type. Each JUM is only compatible with certain sensor types (or combinations of these), so the final step in the reasoning procedure is to match applicable JUMs with BTs.

To illustrate the above, consider two *task types*: "event detection" and "target localization". As result of the first reasoning stage the reasoner suggests to use $BT_1 = \{VideoCamera, AcousticArray\}$ for "event detection", i.e. we can use both video cameras and acoustic sensors; instead $BT_2 = \{AcousticArray\}$ for "target localization". Later on it associates one or more JUMs to each recommended BT by searching the knowledge base. In this case we assume that in the knowledge base there are

[3]http://pellet.owldl.com/

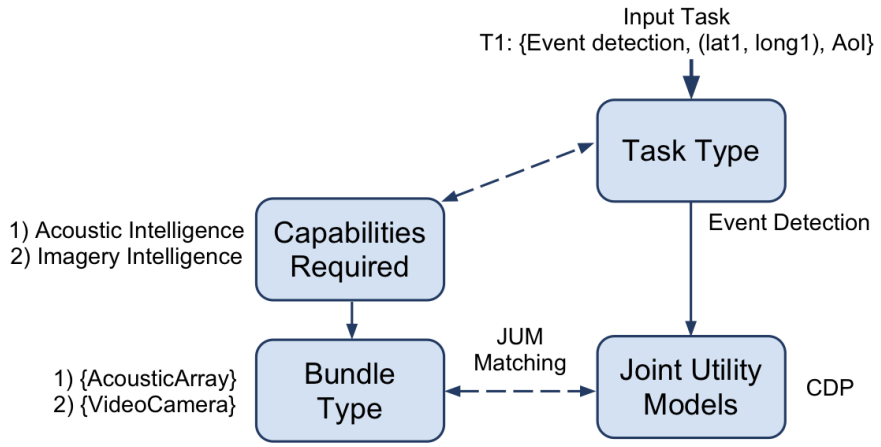

Figure 5.2: Reasoning procedure.

just two utility models compatible with either of the two BTs. Based on the models proposed in Section 4.3 we associate to the *event detection* task a variant of the Cumulative Detection Probability model (which we call *CDP*) and to the *target localization* the 2D-localization function (referred to as *2D-Loc*) based on the distance and angle from the target. Therefore for the *event detection* task the reasoner will recommend ($\{VideoCamera, AcousticArray\}$, *CDP*); instead for *target localization* it will output ($\{AcousticArray\}$, *2D-Loc*). Note that this allows us to be very flexible in terms of allocation as the same task could be satisfied using different combinations of BTs and JUMs, and therefore would increase the chances of satisfying that task.

Below we list the JUMs mentioned above which will be also used in Section 5.5 for evaluation of the allocation protocol. Note that these functions have been chosen as examples and are extensively discussed in Section 4.3. In specific, we extract the core of the static MSTA problems described in that section by restricting the evaluation of the utility to a group of sensors (i.e., sensor bundle B_k) for just a *single* task T_j, without taking into account competition. In fact, the issue of solving the competition among tasks requiring the same sensors is included and resolved in the IP formulation of the dynamic MSTA problem which is implemented by the allocation protocol in Section 5.4:

CDP Given a set of candidates of size l for a *single task*, this model chooses the k sensors maximizing $(1 - \prod_{S_i \to T_j}(1 - e_{ij}))$, where e_{ij} is the probability that a single sensor S_i detects an event for T_j. Here the joint utility is monotonically increasing as sensors are assigned but non-linear. Note that in Section 5.5 for our simulations we will choose $k = 10$, and as an example of detection probability e_{ij} we use the same formula described in Chapter 3 which follows typical signal attenuation models in which signal strength decays inversely with distance squared: $e_{ij} = \frac{1}{1+D_{ij}^2/c}$ if $D_{ij} \leq R_s$ (with $c = 60$) and 0 otherwise.

2D-Loc Given a set of candidates of size l for a *single task*, this model chooses the best *pair of sensors* which maximize $1/U_{i,i'}$, where U represents the uncertainty of the target localization provided by the pair $(S_i, S_{i'})$. The uncertainty is given by $U_{i,i'} = \frac{\sqrt{D_i^2+D_{i'}^2}}{|\sin(\theta_i-\theta_{i'})|}$; where D_i and θ_i are the distance and bearing, respectively, from the task location to the i-th sensor. This function is maximized when the angle separating the sensors is $90°$ and the distances are minimal. It is clear that also in this case the joint utility is non-linear.

Note that, as noted in Section 4.4.2, we cannot directly compare utilities computed with the first and with the second JUM because the objective function values have different meanings: CDP computes the cumulative detection probability, and 2D-Loc the opposite of the uncertainty of the target localization. Therefore we need to normalize those values (e.g. between 0 and 1) to obtain the percentage of satisfaction of the task with a particular utility value generated with one of the two models.

5.3.1 Lightweight KB Bundle Generator

The original implementation of the reasoning process is computationally expensive [44] due to the exponential-time complexity of the classification algorithm used by the Pellet reasoner. However, because the task types and sensor types are relatively stable (it is rare for new kinds of sensor or task to become available during an operation) it is

Task Type	Recommendation
1	(BT_1 + JUM_1)
1	(BT_2 + JUM_2)
2	(BT_3 + JUM_1)
2	(BT_2 + JUM_1)
2	(BT_2 + JUM_2)
...	...
N	

Figure 5.3: Lightweight KB bundle generator.

feasible to pre-compute the results of the reasoner and store these in a look up table; such an implementation is more suitable for deployment on a mobile device, to avoid wasting battery life on expensive computation. The only assumption that this approach makes is that the device will have a sufficiently large storage capacity, which is reasonable for modern mobile devices. The structure of the look up table is represented in Figure 5.3. Note that each row of the table is a tuple composed of a Task Type (e.g., represented as an ID), a Bundle Type and a JUM.

5.4 Allocation Protocol

In this section we describe how we have extended the *disjoint coalition formation protocol* in [115] to solve the MSTA problem instance. The protocol runs on sensors and user devices. When a task is created by a user, the *KB bundle generator* computes the *feasible sensor bundles* and sends the request to the involved sensors. These, then, greedly decide which task to serve using the *allocation protocol*.

The protocol, therefore, consists of two main stages: in the preliminary stage – which we call *initial negotiation* – the user devices compute and distribute the bids to the sensors by first discovering which sensors are good candidates for each task; in the main stage — called *bundle formation* — the sensors decide upon which bundle to join

in order to serve a particular sensing task. The *initial negotiation*, which we illustrate in Figure 5.4 using an example, works as follows:

1. At time t the users create tasks on their devices, characterized by a task type, a priority, an expected duration, a deadline to be satisfied, a utility demand and a geographical area of interest — representing the entire geographical area on which the sensing task will take place. E.g., "event detection" at $(lat_1, long_1)$ within a circular area of radius 50 meters.

2. The user devices query the sensors in the geographical area of interest, asking for their location, sensor type and current status (unallocated or already allocated to another task).

3. The user devices then generate bids of the type $bid_\alpha = \langle U_\alpha : (T_j, B_k, v_{\alpha,t})\rangle$, where U_α is a user, T_j is a task generated by U_α, and B_k is the bundle of sensors with highest value $v_{\alpha,t}$ computed as:

$$v_{\alpha,t}(T_j, B_k) = \begin{cases} p_j \cdot e_{kj} - c_{kj,t} & \text{if } e_{kj} \geq d_j \text{ ,} \\ 0 & \text{otherwise} \end{cases}$$

   where e_{kj} is calculated by the *Knowledge-based bundle generator* using the recommended Joint Utility Model and considering only the sensor instances of the types specified in the Bundle Type which is instantiated by B_k. Also note that we subtract from the profit the cost $c_{kj,t}$ proportional to the number of sensors that are already tasked and that we would need to preempt from other tasks at time t, as discussed in Section 4.4; i.e., $c_{kj,t} = \sum_{ijj'} c_{ijj'} \cdot z_{ijj't}$ with $S_i \in B_k$. We calculate the cost $c_{ijj'}$ of switching each sensor by halving the average contribution of sensor utility to the new task scaled by its priority; i.e., $c_{ijj'} = p_j \frac{e_{kj}}{2|B_k|}$.

4. Finally the user devices send to all the sensors included in the bundle B_k the bid bid_α.

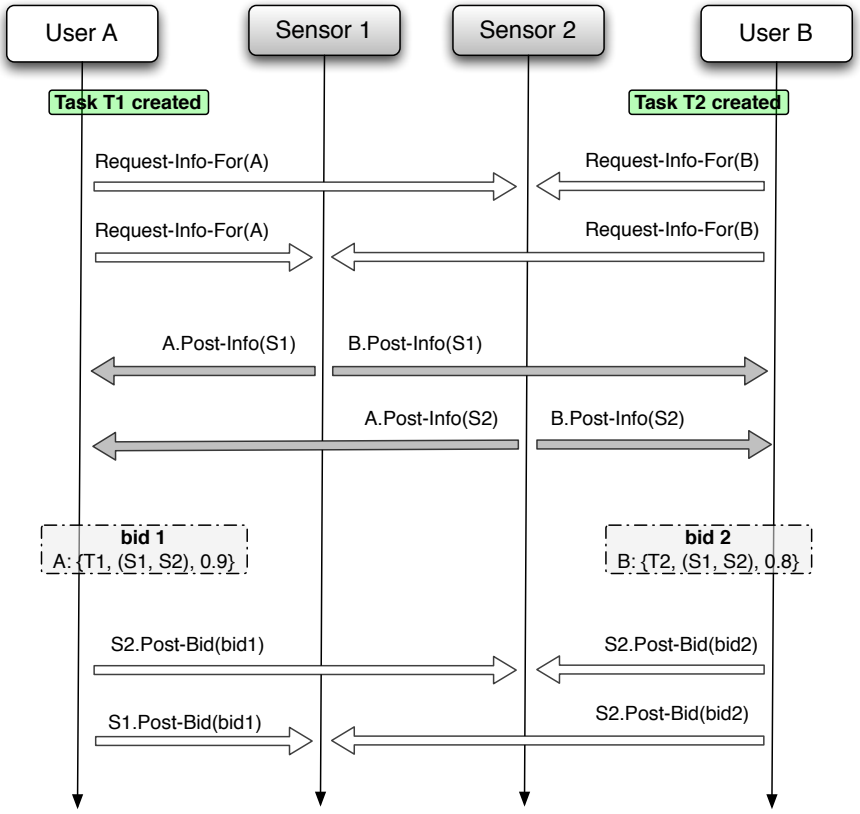

Figure 5.4: Example for the *initial negotiation* protocol.

In the protocol we do not specify any particular discovery mechanism. In the simulations instead we use a very simple device discovery policy based on the communication range of sensors and users. We do not consider the overhead generated by the discovery process; i.e., step (2) of the initial negotiation. This will depend highly on the discovery mechanism which will be implemented on the nodes, e.g. [65], therefore we leave this choice to the reader.

The main stage of the protocol is represented by the *bundle formation* part which mainly resides on the sensors, and for which we provide an example in Figure 5.5:

1. Each sensor node keeps a list of bids in which it is involved sorted by decreasing average value $u_{ij,t} = v_{\alpha,t}/|B_k|$ where $|B_k|$ is the cardinality of bundle k (which follows the heuristic suggested in [115]) .

2. If the sensor is currently not serving any task, it chooses the bid to which it can contribute the most, i.e. with the highest $u_{ij,t}$. It then sends an ACCEPT message to the sensors that are present in this bid (i.e. to all $S_i \in B_k$).

3. The sensor clears this bid (i.e. it officially commits to that task) only when it receives an ACCEPT message from all the sensors involved in the bid. This ensures that a bid is cleared if and only if all the sensors in the bid agree to clear it.

4. When a sensor node clears a bid, it sends a CLEARED to all its neighbors — the set of sensors with which it shares some bids and the users who generated those — notifying them that it has cleared (i.e. it was allocated to task T_j) and all bids including that sensor should be dropped from consideration.

5. The sensors that receive a CLEARED message from another sensor, delete from their bid list all the bids in which the sender sensor is included. The sensors stop the execution when they clear a bid, when their list of bids is empty or when an ACCEPT *timeout* for receiving all the ACCEPTs from the sensors in B_k is expired. If a sensor remains unallocated, it sleeps until it receives the next bid starting again from step 1.

6. The users that receive a CLEARED message from a sensor assigned to another user's task, delete the sensor from the set of neighbor sensors and recompute a new bid with the remaining sensors through the *KB bundle generator*. The users stop execution when they obtain a CLEARED message from a sensor assigned to one of their tasks or when a *convergence timeout* for satisfying the created task is expired since the beginning of the negotiation.

