Harnessing coherence of area decomposition and semantic shared spaces for task allocation in a robotic fleet

Domagoj Drenjanac a,*, Slobodanka Dana Kathrin Tomic a, Lukas Klausner b, Eva Kühn b

a The Telecommunications Research Center Vienna (FTW), Donau-City-Strasse 1, 1220 Vienna, Austria
b Institute of Computer Languages, Vienna University of Technology, Argentinierstrasse 8, 1040 Vienna, Austria

ABSTRACT

Task allocation is a fundamental problem in multi-robot systems where heterogeneous robots cooperate to perform a complex mission. A general requirement in a task allocation algorithm is to find an optimal set of robots to execute a certain task. This paper presents the work that harnesses an area decomposition algorithm, and a space-based middleware to facilitate task allocation process in unstructured and dynamic environments. To reduce spatial interference between robots, area decomposition algorithm divides a working area into cells which are then dynamically assigned to robots. In addition, coordination and collaboration among distributed robots are realized through a space-based middleware. For this purpose, the space-based middleware is extended with a semantic model of robot capabilities to improve task selection in terms of flexibility, scalability, and reduced communication overhead during task allocation. In this way a framework which exploits the synergy of area decomposition and semantically enriched space-based approach is created.

We conducted performance tests in a specific precision agriculture use case focusing on the utilization of a robotic fleet for weed control introduced in the European Project RHEA – Robot Fleets for Highly Effective Agriculture and Forestry Management.

© 2014 China Agricultural University. Production and hosting by Elsevier B.V. All rights reserved.

1. Introduction

Today, cooperating robots are commonly used in controlled and structured environments, such as factories, where they are managed from a central place that supervises mission execution. Due to the advances in the perception and locomotion technology, there is a great potential to use multiple cooperating robots in heterogeneous and unstructured environments. This however, imposes new requirements on communication and coordination of actions in teams, and the well-established centralized coordination approach needs to either be enhanced or replaced with a distributed approach.

Task allocation is a fundamental problem in multi-robot systems where the core requirement is to find an optimal set of heterogeneous robots that have to cooperate in order to execute a complex mission [1]. Task allocation is well known to be an NP-hard problem in multi-agent systems,
leading to a variety of different heuristic-based approaches [2]. A critical enabler for distributed task allocation is an efficient coordination. This work examines how two different coordination approaches, an area decomposition and a space-based middleware, fit together and what their contribution in a task allocation domain can be.

We propose an algorithm for area decomposition based on a computational geometry technique of Voronoi diagram [3]. Due to the constraints posed by an unstructured and volatile environment, we adapted the construction of Voronoi cells to fit our requirements while retaining the original notion. This approach is applicable to domains where geographic positions of robots and tasks are known, which to a great extent corresponds to our agricultural use case [4]. A complementary technique, the space-based middleware, defines coordination model based on a centralized tuple space with a shared message repository, exploiting generative communication among processes. This work extends coordination capabilities of the space-based middleware XVS (eXtensible Virtual Shared Memory) [5,6], in particular its Java-based implementation MozartSpaces. XVSM is based on a Linda tuple space model [7]. Our framework, Semantic MozartSpaces [8] introduces a new data description and query model based on RDF (Resource Description Framework) [9] and SPARQL [10], where RDF is used to construct nested blank nodes in a triple store which was implemented in Jena [11] and SPARQL is used for query and update interactions. To evaluate the performance of the integrated area decomposition algorithm and semantically enhanced MozartSpaces, we conducted series of tests in a specific precision agriculture use case.

The remainder of this paper is structured as follows: Section 2 summarizes related work. Section 3 proposes the presented system architecture and Section 4 provides the implementation details. Section 5 introduces the use case while Section 6 evaluates the framework. Finally, Section 7 concludes the paper and presents future work.

2. Related work

Related work is structured in three parts. The first part discusses the task allocation as a fundamental, ubiquitous, and well-known problem in a multi robot domain. Our focus here is on utilizing semantic technologies for task allocation. This discussion is followed by a related work on various area decomposition approaches, and a systematication of some prominent space-based frameworks.

The use of semantics in task and resource modeling in robotic systems is an emergent research field. In [12] authors explore how semantic description of environments, objects, and tasks can be used to improve task planning in complex scenarios where a robot executes tasks on objects in an unstructured environment with a great number of objects. In [13] authors study a combination of the Web Service paradigm and ontology modeling for service discovery, service composition, and a task allocation. In their solution, all entities expose their functionalities as semantic Web Services allowing their discovery and composition. In SERA [14], tasks and resources are semantically described following resource description ontology. Performance tests with the SERA framework show that the centralized approach performs better than the distributed one when the number of resources is low, which is attributed to the negotiation overhead in distributed systems. In [15], an author compares the performance of the proposed semantic based matchmaking approach against the conventional keyword based matchmaking in a grid environment, and concludes that the semantic based matchmaking mechanism retrieves more closely matching resources. Authors in [16] propose a framework for semantic service discovery in a dynamic and changing environment. Similar to our implementation, the use of context information, such as a current location which facilitates the matching process. Contrary to our approach, the framework in [16] invokes and discovers services locally on robots. In our work robots query the central task repository for matching tasks descriptions, and a semantic matchmaking is performed based on the description of robots’ services, and locations of tasks and robots. In our framework, when a detailed task description is generated, the execution of an area decomposition algorithm is triggered. Two-level filtration mechanism of advertised content is proposed in [17]. In the first level, the broker agent applies a semantic-based mechanism which compares a content requested by users to that advertised by providers. On the second level, the best content provider in terms of both price and quality is selected. Our work differs from the reviewed work as we use semantic approach for both the resource and task modelling. Rational for using semantics is twofold: (1) it provides a basis for automatic mapping between task requirements and available resources and thus makes the whole process more flexible, and (2) it provides a general task description language that most of the reviewed frameworks lack.

With our area decomposition approach we address a problem of robots’ spatial interference which is perceived as a key stumbling block in the way to efficient robotic fleets. Reflecting on the experiments conducted in [18], the authors concluded that the larger the number of robots working in the same global workspace area, the grater the interference and the uncertainty related to the time required for task execution. The works presented in [19,20] utilize the bouystrophedon cellular decomposition approach for partitioning the robots workspace. The presented approach exploits a geometric structure which is a union of non-intersecting rectangular regions that together compose the working environment. Each region is termed a cell and in each cell a coverage path is a simple back-and-forth motion. In [21] authors develop a dynamic partitioning algorithm which assigns subareas to robots during the mission. The authors argue that this dynamic approach is more flexible than the static one because in a case of a robot’s failure, other robots dynamically take over his work. Authors in [4] base their task allocation algorithm on computational geometry techniques, i.e., Delaunay triangulation. Their approach is applicable to domains where geographical positions of robots and tasks are known. Voronoi diagram as another technique for space partitioning is discussed in [22]. Our space decomposition algorithm uses customized mathematical model of the Voronoi diagram. We adapted Voronoi diagram model to the requirements of our domain, i.e., to linear trajectories of robots in agricultural fields. Due to the dynamic environment where our robots
operate, our algorithm supports dynamic partitioning which assigns subareas to robots during the mission.

Several research projects as well as commercial products have adopted the space-based model to construct robust coordination platforms. The Linda tuple space model promoted the shared space-based coordination model, where the tuple space consists of tuples that can be concurrently accessed by several processes. Semantic Tuple Spaces [23] (sTuples) use a web ontology language (OWL) [24] for describing and matching resources. Semantic Web Spaces [25] are based on Linda-inspired coordination model integrated with core semantic technologies such as RDF and it is utilized for coordination between agents sharing semantic data. Our work is based on an open source implementation of the Space Container approach called MozartSpaces [26]. The comparative advantage of the MozartsSpaces over other space-based implementations is that it structures the space into containers that can store entries (tuples) using many different coordination patterns. Our semantic extension of the MozartSpaces borrows some concepts from the Semantic Tuple Centres [27], which treats a semantic tuple as an object of an application domain. However, while Semantic Tuple Centres has its own data format for exchanging semantic tuples and queries, we use Turtle [28] and SPARQL. In contrast, in our solution tuples do not have a mandatory public reference.

3. Proposed system architecture

The proposed system architecture is shown in Fig. 1. Three main components are: (1) an ontology for describing resources, i.e., tasks and services offered by robots (2) an Area Decomposition algorithm, and (3) Space-Based Middleware with a semantic extension.