Note that if a task is not satisfied, the user device generates a new bid asking for an alternative sensor bundle to the KB bundle generator until a *convergence timeout* to satisfy the task expires (Step 6). This extends the original protocol in [115] with a *rebid mechanism* including an *expiration time* for the task, as discussed in Section

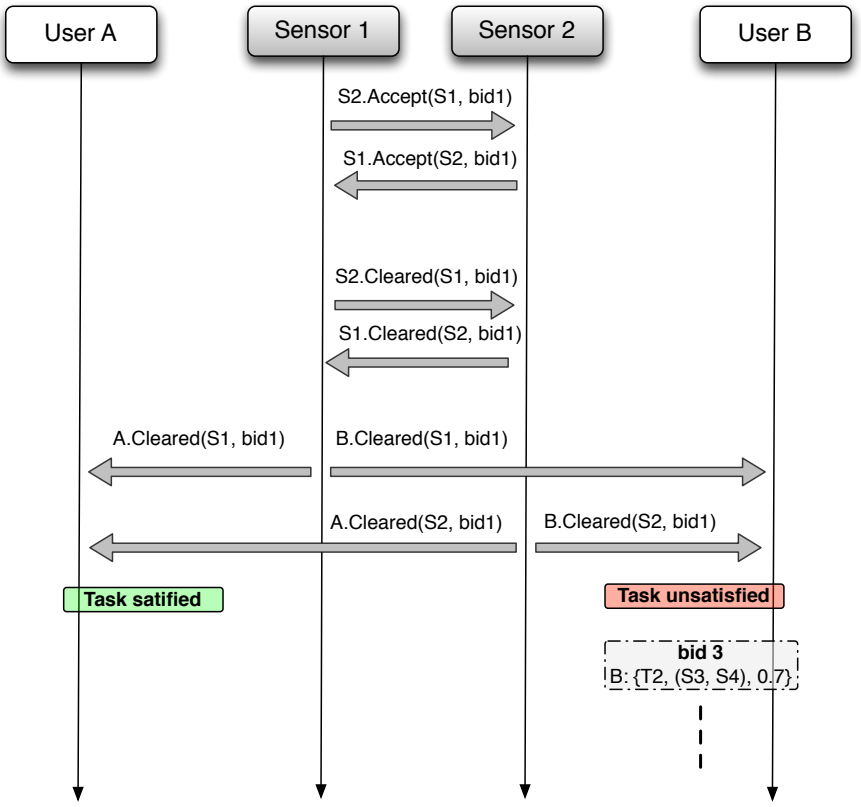

Figure 5.5: Example for the *bundle formation* protocol

5.1. We also introduced an ACCEPT *timeout* which expires if sensors have waited too long for an ACCEPT, allowing the user to rebid before the task's *convergence timeout* expires. In fact a sensor might be stuck in the "waiting-for-accepts" state if at least one sensor involved in the bid had previously committed to another task (sending a CLEARED before the arrival of the new task). This previously allocated sensor might not be freed by its current task before the expiration of the newly generated task's convergence timeout. Therefore the other sensors which accepted the new task would be stuck in a waiting for accepts state. The ACCEPT *timeout* allows the sensors waiting for an ACCEPT to drop the bid in case of a long wait and start from Step 1; allowing the user device to generate a new bid for the task thus improving the chances to satisfy the task thanks to the use of the *rebid mechanism*.

5.4.1 Preemption Mechanism

The bid valuation function $v_{\alpha,t}$ already considers the cost of reassigning sensors from an ongoing task to a newly arrived one. Preempting sensors is important to increase the likelihood of satisfying higher priority tasks appearing overtime as discussed in Section 5.1. We describe the *preemption* steps that we add after Step (2) of the *bundle formation* stage, as follows:

2.a When a sensor receives a $bid_{\alpha'}$ and is *currently allocated*, it will only be preempted from the current task j — and therefore it will send an ACCEPT — if the average contribution to the new task j' is strictly greater than the previous one; i.e., $u_{ij',t} > u_{ij,t}$ (where $u_{ij,t} = v_{\alpha,t}/|B_k|$).

2.b When a sensor is *preempted*, it sends a PREEMPTED message to all the sensors participating in the current bundle and to the user device owning the task. The sensors receiving this message will stop serving the current task, and the user device will drop it (setting the task to inactive).

2.c If instead a sensor is not preempted (i.e. $u_{ij',t} \leq u_{ij,t}$) it will keep serving the task for all its duration, after which all the sensing resources assigned to the task will be released.

5.4.2 Computational complexity

As proved in [115] the original disjoint coalition formation protocol has a computational complexity of order $O\left(n^h \cdot |\mathbf{T}|\right)$ to arrive at a decision where n is the number of sensors in the network, h is the maximum size allowed for a sensor bundle and $|\mathbf{T}|$ is the number of tasks (which in our model is equivalent to m). In the protocol extension presented in this section, the computational complexity is even better due to the sparseness of the problem instance and to the dynamic settings where tasks arrive over time.

In more detail, let us briefly consider the number of computations required in the two main stages of the algorithm, and for each of those consider the operations performed by both user and sensor agents. In the first stage (i.e. *initial negotiation*) a user agent needs to compute all the possible sensor bundles from the set of sensor candidates on the field for each task generated. If we assume that in the worst case all sensors in the field might potentially be sensor candidates for a task, then each user agent performs $\sum_{i=1}^{h} \binom{n}{i}$ operations in order to evaluate all the possible bundles; as noted in [115] this summation is $O\left(n^h\right)$. Given that these operations needs to be performed for each task, in total the set of user agents delivers a computational complexity of $O\left(n^h \cdot m\right)$. Due to the sparseness of our problem instance, the cardinality of the set of sensor candidates on the field is much lower in practice. In fact, if we assume that the number of candidates is for example twice the maximum size of the sensor bundle (h), the computations required are $\sum_{i=1}^{h} \binom{2h}{i}$ which is $O\left(h^h\right)$. Note that in our scenario usually $h << n$, because there is a limited number of sensors which can contribute to each task due to location and sensing range constraints. Therefore for the set of user agents we will have a computational complexity of $O\left(h^h \cdot m\right)$. Considering our dynamic scenario where tasks arrive over time, the number of operations per timestep is in the order of $O\left(h^h \cdot \frac{m}{t_{max}}\right)$ with t_{max} representing the entire duration of the operation on the field. Sensor agents at this stage do not require to perform any computation and therefore they do not count towards the total computational complexity of the *initial negotiation* protocol stage.

In the second stage (i.e. *bundle formation*), each sensor keeps a list of bids and selects the bid to which it can contribute the most (i.e. with the highest average utility for the most profitable task). In order to find this bid the sensor needs to perform $O\left(m\right)$ comparisons, basically finding the maximum profitable bid to join in the list. Considering the dynamic arrival of tasks in the field, as before, this operation yields a computational complexity of $O\left(\frac{m}{t_{max}}\right)$ per sensor in the field, which therefore is in total of the order of $O\left(n \cdot \frac{m}{t_{max}}\right)$. Finally, if we assume as in [115] that each assignment operation per sensor is $O\left(1\right)$, we have $O\left(h\right)$ assignment operations for each task. Therefore in total

the computational complexity is of the order of $O(h \cdot m)$ as, potentially, each task generated on the field might be satisfied by allocating it a bundle. Considering again the dynamic aspect of our problem instance, the computational complexity in this case is $O\left(h \cdot \frac{m}{t_{max}}\right)$. If we take into account also preemption, then each sensor might need to perform in addition a comparison between its own current utility contribution to an ongoing task with its contribution to the newly generated task. Therefore each sensor performs in the worst case $m-1$ comparisons with other bids (i.e. $O(m)$), which leads to a complexity of $O(n \cdot m)$. Although in practice it is more likely to be $O(n)$, as it is rare for a sensor to receive more than one preemption request at the same time due to the sparseness and dynamic nature of our scenario. User agents in this *bundle formation* stage might have to rebid for a task which was unsuccessful, as noted above this has a theoretical computational complexity of $O\left(n^h \cdot m\right)$ which in practice becomes $O\left(h^h \cdot \frac{m}{t_{max}}\right)$.

In conclusion, both the set of user and sensor agents yield for the protocol a theoretical computational complexity of $O\left(n^h \cdot m\right)$, which due to the sparseness and dynamic settings of our scenario becomes in practice of the order of $O\left(h^h \cdot \frac{m}{t_{max}}\right)$. Therefore, the protocol presented in this section has effectively a better computational complexity than the one discussed for the original protocol in [115], which considers a generic multi-agent system scenario.

5.5 Implementation and Performance

In this section we describe the implementation and performance of our fully distributed system. We include benchmarks of the *KB bundle generator* which we implemented on an Apple iPod Touch. We also show the performance of the proposed *allocation protocol* implemented in the REPAST Symphony[4] Java simulation environment.

[4]http://repast.sourceforge.net/ - checked 10th December 2011.

5.5.1 Mobile Device

As mentioned in previous sections, we implement a prototype app on an Apple iPod Touch 2nd Generation device[5], which represents the user agent in our architecture allowing a user to state their tasks and submit them to the system. A screenshot of the user interface is reported in Figure 5.6, which allows the user to set the task type, task priority and expected duration of the task. We also allow to define the radius of the area of interest of the task, the user is then allowed to move the task to the desired location on the map by dragging the center of the area of interest defined. Finally the task's utility demand is randomly generated by the prototype, but we could allow a user to determine the degree of task satisfaction desired. Highly positive feedback was obtained through various demos to US Department of Defence, UK Ministry of Defense, and NATO senior members and surveillance/reconnaissance specialists. Especially, it was appreciated the usability and simplicity of the mobile version in terms of how one can request complex information through a simple task creation form. Moreover it was appreciated that low level details of both sensors and tasks (e.g. particular sensor capabilities required to satisfy a task) can remain hidden to a user – thanks to the KB bundle generator – and therefore making the system accessible also to non-expert users.

In addition, we tested if it was feasible to implement the *Lightweight KB bundle generator* on a modern mobile device, given that the lookup table size could be very large and therefore computationally expensive to use. We implement the table showed in Figure 5.3 as a combination of four tables using the standard ER concept of *foreign keys*. We implement this ER schema using SQLite which is the integrated database engine in iOS (the OS installed in the Apple iPod Touch), and as a consequence the measured performance will be related to the chosen database engine. We run experiments first using a *Synthetic KB* filling the tables with synthetically generated data. Then we use a *Prototype KB* containing real knowledge from the literature regarding sensor/task types and the sensing capabilities required and provided by both. We then

[5]http://www.apple.com/ipodtouch/

Figure 5.6: User agent implemented as a mobile app interface on iOS devices.

measure the size of the full KB stored on the iPod Touch flash memory (summing up the sizes of the four tables), and the time to retrieve all the (BT, JUM) entries associated with a single task type.

To generate a *Synthetic KB* we randomly generate 1000 task types, each consisting of an integer ID and a string of random characters of length 50 representing the task type description (e.g. "event detection"). We then generate a constant number of BTs descriptions with a length of 100 characters and we do the same with the JUM table generating random utility functions represented by strings of source code. We then fill the relationship table associating every task type with all the possible combinations of (BT, JUM). Keeping constant the number of randomly generated task types — 1000 task types — we increase the number of (BT, JUM) per task type and we measure the memory usage and the time to retrieve all the (BT, JUM) entries related to a particular task type. As shown in Figure 5.7 and 5.8 we run the experiments generating first 10, then 50 and finally 100 (BT, JUM) per task type. This leads to an increase in the size of the relationship table respectively containing 10000, 50000 and 100000 entries.

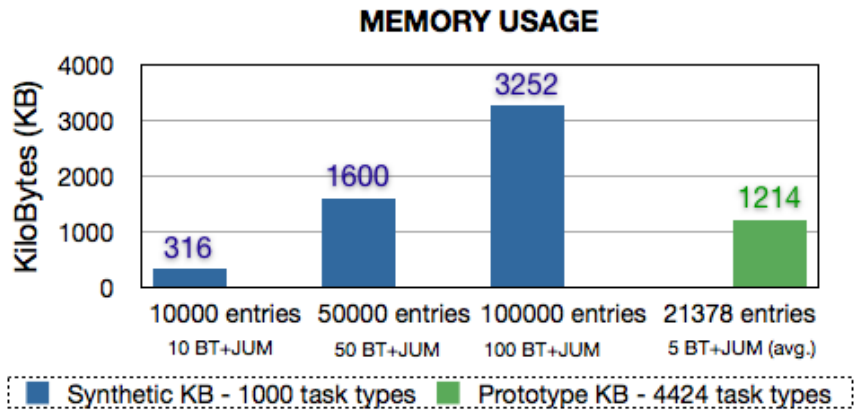

Figure 5.7: Memory usage of Lightweight KB on a mobile device.

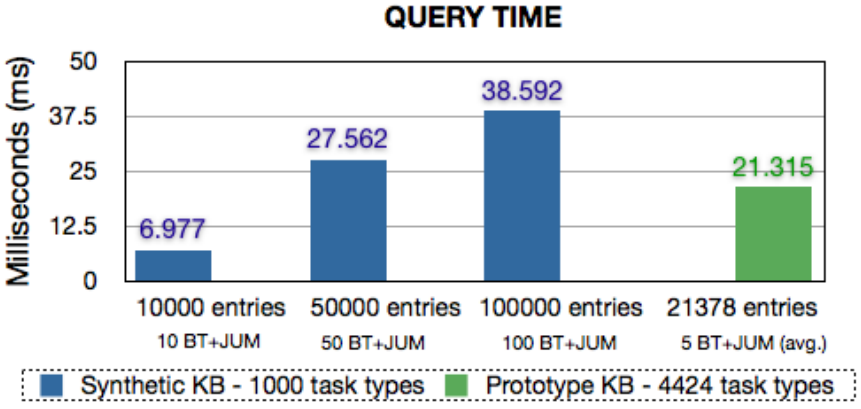

Figure 5.8: Query time for Lightweight KB on a mobile device.

We also carry out benchmarks on a KB filled with realistic data from the literature; we refer to this as *Prototype KB*. To generate these realistic task types and sensor types we use openly available information about sensor devices/platforms and military sensing tasks mainly using the Missions and Means Framework as described in [44]. More in detail, we use the knowledge base described in [27] and compute all the reasonable task types selecting the appropriate detectable, identifiable and distinguishable elements of the ontology. We feed this realistic data to a reasoning process like the one described in Section 5.3 implemented on a laptop computer, and we then generate all the possible outputs given all the possible inputs. We store these results on the iPod Touch using

the previously mentioned ER schema. We provide 4424 task types as an input to the user. Two examples of realistic task type descriptions are: *Detect_NonMilitaryVessel*, *Identify_RefuelingEquipment*. On average the reasoning process generates five different (BT, JUM) per task. Note that all the benchmarks conducted in this section are repeated for 20 times and then averaged to get significant values also with randomly generated entries/queries.