The system ontology we proposed reuses the concepts form ontologies described in [15,29] which partially describe our system, but are still incomplete. Therefore, some changes were required to make these ontologies suitable for task allocation. The basic idea of our proposed model is to formally specify that Tasks can be performed by Executors (robots carrying different implements), and that each Task can be associated with an Executor via Skill, ResourceAmount and Point, which denotes location in a field. Proposed ontology is depicted in Fig. 2.

The area decomposition algorithm is responsible for the workspace partitioning, i.e., partitioning of the agricultural field. Partitioning or area decomposition process divides the field area in smaller cells, based on task locations, in order to make cells suitable for single-robot operations. The task locations, together with task types, are provided by the user. The area decomposition algorithm uses a customized version of Voronoi diagram, where the algorithm takes into account constraints on allowed types of trajectories in agricultural fields. Due to the fact that tractors follow parallel trajectories in a field (e.g., along the crop rows), our algorithm generates squared cells. This differs from the default Voronoi cell, which is a polygon of an arbitrary shape. The outcome of the algorithm is a list of cells with contained tasks.

The task producer receives a list of cells with tasks from the area decomposition algorithm. Upon receiving all cells, the producer builds task objects according to an ontology shared between producer, Semantic MozartSpaces, and an executor. The task producer stores tasks by using middleware functionalities making them available for the executor: the producer first obtains an instance of the Semantic MozartSpaces which has a twofold purpose: (1) to translate task objects received from the area decomposition component into RDF triples, and (2) to write RDF triples embedded within the entry objects into the instance of the Semantic MozartSpaces. Writing an entry into the Semantic MozartSpaces denotes the end of a one task production cycle.

Semantic MozartSpaces with SPARQL interface and an underlying triple store (Jena) is a core component in our adaptation of the Space-Based Middleware. The main part is the data model that exposes the mapping process between MozartSpaces and semantic entries. The basic concept is still a container hosted in a single runtime instance of the space where a container is addressable by URL and therefore can be accessed as any other resource on the Internet. A container hosts different entries where the value of an entry is an object with several properties, which themselves can be either literals or objects. As inherited from the core implementation, Semantic MozartSpaces offers a multitude of coordination patterns for retrieving stored entries, e.g., First-In First-Out (FIFO), Last-In First-Out (LIFO), Random, Key coordinators. The task executor component represents a robot with accompanying resources which are described using the provided ontology. Each robot offers a service, i.e., based on an implement (physical device) for executing a special type of a task, and amount of available resources, e.g., amount of a liquid for spraying tasks. The task executor runs a task selection algorithm wrapped in a SPARQL query. The task selection algorithm implements an ontology based
matchmaking mechanism that determines semantic relationship between the advertised task descriptions and services offered by a robot. The algorithm uses three parameters for selecting a matching task: (1) types of services requested by advertised task, (2) an amount of requested resource, and (3) an executor’s distance from a task. As a result of the query, the Semantic MozartSpaces returns an entry with task description which satisfies requirements stated in the query. The executor also has to obtain an instance of the Semantic MozartSpaces for executing queries on the triple store and for translating semantic entries into task objects.

We reused semantic relationship notion between the requested and advertised resources from [15–17] to describe their matching degree: (1) exact, services (skills plus resource amount) offered by an executor exactly match the resources (skills plus resource amount) requested by a task, (2) subsume, if the services offered by the executor have more capabilities than that of advertised task, (3) plugin, opposite from the previous case, the advertised task expects more capabilities than the services which are offered by the executor have, and (4) fail if none of the above conditions hold. The constructed query determines the semantic relationship between the offered services and advertised tasks.

4. Implementation details

The framework has been implemented in Java, and it consists of two main modules: (1) area decomposition algorithm implemented as a standalone Java application, and (2) Semantic MozartSpaces as a representative of a space-based middleware.

4.1. Dynamic area decomposition

We propose an algorithm for area decomposition based on computational geometry technique Voronoi diagram [3]. The approach is applicable to domains in which agents’ and tasks’ geographical positions are known. In our scenario, the tasks are stationary, but agents are allowed to move.