We summarize the experiments in Figure 5.7 and Figure 5.8, which show three main results. First, the storage space occupied by the KB seems to grow linearly with the number of entries in the relationship table. Second, the query time seems to increase logarithmically with the size of the relationship table. This is mainly due to the database engine used. More precisely, SQLite uses an R*-Tree which is a refinement of the R-Tree[6]. The search time complexity in an R*-Tree is basically the same as a B-Tree: $O(log(n))$ as it performs a binary search. Finally, the last and most important result is that with a realistic knowledge base the memory requirements and lookup performance on the mobile device are acceptable. In fact, our realistic knowledge base with 21378 entries occupies 12 megabytes in the flash memory of the iPod Touch – which considering the synthetic results is expected to grow linearly with the number of entries in the table. Most importantly, the time required to retrieve an entry from this lookup table is around 20 milliseconds, which is expected to increase logarithmically with the number of entries. The Apple iPod Touch which we used as an example of a mobile device has the following technical specifications[7]: iOS 4.1, CPU ARM11 620 MHz (underclocked to 533 MHz), 128 MB DRAM, and 8GB of storage on flash memory.

Note that in both memory and query time benchmarks, the data points reported in the graphs are sufficient to verify that the theoretical performances of the database engine are consistent with its implementation in the iOS environment. These results, in fact,

[6]http://www.sqlite.org/rtree.html – checked on 10th of December 2011

[7]http://daringfireball.net/linked/2008/11/25/new-touch-cpu – checked on 10th of December 2011. Apple does not officially publish full tech specs of its devices, these where obtained using a tool based on Unix `sysctl()` call.

confirm that growing linearly the size of the KB produces a non-exponential increase in the storage space and time required to interact with it. In this support, we also found that the standard error on the average time for querying entries from the synthetic KB is in the range of ± 5 ms. Instead, the error on the memory benchmarks of the synthetic KB is in the range of ± 8 kilobytes due to the slight differences in size of each of the randomly generated entries, but also related to the indexes and other metadata generated by the SQlite implementation in iOS. For the realistic KB instead, memory usage is basically constant among the different benchmarks due to the fact that the KB is always generated using the same data from the literature. The query time varies in the range of ± 5 ms like in the case of the synthetic KB, due to the granularity of time measurements that the iOS APIs and hardware allow to perform. Finally, we highlight that the realistic KB benchmarks diverge slightly from the values projected by the synthetic KB data points collected. The reason for this is that in a realistic KB the number of (BT, JUM) associated to each task type and their size in terms of bytes is not constant, as it is instead for the case of the synthetic KB. Therefore, we necessarily have a slight variation from the projected values of the measurements taken for both storage space and time required to retrieve an entry.

5.5.2 Distributed Protocol

We implement the distributed protocol in Java using *REPAST Simphony* — an open source agent-based discrete time simulation environment[8] — as a platform to simulate a sensor network composed of static wireless sensors of different types and mobile users on the field. The simulation design was inspired by the simulation tool described in [72]. In particular, we have implemented the Global Messenger component in [72] which functions as our message passing infrastructure, receiving and forwarding messages to different nodes in the network. Each node agent in the simulation (i.e. sensors and user devices) interacts with the Global Messenger by (1) fetching all messages re-

[8] http://repast.sourceforge.net/ - checked 10th December 2011.

lated to itself (2) processing this messages (3) uploading the outgoing messages to the global messenger. We have released the source code for both the Global Messenger and the node agents under the MIT license[9]. A screenshot of the simulation environment developed is shown in Figure 5.9, where dots represent users moving on the field, sensors are represented by triangles and squares represent tasks generated by users. Note that sensors are linked in the network if they are within the communication range of each other, and tasks are instead linked to the candidate sensors which could serve each task. A video of the demo which we published in [98] can be found at the following URL: http://www.youtube.com/watch?v=QzrpKRhGFRU – here the user agent which is implemented as a mobile app (in this case on a tablet) interacts with the REPAST simulation environment, a screenshot of the demo (with the iPhone interface) is shown in Figure 5.10.

Using this simulation environment, we tested the distributed architecture on randomly generated problem instances in which tasks arrive over time without warning, and depart after spending a certain amount of time being active. In our simulation setup we deploy 250 sensor nodes with a uniform random distribution on a 2D grid of 500m$\times$500m, randomly generating *two sensor types*: acoustic and video. Each sensor node has a Sensing Range (SR) and a Communication Range (CR), which in our simulation are SR = 30m and CR = 60m. We randomly deploy 50 user nodes on the field, each moving with a random-waypoint model and having a CR = 60m. Every 3 timesteps, 5 randomly selected users create a new sensing task in their surroundings with a uniform distribution (i.e., arrival rate = $5/3$ tasks per timestep), with a maximum distance of the task from the user node equal to CR. Each task is owned by a single user and is characterized by a location (x,y) on the 2D grid, a task type and expected duration (randomly chosen between 10 and 20 timesteps). Each task's priority and utility demand are generated with a uniform random distribution between 0 and 1. We discard a-priori tasks for which there are not enough sensors to be satisfied. Finally each task's expiration time is equal to half of its expected duration. In the simulation we use the

[9]The source code can be downloaded from: http://github.com/diegopizzocaro/GlobalMessenger/.

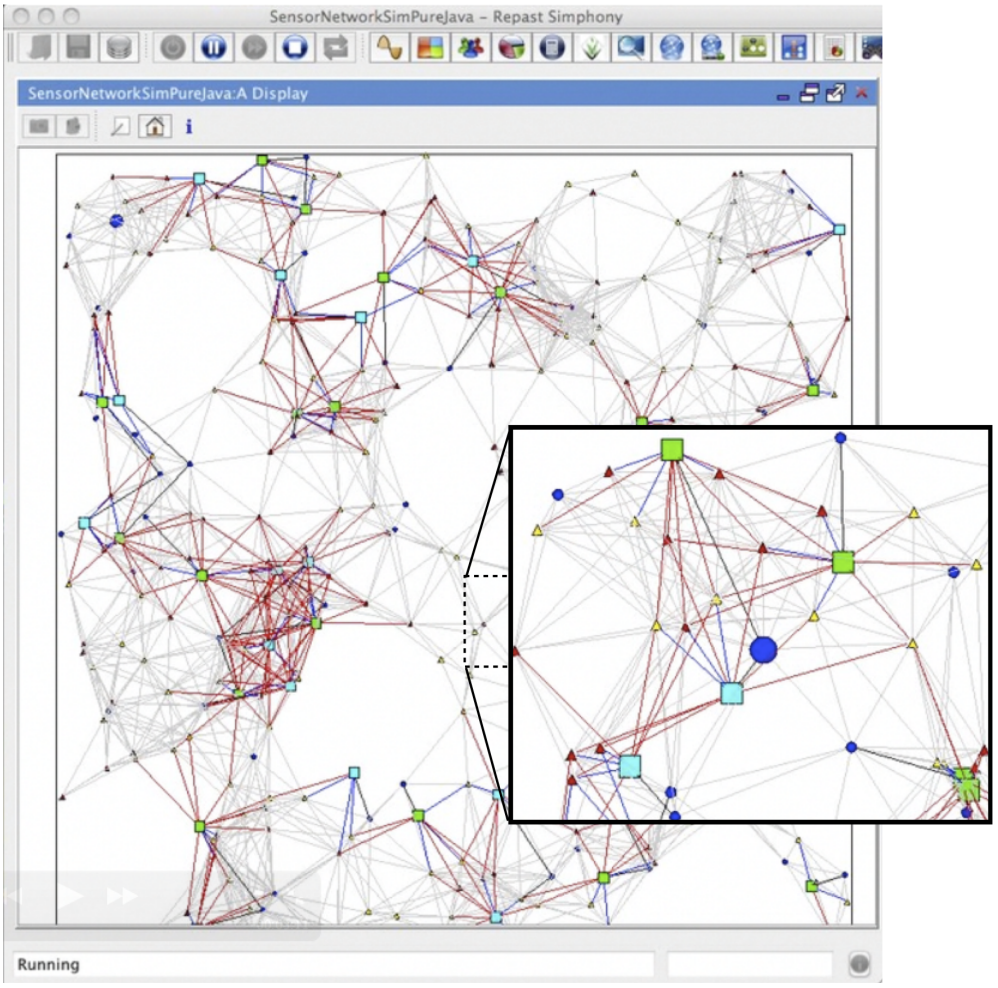

Figure 5.9: The simulation environment implemented in REPAST Simphony.

two *task types* described in Section 5.3: "event detection" and "2D localization". The type is chosen on the task's creation with a uniform random distribution among these two types. We implement *CDP* and *2D-Loc* with a brute force approach enumerating and evaluating all the possible subsets of the candidates, which in the worst case is 2^l (where l is the number of sensor candidates). A brute force approach to be implemented on a user device is computationally efficient by assuming that in an emergency response scenario the number l of sensors surrounding the user is on the order of tens.

We ran simulations for the full version of our protocol which includes both the rebid and preemption mechanisms (referred to as *Cost-driven Preemption* approach in the

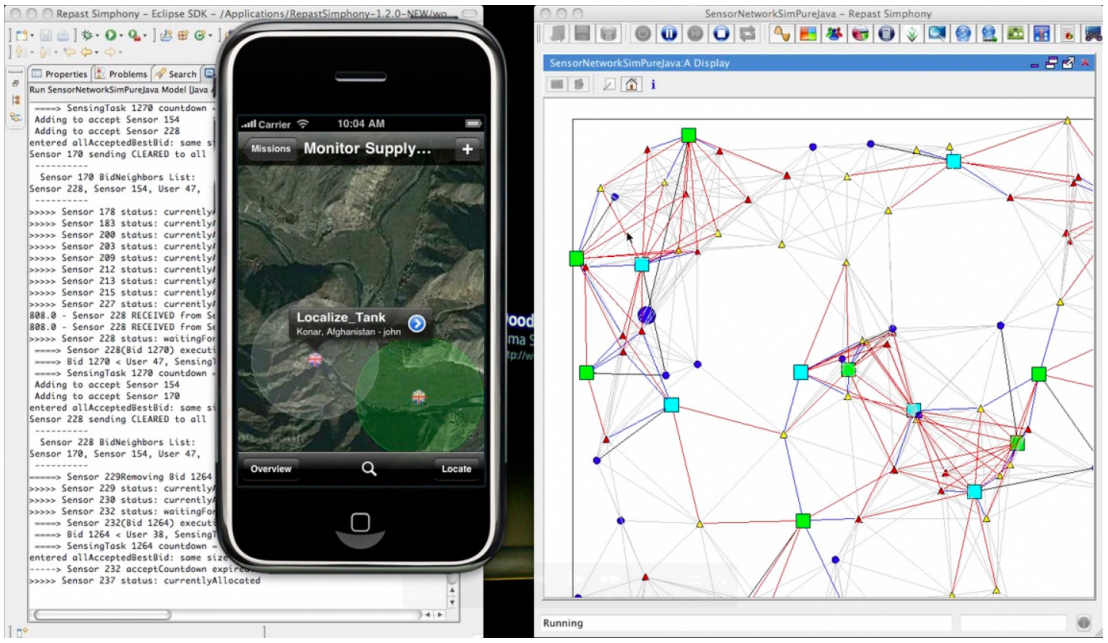

Figure 5.10: A screenshot of our demo in [98] showing how a user can submit tasks to the REPAST simulated environment implementing our distributed system.

graphs), and we compare this with 3 other versions of our protocol to analyze the communication overhead and the benefit of using the preemption and rebid mechanisms. We consider a version without rebid and preemption (*No Rebid*), a version which uses our rebid mechanism but no preemption (*No Preemption*) and a variant in which we use both rebid and preemption but we directly limit a sensor to be consecutively preempted only once (*Preemption = 1*). We repeat each simulation 10 times for 10000 timesteps and we average all the measurements. To compare the quality of the allocation achieved by each version of the protocol we consider the profit calculated at each timestep using the objective function in Section 4.4.1. In Figure 5.11 our experiments show that the *Cost-driven preemption* protocol offers higher profit at each timestep, achieving on average 60% of the total profit. Note that the total profit is obtained by summing at each timestep the profit of all the tasks being served by a sensor bundle (*satisfied tasks*), all the tasks which were not allocated a sensor bundle (*unsatisfied tasks*) and all the tasks which were previously preempted (*preempted tasks*). We keep the unsatisfied tasks and preempted tasks in the field for their remaining duration, so that the total

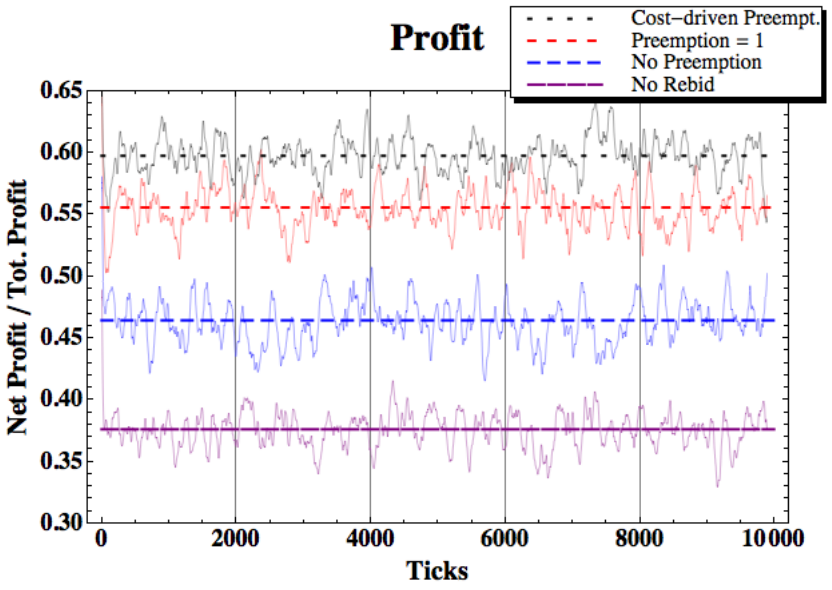

Figure 5.11: Total Profit (5/3 tasks per ts).

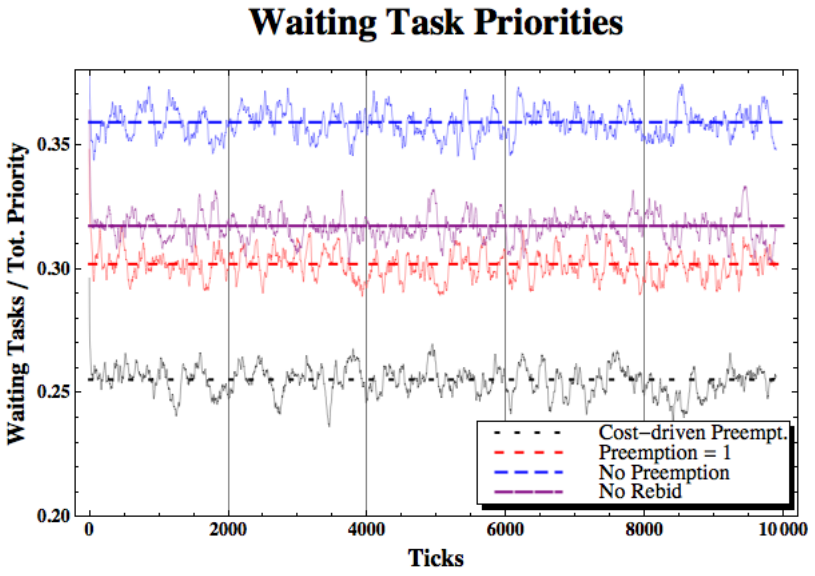

Figure 5.12: Waiting Tasks Priorities (5/3 tasks per ts).

profit represents the total potential task allocation quality. Note that this is a really loose upper bound as in general we usually cannot satisfy every task on the field. The *Preemption = 1* version has performance close to the *Cost-driven preemption*, while the *No Preemption* and *No Rebid* reach less than 50% of the total profit on average. In

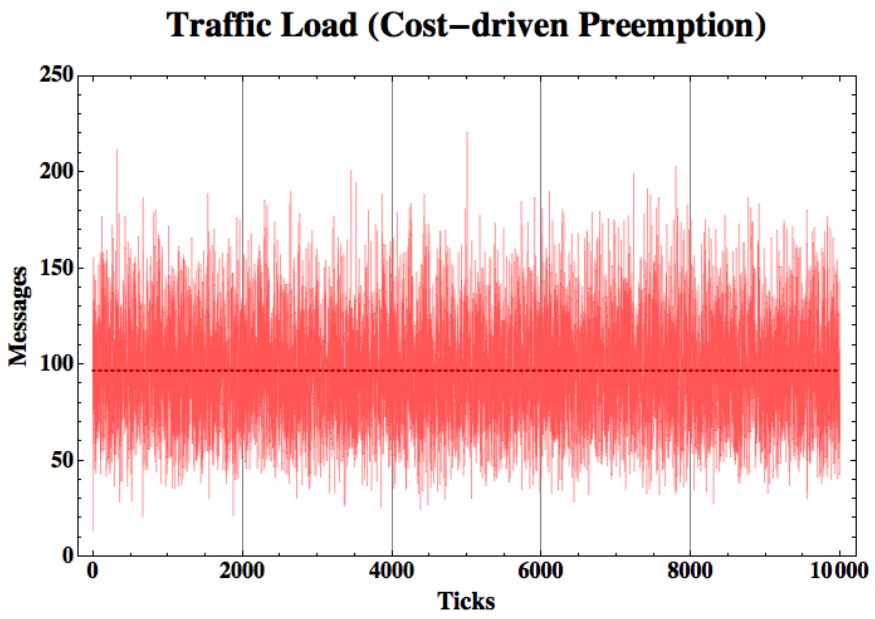

Figure 5.13: Network Traffic for Cost-driven Preemption (5/3 tasks per ts).

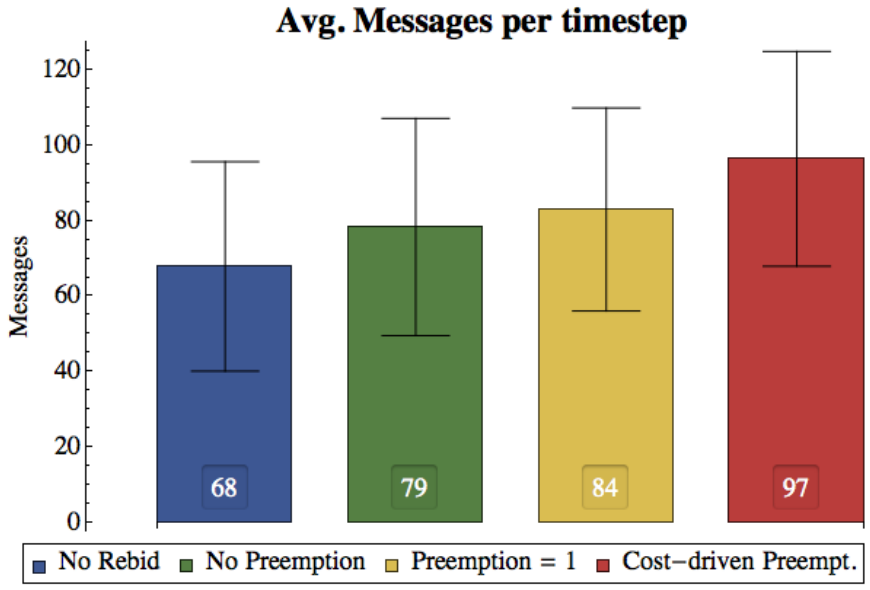

Figure 5.14: Average Messages (5/3 tasks per timestep).

Figure 5.12 we show the fraction of waiting task priorities over the total task priorities — calculated as the sum of the priorities of the satisfied, unsatisfied, preempted and waiting tasks — for each timestep. This graph shows that the *Cost-driven preemption*

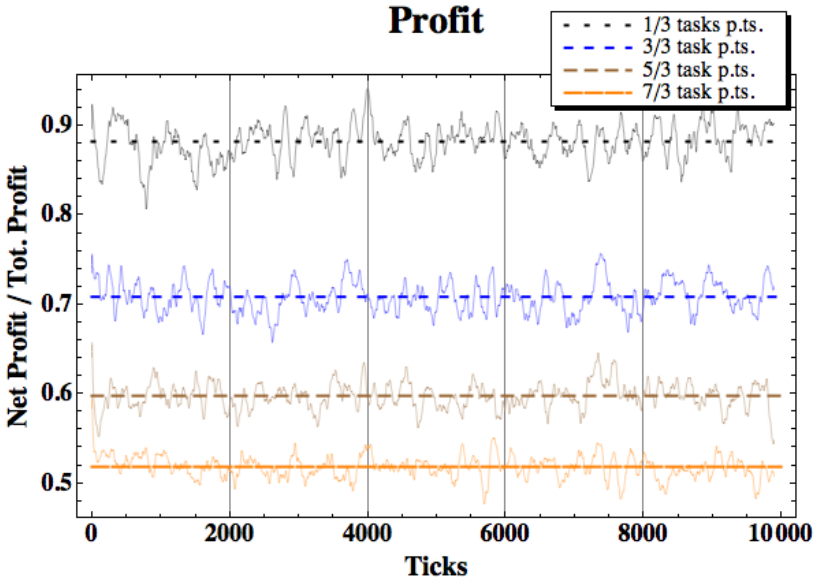

Figure 5.15: Total Profit varying task arrival rate.

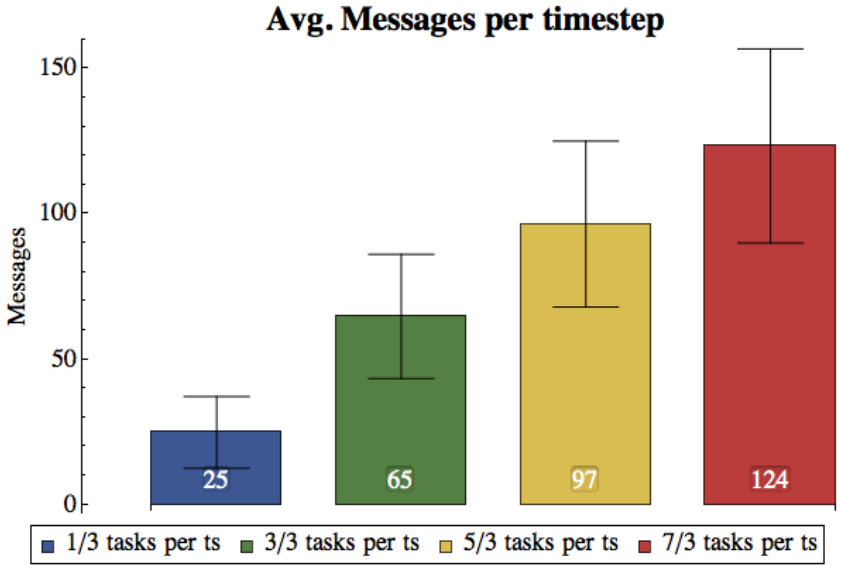

Figure 5.16: Average Messages varying task arrival rate.

version keeps only 26% on average of the most important tasks waiting for a decision. The worst waiting task rate is offered by the *No Preemption* (36%), while *No Rebid* (32%) performs a bit better because it sets the tasks to unsatisfied or satisfied just after the first bid.

Figure 5.13 shows the network traffic generated by the *Cost-driven preemption* protocol described in Section 5.4. Our version exchanges the highest number of messages compared with the other three versions as highlighted in Figure 5.14. Nonetheless the maximum number of messages exchanged every timestep is around 220 messages with a much lower average of 96 messages per timestep, and with an acceptable variance around ± 50 messages. Figure 5.14 confirms that the three other variants of the protocol provide similar network traffic levels and that the least overhead is provided by the *No Rebid* variant. The number of messages exchanged on average at each time step is highly dependent on the maximum size k allowed for each bundle, which is stored in the KB bundle generator. We limited this number to 10 sensors for *CDP*, while instead *2D-Loc* is already limited to exactly 2 sensors. Finally, in Figure 5.15 we vary the task arrival rate showing the total profit achieved overtime by the *Cost-driven preemption* version. We run simulations with 1/3, 3/3, 5/3 and 7/3 tasks per timestep and the results show that the performance of our allocation protocol decreases sub-linearly by increasing linearly the task arrival rate. Figure 5.16 shows how the traffic load generated by the protocol increases linearly arriving at a maximum of 120 messages per timestep on average which is much less than the total number of nodes (i.e. 300). This shows that the allocation protocol is scalable and resilient to high task arrival rates.

5.6 Conclusion

In this chapter, we implemented a distributed system based on the conceptual architecture for the dynamic MSTA problem proposed in Chapter 4. The main novelties of such a system are the use of both qualitative (i.e. knowledge based) and quantitative (i.e. numerical utility based) metrics to allocate sensors to tasks, and the interaction between sensors and mobile devices which we use to deploy part of the allocation mechanism.

We showed that it is feasible to implement a knowledge-based component able to deal

with knowledge-rich task requirements on a mobile device. This component – the *Knowledge-based bundle generator* – generates the best feasible sensor bundle for satisfying each task requirements using an explicit representation of those (such as an ontology) and semantically matches them with different sensing capabilities. We implemented the knowledge-base (KB) on a real mobile device (Apple iPod Touch) as a relationship table in the database engine. We considered both a "Synthetic KB", containing synthetically generated data, and also a "Prototype KB", containing realistic examples of sensor/task types from previous literature. Our experiments on the Apple iPod Touch showed that the storage space seems to grow linearly with the number of entries in the relationship table, while the query time seems to increase logarithmically. This is due to the optimized database engine used in the mobile device. In addition we proved that the Prototype KB, containing 4424 different task types, occupies 12 MB in the flash memory and delivers a query time of 20 ms. These results prove the feasibility of implementing the KB bundle generator on a modern mobile device.

The allocation protocol which we propose is an extension of a well known pre-existing disjoint coalition formation protocol proposed in [115], which was originally developed for the static settings in a generic multi-agent system. We adapted it in order to deal with our dynamic problem setting, to take advantage of the user devices as an entry point for tasks in the system and to generate sensor bundle requests (which we call *bids*) through the KB bundle generator component. The bid generated for each task is broadcasted by the user mobile device to the sensors included in the bundle, then each sensor greedily decides which one is the best task to serve as part of a bundle, calculating the average contribution that the sensor could provide to the task if the bundle is formed. Before effectively starting to serve a task as part of a bundle, each sensor first "agrees" to serve the bundle and then it is effectively allocated to the task only if all the sensors in that bundle agreed to join it. If one of the sensors does not agree to join a particular bundle we then use a *rebid mechanism*, going back to the KB bundle generator and demanding for another feasible bundle until a convergence timeout expires. Finally, we also allow sensors to be *preempted* from ongoing tasks, in the case a more

profitable task appears on the field; i.e., if a sensor receives a bid for a task to which it can provide a higher average contribution of utility then it decides to switch to the new task and it drops the support to the task which was previously serving.