Construction of Voronoi diagram is based on the Euclidian distance between two points p and q by \( \text{dist}(p,q) \). Let \( P = \{p_1, p_2, \ldots, p_n\} \) be a set of n distinct points in the plane, where points are the sites. Voronoi diagram of \( P \) is defined as the subdivision of the plane into \( n \) cells, one for each site in \( P \), with a property that a point \( q \) lies in the cell corresponding to a site \( p_i \) if and only if \( \text{dist}(q,p_i) < \text{dist}(q,p_j) \) for each \( p_j \in P \) with \( j \neq i \).

Our area decomposition algorithm considers the set of \( n \) tasks \( T = \{t_1, t_2, \ldots, t_n\} \) as points in \( P \). Each task is composed of a random number of equal square building blocks. Since tasks can have different shapes and sizes, for each task \( t_i \in T, 1 \leq i \leq n \) we calculate the point \( r_i(x,y) \) presenting the center of mass. Those centers of mass are used for calculating the Euclidian distance between each task, and for constructing the Voronoi cell for a task. Voronoi cell is a polygon with an arbitrary shape (Fig. 3a). Due to the requirements imposed by the agricultural domain, i.e., parallel trajectories, we introduce following constraints on a cell construction:

- Each cell is either a rectangle or a square
- Width and height of a cell are equal to the length of the building block of a task multiplied by some integer value

Due to the introduced constraints, we customized existing Voronoi diagram. Our decomposition algorithm consists of following steps:

1. Order all tasks \( t_i \) and \( t_{i+1} \), \( 1 \leq i < n \), such that \( r_{i+1}(x) < r_i(x) \). This means to sort all tasks from left to right based on the \( x \) value of a central point.
2. For each task \( t_i \in T, 1 < i \leq n \), create a line between points \( r_i \) and \( r_{i-1} \).
3. Check if the line \( r_{i}r_{i-1} \) intersects with existing lines denoting borders between cells:
   a. If it intersects, then choose a task \( t_j, j < i \), such that a line \( r_{i}r_{j} \) does not intersect with existing lines denoting borders between cells and set \( r_{i-1} = r_{j} \).
   b. Otherwise retain the line.
(4) Calculate the point \( h \) such that \( \text{dist}(h, r_{t_i}) = \text{dist}(h, r_{t_j}) \), i.e., point \( h \) is in the middle of two tasks \( t_i \) and \( t_{j+1} \).

(5) Create a line denoting a border between a cell containing \( t_i \) and a cell with \( t_{j+1} \). Calculation is based on a parameter \( k \) denoting whether to prefer horizontal or vertical decomposition.
   a. If \( |\text{dist}_{xy}(r_{t_i}, r_{t_{j+1}})| < k \) or \( |\text{dist}_{xy}(r_{t_{j+1}}, r_{t_{j+1}})| > k \) (\( \text{dist}_{xy} \) is distance between points on x-axis and y-axis), then create a vertical line parallel to y-axis which passes through the point \( h \) denoting the border between two cells.
   b. Else create the horizontal line parallel to x-axis which passes through the point \( h \) denoting the border between two cells.

(6) Ensure that the new line denoting the border between cells containing \( t_i \) and \( t_{j+1} \) does not intersect with existing borders. Add the line to the list containing cell borders.

(7) Repeat step 2 if there are tasks in the list \( T \), otherwise return the list of lines denoting cells.

Cell construction in step 5 creates lines (cell borders) parallel to x or y axis and thus conforms to the constraint that each cell is either a square or a rectangle. Algorithm performed in step 6 is quite complex and is out of the scope of this work. As a result of introduced constraints, it is possible that some tasks partially lay out of their cells and span to other cell. Moreover, parameter \( k \) instructs the decomposition process to either prefer vertical or horizontal decomposition. Fig. 3b and c illustrate how the shape of cells depends on the parameter \( k \). When horizontal decomposition is preferred, \( k = 5 \), cells are more squared (Fig. 3b). On the other hand, cells are long and narrow when vertical decomposition, \( k = 9 \), is active (Fig. 3c). Fig. 3b also illustrates a task which spans over two cells (brown object in a middle).

4.2. Semantic extension for MozartSpaces

This section gives an architecture overview of the Semantic MozartSpaces. The core part of the architecture is the data model that exposes a mapping process between MozartSpaces and semantic entries. The component maps Java objects to nested blank nodes. To accomplish this, Java classes have to be respectively annotated (@RDFType, @RDFField). Fig. 4 shows an example of annotated Java class and a nested blank node. The example uses Turtle syntax to provide a suitable representation for a nested blank node. Fig. 4 illustrates the structure of the Java class TaskEnhanced in the use case. The Set property needsSkill refers to the type of the implement required on the robot (e.g., spraying).