We implemented the distributed system in a discrete time simulation environment and we ran experiments for a network of static sensors and mobile users moving on the field and generating tasks at different rates. We used as an example the two types of tasks with non-additive sensor utility models defined in Chapter 4 for event detection and 2D localization tasks. We considered two simple different types of sensors: video and acoustic, where only the second was able to serve both tasks. We evaluated the total utility (scaled by the task profits) provided over time to the set of satisfied tasks by the bundles formed using both the KB bundle generator and the allocation protocol. We found that using a rebid mechanism, taking advantage of the KB bundle generator, offers flexibility in terms of allocation and as a consequence it increases the total profits of the task satisfied over time. We also showed that the overall profit of the satisfied tasks over time increases if we allow for sensors to be preempted from ongoing tasks in the case a more profitable task appears on the field. Finally, we showed that by linearly increasing the task generation rate, the overhead traffic generated by the mechanism grows linearly, and that the quality of allocation decreases sub-linearly. We therefore answered *Research Question 2* by developing a scalable solution for the *ST-MS-IA-HE* problem in a dynamic setting, with both linear and non-linear sensor utility models and richer knowledge-based requirements for each task.

Although the dynamic MSTA model is not constrained to static sensors, as we could for example include the cost of moving a sensor from one location to another in the calculation of the bundle utility, our experiments are instead focused on a network of static sensors and therefore we do not have a considerable evaluation of how the system would perform in the case of mobile sensors. This is an interesting direction for future work, which might also consider extending the dynamic MSTA model to include more explicitly spatial and temporal constraints when deciding to move a sensor from one

location to another, using a time-extended allocation mechanism similar to [100]. We are also aware that limiting our attention to *Single-Task sensors* restricts the scope of applicability of our system and therefore we propose to explore a variant able to deal with the *MT-MS-IA-HE* problem, where sensors can serve multiple tasks at the same time. We note that the distributed system architecture proposed in this chapter could be adapted to solve the *MT-MS-IA-HE* problem by extending the *overlapping coalition formation protocol* described also in [115].

Chapter 6

Conclusion & Future Work

In this thesis, we considered the problem of allocating sensors to multiple sensing tasks in an environment that does not provide enough information to plan for future allocation, such as emergency response or military operations, therefore constraining our mechanisms to perform *instantaneous allocation* (*IA*). We considered a sensor network consisting of a large number of sensing devices of different types, which we referred to as *heterogeneous sensor network* (*HE*). These are increasingly used to support emergency responders in the field usually requiring many *sensing tasks* to be supported at the same time. By a sensing task we mean any job that requires some amount of sensing resources to be accomplished such as localizing persons in need of help or detecting an event. We considered the most general case where each task usually requires a group of sensors (*sensor bundles*), and we called these *Multi-Sensor tasks* (*MS*) as opposed to tasks which require just a single sensor to be accomplished. We associated to each task a *demand* for sensing capabilities and a *priority* (or importance). Tasks might share the usage of a sensor, but, more often in this scenario, they compete to exclusively control it because of the limited number of sensors and large number of tasks. Sensors are in fact *scarce* and in *high demand*. We considered a particular subset of sensors which can serve exclusively one task at a time, like directional sensors, we referred to these as *Single-Task sensors* (*ST*). Each *sensor bundle* can offer a certain *utility* (or amount/quality of information) to each task, given that the aggregated capabilities match the task sensing requirements.

In such cases, it might not be possible to satisfy the requirements of all tasks us-

ing available sensors, and therefore we need to answer the question "Which sensor should be allocated to which task?", which summarizes the *Multi-Sensor Task Allocation* (MSTA) problem. In particular, considering the assumptions above we identified our MSTA problem with the label *ST-MS-IA-HE*. We studied this problem in both a *static setting*, in which all task requests from emergency responders arrive at once, and a *dynamic setting*, where tasks arrive and depart over time. The static setting is motivated by a scenario in which we have to support a set of long lived tasks (e.g. perimeter monitoring) or in general when tasks requested at different times can be batched together. The dynamic setting, instead, is the more common scenario for an emergency response operation, where we need to provide fast-response to tasks and adapt quickly to changes. We considered different constraints for each of these problems and we provided novel solutions based on *centralized* and *distributed* approaches. These schemes aimed at maximizing the overall *utility* of the network and we evaluated their performance using mainly simulations on randomly generated problem instances. Our practical algorithms scale well with different number of task requests and manage to improve the utility of the network, prioritizing the most important tasks. Therefore, we answered our two *research questions* showing at what degree we can provide a scalable and optimized solution to our *ST-MS-IA-HE* problem in both static and dynamic settings. Below, we summarize our novel research contributions and discuss future directions.

6.1 Summary of Contributions

In this section, we summarize our research contributions highlighting the features of the static and dynamic MSTA models proposed for *ST-MS-IA-HE*, and the *centralized* and *distributed* approaches proposed for each. We summarize our research major contributions in the following list:

1. In Chapter 3 we formalized the *ST-MS-IA-HE* in *static settings*. We explored

a simplified utility model in which utilities of sensors contributing to the same task are additive; i.e., they sum up linearly. We constrained each task usage of sensing resources by assigning them a *budget* (e.g. energy or monetary). Each task is characterized by a *demand* of utility and a *threshold* on its demand to be reached in order to be satisfied. For this setting we used only quantitative utility measurements, based on physical attributes of the sensor instance as an example. We considered a utility value inversely proportional to the degradation of the radio signal with the sensor-task distance. We modelled this problem as a bipartite semi-matching problem with both packing (i.e., maximum budget) and covering constraints (i.e., minimum threshold on demand).

2. In Chapter 3 we proposed three *centralized algorithms* for the *ST-MS-IA-HE* model in static settings. The first algorithm is a simple centralized greedy algorithm that repeatedly attempts to satisfy the tasks with the highest potential profit. The second algorithm, Multi-Round GAP (*MRGAP*), is a novel algorithm that treats tasks as knapsacks that together form an instance of the Generalized Assignment Problem (GAP) and uses a multi-round heuristic for dealing with the threshold on task demands. The third algorithm is instead an adapted combinatorial auction algorithm where tasks bid for groups of sensors. We evaluated the different algorithms on randomly generated problem instances. In particular, the experiments showed that the best performance in terms of profit of task satisfied and fraction of budget spent are offered by our *MRGAP algorithm*, outperforming the greedy and the combinatorial auction based algorithm. We also found that by giving higher budgets to tasks does not help since what limits the amount of achievable profits is competition and not the amount of budget provided. Finally, we showed that the three algorithms provide timely solutions by benchmarking their computational runtime. We noted that the greedy algorithm offered the best time performance, and MRGAP displayed a much slower increase in the computational time compared to the auction based approach when growing the problem instance size.

3. In Chapter 4, we formalized *ST-MS-IA-HE* in *dynamic settings*, allowing for more general sub-additive or super-additive sensor utility models; i.e., utility from sensors contributing to the same task might not sum up linearly. Given that each task now has a duration, we include the time dimension in the problem formulation. This problem can be modeled as a tripartite semi-matching problem. As in Chapter 3, we associate each task with a demand value and a profit value; although differently we now associate each *sensor bundle*-task pair with a possibly non-additive utility offer. So we first need to group sensors into bundles, evaluate their joint utilities, and finally decide which one is the best assignment of *sensor bundles* to *tasks*. In this version of the problem, we assume there are no budget constraints for each task motivated by the fact that in a dynamic environment we need more flexibility in the allocation of resources because we lack of exact information about what will happen in the future. Finally, we allow for re-allocation of sensors to newly created tasks, and in order to avoid preemption of sensors from ongoing tasks to happen too often, we include a *re-allocation cost* associated to each sensor which we aim at minimizing while maximizing the total profit.

4. Finally, in Chapter 5, we presented a novel *distributed architecture* to solve *ST-MS-IA-HE* in dynamic settings, which uses both knowledge-based matching (i.e., qualitative metrics) and numerical utility functions (i.e. quantitative metrics) to better deal with heterogeneous sensors networks. We extended a pre-existing distributed coalition formation protocol and deployed part of the architecture on mobile devices showing that it is feasible to do knowledge based sensor-task matching on such devices. Our approach utilizes formal tools, such as ontologies, to explicitly represent sensing capabilities and task requirements in order to reason with these; the aim is to be able to find which individual or group of sensor types matches the sensing capabilities required by a certain task type. We evaluated the performance of our architecture in two ways. First, we implemented a prototype mobile application which contains the knowledge-

based component and we measured its performances on a real device in terms of memory usage and query time. In addition, we ran simulations and evaluated the total sensor utility over time on randomly generated scenarios with users moving on the field and requiring sensing tasks to be satisfied by the sensors composing the network. In these simulations, we used the two non-linear utility models which we proposed for event detection and 2D localization tasks in Chapter 4, and we showed that by allowing for preemption and a rebidding mechanism in the protocol we achieve a higher total profit over time. Our simulation results showed also that the distributed protocol is scalable, performing well in terms of overhead traffic and allocation quality when increasing the task arrival rate (i.e. the rate at which tasks are generated by users on the field).

In addition, we also summarize two minor contributions which are related to the above major contributions:

(A) In Chapter 4, we formalized from the literature two non-additive *sensor utility models*, with the aim of showing how it is necessary to consider general non-linear utility functions. We use these as examples to evaluate the performance of distributed mechanisms for *ST-MS-IA-HE* in dynamic settings. For *event detection*, we propose a model based on cumulative detection probabilities; for *two-dimensional localization* tasks, we propose a model based on the distance and bearings of a pair of sensors to a task. We formalize these two problems as static MSTA problems with non-linear utilities, in restricted scenarios where we only have event detection or two-dimensional localization tasks. We then generalize these two problems into a more abstract *static MSTA problem*, in which we allow for both linear and non-linear utility functions and for any task/sensor type to be used. We then use this problem as a base for the *dynamic MSTA problem* in Contribution 3.

(B) In Chapter 4, we proposed a *conceptual architecture* to solve *ST-MS-IA-HE* in dynamic settings with generic sensor utility models (Contribution 3), using a combination of qualitative (i.e., knowledge based) and quantitative (i.e., numerical

utility based) metrics to drive the allocation. Our architecture is therefore composed mainly by two components: the first is a *Knowledge-based bundle generator* which recommends sensor bundles fit for purpose for a particular task combining semantic matching between the capabilities provided and required, with (possibly non-linear) utility functions. The second component is an allocation mechanism which given these recommendations sorts out the competition for exclusive usage of sensing resources among multiple tasks. We also include the users in the loop, by allowing them to submit tasks to the system through a mobile app. We describe how mobile smart-devices could be used to deploy components of such an architecture, discussing the trade-offs related in choosing different implementations, as a centralized or distributed system. In particular, given that a centralized approach is unrealistic in our dynamic setting, we focus on a distributed implementation of such architecture contained in Contribution 4.

6.2 Future Research Directions

Although our focus in this dissertation has been on the *ST-MS-IA-HE* problem instance, some interesting research directions are represented by exploring other problem instances relevant for certain emergency response scenarios. For example, as proposed in [114], we could consider *Multi-Task sensors*; i.e., where sensors are able to provide information to multiple tasks at the same time. Some seismic and ambient temperature sensors, in fact, are able to serve potentially multiple tasks in different locations, although each sensor will still have a maximum number of tasks which it could serve provided bandwidth or energy constraints and sensing limitations. Directional sensors could also serve multiple tasks as suggested in [54], where cameras are optimally configured for capturing the largest number of targets (i.e. tasks). In such cases, MSTA mechanisms are still required in order to decide which tasks should be served, and to optimize the utility of the network under those constraints.

Our assumption of *instantaneous allocation* is sound, but in certain cases we might be provided with enough information about the future to schedule sensor allocations in a *time-extended* fashion. For example, we might be provided with a certain level of confidence in the plan of a mission, which we could model as a graph of tasks with temporal and probabilistic relations. Missions could be compatible with each other if their needs do not exceed the limits of the sensors, e.g. in terms of bandwidth or energy; otherwise, we could send feedback to the responder in the field, indicating which information requirements need to be adjusted. The decision loop should eventually converge and the utilization of the resources maximized. This is similar to tasks precedence links and goal lattices studied for example in [77, 114].

Another venue for research is to study how we could close the loop between data requested and data delivered in systems solving MSTA problems by developing dissemination and delivery mechanisms. For example in [105] we have explored ways of integrating allocation schemes with sensor data rate optimization for a variant of the *ST-MS-IA-HE* problem in static settings. The main result is that the convergence time and overall utility may be improved by sharing information across allocation and rate adaptation (i.e., dissemination) mechanisms. In [21] we studied instead, independently from MSTA mechanisms, how to efficiently deliver information bundles to mobile users in the field, where by information bundles we mean correlated data items to be delivered as a package to users, such as a map bundled with a set of geotagged pictures. Such works point at the interesting challenge of adapting MSTA systems to include dissemination mechanisms. These as suggested should be able to exchange as much information as possible with both the allocation algorithms and the knowledge based component (as in [105]); moreover, we think they should focus on delivery of information packages (i.e. data bundles) rather than single data items to mobile users ([21]). Other works addressed different instances of sensor selection with data dissemination aspects, such as in [2, 15], but we think that the focus should shift to include mobile user devices in the delivery mechanisms, perhaps with the use of opportunistic routing techniques [79].

Including dissemination and delivery of sensor data, in response to task requests, offers also the opportunity for a user to share this data with other users who might be interested in the same task, thus giving the opportunity to include social network ideas in such MSTA systems. Namely, we think that if a user could see sensing tasks generated by other users currently being served by sensors in the field; e.g., using the mobile app interface in Chapter 5, then we might push them to subscribe to such tasks instead of creating new ones, so that the information collected will also be delivered to them. We hypothesize that by providing the users with enough trust in the information generated by other users' tasks, this would yield to a decrease in the new tasks generated over time (i.e. task arrival rate) and therefore an effective decrease in the competition for sensing resources with an overall higher utility.

It is therefore clear that data and sensor privacy concerns in our distributed system exist because sensitive information about users, sensors and tasks could be collected, stored and shared. By data and sensor privacy we mean that data generated by sensors and metadata about these (e.g. sensing capabilities) should be securely shared only with selected users. To ensure data privacy we are planning to explore how to integrate data/sensor sharing policies, similar to the ones proposed in [62, 92], with our allocation mechanisms; so that within the system we might securely share information about sensors, tasks and users.

Another research opportunity lies in the study of the effects of mobility on our solutions. In fact, even if our schemes are quite general and could include the mobility inside the cost parameter in both static and dynamic settings, this approach is far from optimal. It would be interesting to evaluate how we can design better solutions taking advantage of mobility. We can see two types of mobility: controlled and uncontrolled. Controlled mobility is the ability to move all or some of the sensing resources in order to achieve better assignments to tasks. In uncontrolled mobility the sensing resources might be mounted on people (for example, helmets of emergency responders) or vehicles in which case the system has no control on where the sensors are to be

moved. Examples of the first are given in [100, 113, 124], which consider both travel time and distance as a parameters in the allocation of sensors (or autonomous agents) to different tasks.

The case of uncontrolled mobility offers probably the most exciting research opportunities, in particular if we look at this with the more general perspective of utilizing people as an extension of the sensor network. In such a scenario, in fact, people could be wearing sensors or reporting observations about the real world, while at the same time requesting data from the sensor network or from other people. In this thesis, we showed that smart devices provide a handy access point to our allocation system, through which users can generate sensing task requests; moreover these can be used to deploy part of the architecture. We think that smart devices have the potential to be used as tools to integrate eyesight witnesses with data coming from the sensor network, e.g. users could use such mobile interfaces to report observations about the real world. Moreover, we could think of using the sensors integrated in such devices (e.g. light sensor, microphone or camera) to extend the sensor network, effectively considering people as "uncontrolled" mobile sensors. This has very recently been defined as "mobile phone sensing" in [43], which proposes an opportunistic computing approach to the problem of collecting sensing data exclusively using the user mobile devices.

We think that by going a step further and actually integrating mobile phone sensing with an already deployed sensor network could provide extremely interesting research opportunities. We foresee three main research themes in this direction: (1) people as sensors, (2) social sensing and (3) app-centric architectures. As recently noted in [45], people can be seen as "intelligent sensors" since they can share their observations about real world or also actual sensor data generated with their mobile devices. It would be interesting to explore if we can "task allocate" humans, e.g. by asking them to take a picture of something we are interested in; alternatively, if a user shares eye witness information we could explore how to combine this with real sensor data. The second theme we propose is *social sensing*, i.e. the integration of sensor networks with mobile

social networks. Some examples lies in the exploration of which sensing data we can crowdsource from the users in the field, or alternatively which real sensor data we can improve by using social information filters (e.g. rating or review mechanisms). Researchers have started to address these issue, for example in [14, 67, 99], however we think these problems still need to be studied from a distributed optimization and opportunistic computing point of view. Finally the third theme, *app-centric architecture*, covers how we could use mobile devices, and in general systems centered around the mobile app paradigm, to deploy different parts of architectures on top of heterogeneous networks of sensors and mobile devices. In Chapter 4, we have explored how we could achieve this in a centralized or distributed system but this needs more investigation, especially for the case of a hybrid system where cloud services, local mobile and sensing devices might collaborate to achieve higher network utilities for a particular set of tasks.

Appendix A

Appendix: $(2+\epsilon)$-approximation algorithm for GAP

In Chapter 3 we showed that the MRGAP algorithm uses the Generalized Assignment Problem (GAP) algorithm as a subroutine to find a solution to the static MSTA problem. Here we describe how we combined the approximation algorithm in Cohen et al [22] with the algorithm in Ibarra & Kim [51] obtaining a $(2+\epsilon)$-approximation algorithm for GAP. This work derives in part from the master thesis [93] by the same author of this dissertation, however the remarks on implementing the combination of Knapsack and GAP are entirely new to this thesis and were published in [94].

An algorithm is an α-approximation algorithm if $|OPT| \leq \alpha \cdot |ALG|$ for $\alpha \geq 1$, where $|ALG|$ is the value of the objective function for the feasible solution returned by the approximation algorithm, and $|OPT|$ is the value of the objective function for the optimal solution[1].

(α+1)-Approximation Algorithm for GAP

Given any other α-approximation algorithm for the knapsack problem, Cohen et al's algorithm (Algorithm A.1) is an (α+1)-approximation algorithm for the Generalized Assignment Problem. The main idea of the algorithm is to iteratively solve the knap-

[1] It is probably easier to understand the definition of α-approximation algorithm if we consider $\alpha \leq 1$, then we have that the condition is: $\alpha \cdot |OPT| \leq |ALG| \leq |OPT|$.

sack problem, each time selecting only one of the bins of the GAP model as a knapsack, and considering as items to insert into the knapsack all the items given as input to the GAP. During each iteration, the knapsack problem will use as size for each item the size that each of them has for the current bin, and as profit the residual profit. The residual profit p_i^{RES} of an item i for bin j is defined as: $p_i^{RES} = p_{ij}$ if i is not selected for any other bin, otherwise $p_i^{RES} = p_{ij} - p_{ik}$ if i is selected for bin k.

This algorithm has complexity $O(m \cdot f(n) + m \cdot n)$, where n is the number of items (sensors), m is the number of bins (tasks) and $f(n)$ is the running time of the α-approximation algorithm for the knapsack problem.

Algorithm A.1: Cohen et al's (α+1)-Approximation Algorithm for GAP

1: **for** each sensor S_i **do**
2: $T[i] = -1$
3: **for** each task T_j **do**
4: Call α-*Approximation Alg. for Knapsack* to find a solution to task T_j using the residual profit function P_j^{RES}
5: **for** each S_i scheduled by the α-*Approximation Alg.* to T_j **do**
6: $T[i] = j$

α-Approximation Algorithm for Knapsack Problem

Ibarra & Kim's algorithm (Algorithm A.2) is an efficient (1+ϵ)-approximation algorithm for an arbitrary $\epsilon > 0$, so if we set $\alpha = (1+\epsilon)$ we can use the notation α-approximation algorithm. This algorithm is an FPTAS (Fully Polynomial Time Approximation Scheme) i.e. its time and space complexity grows polynomially with n (number of items) and exponentially with $1/\epsilon$ (both space and time complexity are $O(n/\epsilon^2)$).

This is a dynamic programming algorithm which defines the elements of a matrix A with n rows (one for each item) and $|ALG|$ columns (one for each possible value of the objective function). Since the value of $|ALG|$ is unknown, the matrix A is defined

on U column where U is any upper bound on the value of the optimal solution $|OPT|$. In particular, for $l = 1, \ldots, n$ and $v = 1, \ldots, U$ the element $A[l, v]$ represents the minimum capacity required to obtain a value for the objective function that at least equals v using only the first l items. If it is impossible to get v for the objective function using only the first l items then the rule is to set $A[l, v] = +\infty$. Formally:

$$A[l, v] = \min\{q : \sum_{i=1}^{l} p_i x_i \geq v, \sum_{i=1}^{l} w_i x_i \leq 1, x_i \in \{0, 1\}, i = 1, \ldots, l\}$$

Once all the elements of the matrix A are defined, the value of the optimal solution is:

$$|ALG| := \max\{v : A[n, v] \leq c\}$$

The Ibarra & Kim's algorithm is shown in Algorithm A.2 for completeness.

Since this algorithm requires that each profit value is an integer, and since the time complexity depends strongly on the maximum item profit, we need to use scaled integer profits different from original item profits. Formally stated, we will use $p'_i = \lfloor p_i/\delta \rfloor$ for $i = 1, \ldots, n$, where δ depends on the required approximation error ϵ in the returned solution ($\delta := \frac{\epsilon \tilde{p}}{n}$, where $\tilde{p} = \max_i\{p_i\}$).

Finally, to find the real value of the objective function we will have to backtrack on which items are chosen to be inserted into the knapsack and then use the non-scaled profits to compute the original value of the objective function. Therefore, as in all the dynamic programming algorithms, we also need to have another matrix of pointers that will let us backtrack the solution, i.e. to understand which items were inserted in the knapsack. This matrix of pointers will have the same dimensions as the matrix A used to compute the optimal value of the objective function.

Algorithm A.2: Ibarra & Kim's α-Approximation Algorithm for Knaspack (with $\alpha = \epsilon + 1$).

1: **for** $v = 1, \ldots, p'_1$ **do**
2: $\quad A[1, v] := w_1$
3: **for** $v = p'_1 + 1, \ldots, U$ **do**
4: $\quad A[1, v] := +\infty$
5: **for** $l = 2, \ldots, n$ **do**
6: $\quad$ **for** $v = 1, \ldots, p'_l$ **do**
7: $\quad\quad A[l, v] := \min\{A[l-1, v], w_l\}$
8: $\quad$ **for** $v = p'_l + 1, \ldots, U$ **do**
9: $\quad\quad A[l, v] := \min\{A[l-1, v], A[l-1, v-p'_l] + w_l\}$

Combining GAP and Knapsack algorithms

Combining the GAP $(\alpha+1)$-approximation algorithm with the Knapsack α-approximation algorithm where $\alpha = 1 + \epsilon$, leads to a $(2+\epsilon)$-approximation algorithm where $\epsilon > 0$ is the maximum error in the returned solution (which in our case will be $\epsilon = 0.005$). As stated previously the time complexity of this algorithm is a function of the time complexity of the Knapsack α-approximation algorithm used (that is $O\left(\frac{n}{\epsilon^2}\right)$), and in particular it will be $O\left(\frac{mn}{\epsilon^2}\right)$ where m is the number of tasks and n is the number of sensors.

Although the FPTAS has a good asymptotic performance guarantee, when the input values are scaled to a not-unreasonable precision (e.g. with $\epsilon = 0.005$) and a naive implementation is used, the running time can actually become significant. This is especially so in our setting, which involves solving a knapsack problem for each task. Various coding tricks and improvements have been discovered (and rediscovered) for the basic knapsack algorithms since they appeared in the 1970s. Following, we describe in detail the specific implementation improvements we used, which allowed our implementation to run in reasonable time on static MSTA problem instances of the size

we considered in our experiments in Chapter 3 (e.g. 500 sensors and 100 tasks).

Implementation of GAP and Knapsack Algorithms

As a matter of note, the Knapsack algorithm requires a lot of memory, since it generates two matrices (one to compute the objective function and the other to backtrack the solution) of dimension $n \times U$, where both n and U are fairly large. The number of items n is the number of sensors generated and in our experiments this is a fairly large number (in fact, we want to be able to deal with large sensor networks composed by hundreds of nodes). The upper bound U is large since in the classic implementation of the Knapsack algorithm this upper bound is computed by: $U := \tilde{p} \cdot n$ (where $\tilde{p} = \max_i (p_i)$); such an upper bound is not beneficial, especially if we give as input to the knapsack algorithm a large number of items and the knapsack has a small capacity (which is the case of the static MSTA problem, when in the presence of dense sensor networks and low budget). The fact that the dimensions of the matrices are very large also has strong consequences on the effective computational time of the Knapsack algorithm, since it will have to fill a large matrix to find the solution. We solved this issue related to *memory usage* and *computational time* using three techniques. The first technique is to reduce the number of items n given in input at each round of the GAP algorithm to the Knapsack subroutine. We delete from the input those items that have zero profit for that specific bin j, and also items that are too large to fit into the bin (i.e. in terms of the static MSTA problem $c_{ij} > b_j$, and in terms of the Knapsack problem $w_i > c$). In this way we obtain a reduction of n, that in our specific test cases are from hundreds of sensors (all the sensors) to tens (only the ones with non-zero utility); drastically reducing the number of matrix rows created by the Knapsack subroutine.

The second technique applied was to reduce the number of columns of the matrices (U) used in the Knapsack subroutine. We first solve the knapsack problem using the FPTAS with $U = \sum_i p_i$ and without creating the matrix of pointers. We then find the value of $|ALG|$ (the optimal value of the objective function). This computation requires much less memory, since it uses a closer upper bound and it does not create

the matrix of pointers. We then rerun the Knapsack subroutine with $U = |ALG|$ and this time generate the matrix of pointers. Therefore the algorithm will generate two matrices with a minimal number of columns. These two techniques improve both memory usage and effective computational time, since they reduce the dimensions of the matrices used. Simply stated they limit the search space by reducing the number of possible states.

The third technique improves the memory usage but not the effective time spent to compute the solution in the Knapsack subroutine. It consists of substituting the $n \times U$ matrix used to compute the value of the objective function with a $2 \times U$ matrix (the matrix of pointers remains $n \times U$). This idea derives from the dynamic programming implementation of the knapsack algorithm (Algorithm A.2), requiring only the value contained in the previous row $l - 1$ to compute the value of row l. Making use of this we can generate just a $2 \times U$ matrix, and we can compute the value of the cells of a row by keeping only the previous row and overwriting with the new computed values the current row (that contain older values no longer required).

Bibliography

[1] ABRACHE, J., CRAINIC, T. G., GENDREAU, M., AND REKIK, M. Combinatorial auctions. *Annals of Operations Research 153*, 1 (2007), 131–164.

[2] AGGARWAL, C. C., BAR-NOY, A., AND SHAMOUN, S. On sensor selection in linked information networks. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)* (June 2011), IEEE, pp. 1–8.

[3] AHMED, N., KANHERE, S. S., AND JHA, S. Probabilistic coverage in wireless sensor networks. In *Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary* (2005), IEEE Computer Society, pp. 672–681.

[4] AI, J., AND ABOUZEID, A. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization 11*, 1 (Feb. 2006), 21–41.

[5] AJI, S., AND MCELIECE, R. The generalized distributive law. *Information Theory, IEEE Transactions on 46*, 2 (Mar. 2000), 325 –343.

[6] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. Wireless sensor networks: a survey. *Computer Networks 38*, 4 (Mar. 2002), 393–422.

[7] ATAMTURK, A., NEMHAUSER, G. L., AND SAVELSBERGH, M. W. P. A combined lagrangian, linear programming, and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics 1*, 2 (Mar. 1996), 247–259.

[8] AURENHAMMER, F. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys 23*, 3 (Sept. 1991), 345–405.

[9] BALAS, E., AND PADBERG, M. W. Set partitioning: A survey. *SIAM Review 18*, 4 (Oct. 1976), 710–760.

[10] BASS, F. M., TIGERT, D. J., AND PESSEMIER, E. A. Complementary and substitute patterns of purchasing and use. *Journal of Advertising Research 9*, 2 (1969), 19–27.

[11] BERTSEKAS, D. *Network optimization: continuous and discrete models*. Athena Scientific, 1998. Chapter 3–4, pp. 115–164.

[12] BIAN, F., KEMPE, D., AND GOVINDAN, R. Utility-based sensor selection. In *Proceedings of the IEEE Conference on Information Processing in Sensor Network (IPSN 2006)* (April 2006).

[13] BISDIKIAN, C. On sensor sampling and quality of information: A starting point. In *3rd IEEE Int'l Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2007), a PerCom'07 Workshop* (White Plains, NY).

[14] BRESLIN, J., DECKER, S., HAUSWIRTH, M., HYNES, G., LE PHUOC, D., PASSANT, A., POLLERES, A., RABSCH, C., AND REYNOLDS, V. Integrating social networks and sensor networks. In *W3C Workshop on the Future of Social Networking* (Jan. 2009).

[15] BYERS, J., AND NASSER, G. Utility-based decision-making in wireless sensor networks. In *First Annual Workshop on Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC* (2000), IEEE, pp. 143–144.

[16] CAI, Y., LOU, W., LI, M., AND LI, X. Target-Oriented scheduling in directional sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE* (2007), pp. 1550–1558.

[17] CARDEI, M., AND DU, D. Improving wireless sensor network lifetime through power aware organization. *ACM Wireless Networks 11*, 3 (May 2005), 333–340.

[18] CHAIMOWICZ, L., CAMPOS, M. F., AND KUMAR, V. Dynamic role assignment for cooperative robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE Conference on* (2002), vol. 1, pp. 293–298.

[19] CHATTERJEE, R., TAKAO, I., MATSUNO, F., AND TADOKORO, S. Robot description ontology and bases for surface locomotion evaluation. In *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International* (June 2005), IEEE, pp. 242– 247.

[20] CHATZIGIANNAKIS, I., MYLONAS, G., AND NIKOLETSEAS, S. 50 ways to build your application: A survey of middleware and systems for wireless sensor networks. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on* (2007), pp. 466–473.

[21] CHEN, F., PIZZOCARO, D., JOHNSON, M. P., BAR-NOY, A., PREECE, A., AND LA PORTA, T. Broadcast scheduling with data bundles. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR II* (2011), vol. 8047, SPIE, pp. 80470W–80470W–8.

[22] COHEN, R., KATZIR, L., AND RAZ, D. An efficient approximation for the generalized assignment problem. *Information Processing Letters 100(4)* (2006), 162–166.

[23] COLOZZA, A., AND DOLCE, J. Initial feasibility assessment of a high altitude long endurance airship. *NASA/CR Technical Report 212724* (2003), 1 – 6.

[24] DANG, V. D., DASH, R. K., ROGERS, A., AND JENNINGS, N. R. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Twenty-First National Conference on Artificial Intelligence (AAAI-06)* (2006), pp. 635–640.

[25] D'AQUIN, M., NIKOLOV, A., AND MOTTA, E. How much semantic data on small devices? In *Knowledge Engineering and Management by the Masses*, vol. 6317. Springer, Berlin, Heidelberg, 2010, pp. 565–575.

[26] DAS, A., AND KEMPE, D. Sensor selection for minimizing Worst-Case prediction error. In *International Conference on Information Processing in Sensor Networks, 2008. IPSN '08* (Apr. 2008), IEEE, pp. 97–108.

[27] DE MEL, G., SENSOY, M., VASCONCELOS, W., AND PREECE, A. D. Flexible resource assignment in sensor networks: A hybrid reasoning approach. *1st International Workshop on the Semantic Sensor Web SemSensWeb 2009 2009*, June (2009), 1–15.

[28] DE VRIES, S., AND VOHRA, R. Combinatorial auctions: a survey. *INFORMS Journal on Computing 15-3* (2003), 284–309.

[29] DELLE FAVE, F. M., ROGERS, A., XU, Z., SUKKARIEH, S., AND JENNINGS, N. Deploying the Max-Sum algorithm for coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2012), IEEE.

[30] DERTOUZOS, M. L., AND MOK, A. K. Multiprocessor online scheduling of hard-real-time tasks. *IEEE Transactions on Software Engineering 15*, 12 (Dec. 1989), 1497–1506.

[31] DJURIC, P., VEMULA, M., AND BUGALLO, M. Target tracking by particle filtering in binary sensor networks. *IEEE Transactions on Signal Processing 56*, 6 (2008), 2229–2238.

[32] FANELLI, L., FARINELLI, A., IOCCHI, L., NARDI, D., AND SETTEMBRE, G. P. Ontology-based coalition formation in heterogeneous MRS. In *Proceedings of the 2006 international symposium on Practical cognitive agents and robots* (Perth, Australia, 2006), ACM, pp. 105–116.

[33] FARINELLI, A., ROGERS, A., PETCU, A., AND JENNINGS, N. R. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2* (Richland, SC, 2008), AAMAS '08, International Foundation for Autonomous Agents and Multiagent Systems, pp. 639–646.

[34] FITZPATRICK, S., AND MEERTENS, L. Distributed coordination through anarchic optimization. In *Distributed Sensor Networks*, V. Lesser, C. L. Ortiz, M. Tambe, and G. Weiss, Eds., vol. 9 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer US, 2003, pp. 257–295.

[35] FLEISCHER, L., GOEMANS, M. X., MIRROKNI, V. S., AND SVIRIDENKO, M. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* (New York, NY, USA, 2006), SODA '06, ACM, pp. 611–620.

[36] FOTAKIS, D., AND SPIRAKIS, P. G. Minimum congestion redundant assignments to tolerate random faults. *Algorithmica 32*, 3 (2002), 396–422.

[37] FRANK, C., AND RÖMER, K. Algorithms for generic role assignment in wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), SenSys '05, ACM, pp. 230–242.

[38] FUJISHIMA, Y., LEYTON-BROWN, K., AND SHOHAM, Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate ap-

proaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (1999), Morgan Kaufmann Publishers Inc., pp. 548–553.

[39] GAREY, M., AND JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-Colmpleteness*. Freeman, 1979.

[40] GERKEY, B., AND MATARIC, M. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research 23*, 9 (2004), 939.

[41] GFELLER, B., MIHALÁK, M., SURI, S., VICARI, E., AND WIDMAYER, P. Angle optimization in target tracking. In *Algorithm Theory – SWAT 2008* (2008), vol. 5124, pp. 65–76.

[42] GIBSON, M., KANADE, G., KROHN, E., AND VARADARAJAN, K. Quasi-Polynomial time approximation schemes for target tracking. *arXiv:0907.1080* (July 2009).

[43] GIORDANO, S., AND PUCCINELLI, D. The human element as the key enabler of pervasiveness. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2011 The 10th IFIP Annual Mediterranean* (June 2011), IEEE, pp. 150–156.

[44] GOMEZ, M., PREECE, A. D., JOHNSON, M. P., DE MEL, G., VASCONCELOS, W., GIBSON, C., BAR-NOY, A., BOROWIECKI, K., LA PORTA, T. F., PIZZOCARO, D., ROWAIHY, H., PEARSON, G., AND PHAM, T. An ontology-centric approach to sensor-mission assignment. In *EKAW'08* (2008), vol. 5268 of *Lecture Notes in Computer Science*, Springer, pp. 347–363.

[45] GOODCHILD, M. F. Citizens as sensors: the world of volunteered geography. *GeoJournal 69*, 4 (Nov. 2007), 211–221.

[46] GOOSSENS, D., POLYAKOVSKIY, S., SPIEKSMA, F. C. R., AND WOEGINGER, G. J. The focus of attention problem. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia,

PA, USA, 2010), SODA '10, Society for Industrial and Applied Mathematics, pp. 312–317.

[47] GRADWELL, P., AND PADGET, J. A comparison of distributed and centralised agent based bundling systems. In *Proceedings of the ninth international conference on Electronic commerce* (New York, NY, USA, 2007), ICEC '07, ACM, pp. 25–34.

[48] HANSEN, R. C. *Phased array antennas*, vol. 213. Wiley-Interscience, 2009.

[49] HEFEEDA, M., AND AHMADI, H. A probabilistic coverage protocol for wireless sensor networks. In *IEEE International Conference on Network Protocols, 2007. ICNP 2007* (Oct. 2007), IEEE, pp. 41–50.

[50] HORLING, B., VINCENT, R., MAILLER, R., SHEN, J., BECKER, R., RAWLINS, K., AND LESSER, V. Distributed sensor network for real time tracking. In *Proceedings of the fifth international conference on Autonomous agents* (Montreal, Quebec, Canada, 2001), AGENTS '01, ACM, pp. 417–424.

[51] IBARRA, O. H., AND KIM, C. E. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM) 22*, 4 (1975), 463–468.

[52] ISLER, V., AND BAJCSY, R. The sensor selection problem for bounded uncertainty sensing models. In *Proceedings of the 4th international symposium on Information processing in sensor networks* (Piscataway, NJ, USA, 2005), IPSN '05, IEEE Press.

[53] ISLER, V., KHANNA, S., SPLETZER, J. R., AND TAYLOR, C. J. Target tracking with distributed sensors: The focus of attention problem. *Computer Vision and Image Understanding 100*, 1-2 (2005), 225–247.

[54] JOHNSON, M. P., AND BAR-NOY, A. Pan and scan: Configuring cameras for coverage. In *2011 Proceedings IEEE INFOCOM* (Apr. 2011), IEEE, pp. 1071–1079.

[55] JOHNSON, M. P., ROWAIHY, H., PIZZOCARO, D., BAR-NOY, A., CHALMERS, S., LA PORTA, T., AND PREECE, A. Frugal sensor assignment. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS 2008)* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 219–236.

[56] JOHNSON, M. P., ROWAIHY, H., PIZZOCARO, D., BAR-NOY, A., CHALMERS, S., LA PORTA, T., AND PREECE, A. Sensor-Mission assignment in constrained environments. *IEEE Transactions on Parallel and Distributed Systems 21*, 11 (Nov. 2010), 1692–1705.

[57] KADAR, I. Optimum geometry selection for sensor fusion. In *Signal Processing, Sensor Fusion, and Target Recognition VII* (Orlando, FL, USA, July 1998), vol. 3374, SPIE, pp. 96–107.

[58] KALYANASUNDARAM, B., AND PRUHS, K. Online weighted matching. *Journal of Algorithms 14* (1993), 478–488.

[59] KAPLAN, L. Global node selection for localization in a distributed sensor network. *IEEE Transactions on Aerospace and Electronic Systems 42*, 1 (January 2006), 113–135.

[60] KAPLAN, L. Local node selection for localization in a distributed sensor network. *IEEE Transactions on Aerospace and Electronic Systems 42*, 1 (January 2006), 136–146.

[61] KAPLAN, L. M., AND LE, Q. On exploiting propagation delays for passive target localization using bearings-only measurements. *Journal of the Franklin Institute 342*, 2 (Mar. 2005), 193–211.

[62] KARAT, J., SIECK, W., NORMAN, T. J., KARAT, C., BRODIE, C., RASMUSSEN, L., AND SYCARA, K. A framework for culturally adaptive policy management in ad hoc collaborative contexts. In *Proceeding of the 2009 international workshop on Intercultural collaboration* (New York, NY, USA, 2009), IWIC '09, ACM, pp. 257–260.

[63] KELLY, A. Precision dilution in triangulation-based mobile robot position estimation. In *Proceedings of Intelligent Autonomous Systems* (Amsterdam, 2003).

[64] KITANO, H., TADOKORO, S., NODA, I., MATSUBARA, H., TAKAHASHI, T., SHINJOU, A., AND SHIMADA, S. RoboCup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *1999 IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings* (1999), vol. 6, IEEE, pp. 739–743 vol.6.

[65] KOHVAKKA, M., SUHONEN, J., KUORILEHTO, M., KASEVA, V., HÄNNIKÄINEN, M., AND HÄMÄLÄINEN, T. D. Energy-efficient neighbor discovery protocol for mobile wireless sensor networks. *Ad Hoc Networks 7* (January 2009), 24–41.

[66] KRAUS, J. *Antennas*. McGraw-Hill Education, May 1988. Chapter 2, 2-6 Radiation Intensity, p. 25.

[67] LASNIA, D., BROERING, A., JIRKA, S., AND REMKE, A. Crowdsourcing sensor tasks to a Socio-Geographic network. In *AGILE 2010: The 13th AGILE International Conference on Geographic Information Science* (2010).

[68] LEHMANN, B., LEHMANN, D., AND NISAN, N. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior 55*, 2 (May 2006), 270–296.

[69] LERMAN, K., JONES, C., GALSTYAN, A., AND MATARIC, M. J. Analysis of dynamic task allocation in Multi-Robot systems. *The International Journal of Robotics Research 25*, 3 (Mar. 2006), 225 –241.

[70] LESSER, V., ORTIZ, C. L., AND TAMBE, M. *Distributed sensor networks: a multiagent perspective*. Springer, July 2003.

[71] LEVIS, P., LEE, N., WELSH, M., AND CULLER, D. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems* (New York, NY, USA, 2003), SenSys '03, ACM, pp. 126–137.

[72] LIM, H. B., WANG, B., FU, C., PHULL, A., AND MA, D. A middleware services simulation platform for wireless sensor networks. In *28th International Conference on Distributed Computing Systems Workshops, 2008. ICDCS '08* (June 2008), IEEE, pp. 168–173.

[73] LOW, K. H., LEOW, W., AND ANG, M. Autonomic mobile sensor network with self-coordinated task allocation and execution. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 36*, 3 (May 2006), 315–327.

[74] LU, J., BAO, L., AND SUDA, T. Coverage-Aware sensor engagement in dense sensor networks. In *Embedded and Ubiquitous Computing – EUC 2005*, vol. 3824. Springer Berlin Heidelberg, 2005, pp. 639–650.

[75] MEDIONI, G., COHEN, I., BREMOND, F., HONGENG, S., AND NEVATIA, R. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 8 (2001), 873–889.

[76] MODI, P. J., SHEN, W., TAMBE, M., AND YOKOO, M. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence 161*, 1Ð2 (Jan. 2005), 149–180.

[77] MULLEN, T., AVASARALA, V., AND HALL, D. L. Customer-driven sensor management. *IEEE Intelligent Systems 21*, 2 (Mar/April 2006), 41–49.

[78] NARUMANCHI, M. V., AND VIDAL, J. M. Algorithms for distributed winner determination in combinatorial auctions. In *Agent-Mediated Electronic Commerce. Designing Trading Agents and Mechanisms*, vol. 3937. Springer Berlin Heidelberg, 2006, pp. 43–56.

[79] NGUYEN, H. A., AND GIORDANO, S. Context information prediction for social-based routing in opportunistic networks. *Ad Hoc Networks* (2011). In press.

[80] NISAN, N. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM conference on Electronic commerce* (Minneapolis, Minnesota, United States, 2000), ACM, pp. 1–12.

[81] NORMAN, T. J., PREECE, A., CHALMERS, S., JENNINGS, N. R., LUCK, M., DANG, V. D., NGUYEN, T. D., DEORA, V., SHAO, J., GRAY, W. A., AND FIDDIAN, N. J. Agent-based formation of virtual organisations. *Knowledge-Based Systems 17*, 2-4 (May 2004), 103–111.

[82] OSTWALD, J., LESSER, V., AND ABDALLAH, S. Combinatorial auctions for resource allocation in a distributed sensor network. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium, 2005. (RTSS 2005)* (December 2005), pp. 266–274.

[83] PAPADIMITRIOU, C., AND STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[84] PARKER, L. E. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on 14*, 2 (1998), 220–240.

[85] PARKER, L. E., AND TANG, F. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE 94*, 7 (July 2006), 1289–1305.

[86] PATHAK, A., AND PRASANNA, V. K. Energy-efficient task mapping for data-driven sensor network macroprogramming. In *DCOSS* (2008), vol. 5067 of *Lecture Notes in Computer Science*, Springer, pp. 516–524.

[87] PEARSON, G. A vision of network-centric ISTAR and the resulting challenges. vol. 6963, SPIE, pp. 696302–696302–11.

[88] PEARSON, G., AND PHAM, T. The challenge of sensor information processing and delivery within network and information science research. vol. 6981, SPIE, pp. 698105–698105–9.

[89] PERILLO, M. A., AND HEINZELMAN, W. B. Optimal sensor management under energy and reliability constraints. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003* (Mar. 2003), vol. 3, IEEE, pp. 1621–1626.

[90] PETCU, A., AND FALTINGS, B. DPOP: A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence* (2005), vol. 19, pp. 266–271.

[91] PETCU, A., AND FALTINGS, B. MB-DPOP: a new memory-bounded algorithm for distributed optimization. In *Proceedings of the 20th international joint conference on Artifical intelligence* (San Francisco, CA, USA, 2007), IJCAI'07, Morgan Kaufmann Publishers Inc., pp. 1452–1457.

[92] PHAM, T., CIRINCIONE, G. H., VERMA, D., AND PEARSON, G. Intelligence, surveillance, and reconnaissance fusion for coalition operations. In *2008 11th International Conference on Information Fusion* (July 2008), IEEE, pp. 1–8.

[93] PIZZOCARO, D. *Sensor Deployment in a Battlefield and Sensor Assignment to Competing Missions.* Universita' degli Studi di Padova, Italy, October 2007. Master Thesis.

[94] PIZZOCARO, D., JOHNSON, M. P., ROWAIHY, H., CHALMERS, S., PREECE, A., BAR-NOY, A., AND LA PORTA, T. A knapsack approach to sensor-mission assignment with uncertain demands. SPIE, pp. 711205–711205–12.

[95] PIZZOCARO, D., AND PREECE, A. Towards a taxonomy of task allocation in sensor networks. In *Proceedings of the 28th IEEE international conference on Computer Communications (INFOCOM 2009) Workshops* (Rio de Janeiro, Brazil, Apr. 2009), IEEE, pp. 413–414.

[96] PIZZOCARO, D., PREECE, A., BOROWIECKI, K., CHEN, F., LA PORTA, T., JOHNSON, M., AND BAR-NOY, A. System architectures for Multi-Sensor task allocation. In *Proceedings of the 4th Annual Conference of the International Technology Alliance (ACITA 2010)* (2010).

[97] PIZZOCARO, D., PREECE, A., CHEN, F., LA PORTA, T., AND BAR-NOY, A. A distributed architecture for heterogeneous multi sensor-task allocation. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011)* (June 2011), IEEE, pp. 1–8.

[98] PIZZOCARO, D., PREECE, A., CHEN, F., LA PORTA, T., AND BAR-NOY, A. A distributed architecture for heterogeneous multi sensor-task allocation: Demo. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011)* (Barcelona, Spain, June 2011), IEEE.

[99] POSER, K., AND DRANSCH, D. Volunteered geographic information for disaster management with application to rapid flood damage estimation. *Geomatica 64*, 1 (2010), 89–98.

[100] RAMCHURN, S. D., POLUKAROV, M., FARINELLI, A., TRUONG, C., AND JENNINGS, N. R. Coalition formation with spatial and temporal constraints.

In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 3-Volume 3* (2010), pp. 1181–1188.

[101] RAUENBUSCH, T. The mediation algorithm for real time negotiation. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3* (New York, NY, USA, 2002), AAMAS '02, ACM, pp. 1139–1140.

[102] ROGERS, A., FARINELLI, A., STRANDERS, R., AND JENNINGS, N. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence 175*, 2 (Feb. 2011), 730–759.

[103] ROTHKOPF, M. H., PEKEÈ, A., AND HARSTAD, R. M. Computationally manageable combinational auctions. *Management Science 44*, 8 (Aug. 1998), 1131–1147.

[104] ROWAIHY, H., ESWARAN, S., JOHNSON, M., VERMA, D., BAR-NOY, A., BROWN, T., AND LA PORTA, T. A survey of sensor selection schemes in wireless sensor networks. vol. 6562, SPIE, pp. 65621A–65621A–13.

[105] ROWAIHY, H., JOHNSON, M. P., ESWARAN, S., PIZZOCARO, D., BAR-NOY, A., LA PORTA, T., MISRA, A., AND PREECE, A. Utility-based joint sensor selection and congestion control for task-oriented WSNs. In *2008 42nd Asilomar Conference on Signals, Systems and Computers* (Oct. 2008), IEEE, pp. 1165–1169.

[106] ROWAIHY, H., JOHNSON, M. P., LIU, O., BAR-NOY, A., BROWN, T., AND PORTA, T. L. Sensor-mission assignment in wireless sensor networks. *ACM Transaction on Sensor Networks 6*, 4 (July 2010), 36:1–36:33.

[107] ROWAIHY, H., JOHNSON, M. P., PIZZOCARO, D., BAR-NOY, A., KAPLAN, L., LA PORTA, T., AND PREECE, A. Detection and localization sensor assignment with exact and fuzzy locations. In *Proceedings of the 5th IEEE Interna-*

tional Conference on Distributed Computing in Sensor Systems (DCOSS 2009) (Berlin, Heidelberg, June 2009), Springer-Verlag, pp. 28–43.

[108] SANDHLOM, T., SURI, S., GILPIN, A., AND LEVINE, D. CABOB: a fast optimal algorithm for combinatorial auctions. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2* (Seattle, WA, USA, 2001), Morgan Kaufmann Publishers Inc., pp. 1102–1108.

[109] SANDHLOM, T. W., AND LESSER, V. R. T. Coalitions among computationally bounded agents. *Artificial Intelligence 94*, 1-2 (July 1997), 99–137.

[110] SCERRI, P., FARINELLI, A., OKAMOTO, S., AND TAMBE, M. Allocating tasks in extreme teams. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2005), ACM, pp. 727–734.

[111] SCHOENHARL, T., BRAVO, R., AND MADEY, G. Wiper: Leveraging the cell phone network for emergency response. *International Journal of Intelligent Control and Systems 11* (2007), 2006.

[112] SCHOOMER, B. A. The incorporation of step functions and ramp functions into a linear programming model. *Operations Research 12*, 5 (Sept. 1964), 773–777.

[113] SEKHAR, A., MANOJ, B. S., AND MURTHY, C. S. Dynamic coverage maintenance algorithms for sensor networks with limited mobility. In *Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005* (Mar. 2005), IEEE, pp. 51–60.

[114] SENSOY, M., LE, T., VASCONCELOS, W. W., NORMAN, T. J., AND PREECE, A. D. Resource determination and allocation in sensor networks: A hybrid approach. *The Computer Journal 54*, 3 (Mar. 2011), 356 –372.

[115] SHEHORY, O., AND KRAUS, S. Methods for task allocation via agent coalition formation. *Artificial Intelligence 101*, 1-2 (May 1998), 165–200.

[116] SHIH, K., CHEN, Y., CHIANG, C., AND LIU, B. A distributed active sensor selection scheme for wireless sensor networks. In *11th IEEE Symposium on Computers and Communications, 2006. ISCC '06. Proceedings* (June 2006), IEEE, pp. 923–928.

[117] SLOGGETT, D. Intelligence support to contemporary information operations. *IO Sphere* (Spring 2007), 19 – 23.

[118] SMART, P., MOTT, D., GENTLE, E., BRAINES, D., SIECK, W., POLTROCK, S., HOUGHTON, P., PREECE, A., NIXON, M., STRUB, M., ROBERTS, D., VERMA, D., AND SHADBOLT, N. Holistan revisited: Demonstrating agent- and Knowledge-Based capabilities for future coalition military operations. In *Proceedings of the Second Annual Conference of the International Technology Alliance* (2008).

[119] SUNG, S. C., AND VLACH, M. Maximizing weighted number of Just-in-Time jobs on unrelated parallel machines. *Journal of Scheduling 8*, 5 (Oct. 2005), 453–460.

[120] TANG, F., AND SAHA, S. An anytime winner determination algorithm for time-extended multi-robot task allocation. In *Proceedings of the International Conference on Automation, Robotics, and Control Systems (ARCS)* (2008), pp. 123–130.

[121] TUTTON, S. J. Optimizing the allocation of sensor assets for the unit of action. Tech. rep., Naval Postgraduate School, California, 2006.

[122] VAZIRANI, V. V. *Approximation Algorithms*. Springer Verlag, 2001.

[123] VINYALS, M., RODRIGUEZ-AGUILAR, J. A., AND CERQUIDES, J. A survey on sensor networks from a multiagent perspective. *The Computer Journal 54*, 3 (Mar. 2011), 455–470.

[124] WANG, G., CAO, G., AND LA PORTA, T. A bidding protocol for deploying mobile sensors. In *11th IEEE International Conference on Network Protocols, 2003. Proceedings* (Nov. 2003), IEEE, pp. 315– 324.

[125] WANG, H., YAO, K., POTTIE, G., AND ESTRIN, D. Entropy-based sensor selection heuristic for target localization. In *Proceedings of the 3rd international symposium on Information processing in sensor networks. IPSN 2004* (Berkeley, California, USA, 2004), ACM, pp. 36–45.

[126] WANG, R., MULLEN, T., AVASARALA, V., AND YEN, J. A Market-Based adaptation for resolving competing needs for scarce resources. In *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on* (2006), pp. 350–356.

[127] WEISS, C., BERNSTEIN, A., AND BOCCUZZO, S. i-MoCo : Mobile conference guide – storing and querying huge amounts of semantic web data on the iPhone / iPod touch. *7th International Semantic Web Conference* (2008), 8.

[128] WEISS, C., KARRAS, P., AND BERNSTEIN, A. Hexastore: sextuple indexing for semantic web data management. *Proceedings VLDB Endowment 1*, 1 (Aug. 2008), 1008–1019.

[129] WERNER-ALLEN, G., LORINCZ, K., RUIZ, M., MARCILLO, O., JOHNSON, J., LEES, J., AND WELSH, M. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing 10*, 2 (Apr. 2006), 18–25.

[130] YARVIS, M., KUSHALNAGAR, N., SINGH, H., RANGARAJAN, A., LIU, Y., AND SINGH, S. Exploiting heterogeneity in sensor networks. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE 2* (March 2005), 878–890.

[131] YEOH, W., FELNER, A., AND KOENIG, S. BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm. In *Proceedings of the 7th international*

joint conference on Autonomous agents and multiagent systems - Volume 2 (Richland, SC, 2008), AAMAS '08, International Foundation for Autonomous Agents and Multiagent Systems, pp. 591–598.

[132] YU, Y., AND PRASANNA, V. K. Energy-Balanced task allocation for collaborative processing in wireless sensor networks. *Mobile Networks and Applications 10*, 1/2 (Feb. 2005), 115–131.

[133] ZHAO, F., SHIN, J., AND REICH, J. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine 19*, 2 (March 2002), 61–72.