Fig. 5 shows an example of semantic entries presenting a spraying task (left side of figure) and an agent with a spraying skill (right side of figure). Both, a task and an agent use the same ontology, illustrated in Fig. 2, for describing properties, e.g., location (lines 4–10), resources (lines 12–16), and needed skills (line 18). These semantic entries are used throughout our use case.

Fig. 6 illustrates the usage of the Semantic MozartSpaces. On the client-side the instances of Java classes (e.g. TaskEnhanced class) are created and the resource mapper component translates them into their Turtle representations.
i.e., nested blank nodes (Fig. 4), and writes them into the container. When the entry is written in the container of the MozartSpaces, it is assigned an internal ID. This ID is used to generate a URI that identifies the entry resource in the triple store. On the contrary, at the client side the nested blank node can be read from the container and translated into the Java class instance.

A SPARQL query which agents execute to retrieve suitable tasks, i.e., tasks that correspond to their skills, is presented in Fig. 7. The query uses the same ontology, illustrated in Fig. 2, as tasks and agents. Lines 6–8 define a variable with the type of TaskEnhanced and line 7 retrieves the context entry describing an executor agent. Context entries can be used as a parameter for SPARQL queries, so that more general and flexible queries are supported. In lines 10–16, query searches for tasks which require skills that overlap with those offered by the agent. Furthermore, lines 18–22 retrieve the amount of resources each task requires. Lines 24–39 ensure that tasks and the agent have same type of resources and that the agent has enough resources to execute a task. Finally, lines from 41 to 57 calculate a distance between a task and the agent, and sort the list of tasks in a way that a closest task is on the first place, i.e., the agent will first execute a task closest to his current position.

The advantage of Semantic MozartSpaces is that SPARQL queries can be used for entry selections (read in Fig. 6). For this purpose, a new semantic selector is created and it can be combined in a chain with other MozartSpaces selectors. A selector chain is a sequence of selectors where the result of one selector is piped to the next selector as an input. Within our implementation of the SPARQL query, the result is a list of entry URIs and the dataset of the query is a named graph of the MozartSpaces container. A selector can be extended to enable the use of external context entries in the query.

Out of the six RDF concepts [9], (1) a graph data model, (2) a URI-based vocabulary, (3) datatypes, (4) literals, (5) expression of simple facts, and (6) entailment, we use five concepts to describe our semantic entities. We express simple facts about robots in the form of graph model by using URIs for describing predicates, datatypes, and literals to identify values. We use RDF instead of OWL because we do not see the need for logic-based reasoning capabilities over our simple ontology. Moreover, we decided to use fast response time of the Jena triple store to the SPARQL queries, instead of rather slow reasoning.
Using OWL in a combination with an external reasoner is a part of the future work.

5. Semantic MozartSpaces in precision farming as a use case

Precise farming aims at diminishing the use of chemical inputs and improve crop quality by using a fleet of heterogeneous robots equipped with advanced sensors and actuators. This section uses previously presented concepts and describes in a greater detail the whole process of the task allocation in a precision farming scenario with a completely autonomous robotic fleet – with coordination and collaboration capability. In this respect, this scenario extends the scenario of the RHEA project [30] in which task allocation is performed centrally. While in both scenarios the mission is still created centrally, in our extended scenario the task allocation is performed in a distributed way, using Semantic MozartSpaces.

As illustrated in Fig. 8, the process starts with the central generation of tasks. The robots are aware of their local context and of capabilities of other robots in the fleet. Each robot has a specific implement and is able to execute one or more different tasks, depending on the implement type. Accordingly, within a complex mission they can autonomously select which task to perform based on the required combination of different implements (skills), such as spraying and flaming implement. Fig. 8 gives an overview of the interactions between the entities participating in the mission execution. The centrally generated mission consists of the number of tasks that require some amount of resources (1), e.g., spraying resources (blue objects on Fig. 8), flaming (orange objects on Fig. 8) or tilling resources (brown objects on Fig. 8). After the area decomposition algorithm assigns each task to a

```
@prefix moz: <xsvm://mozartspaces.org/semantic#>
@prefix ma: <http://mozartspaces.org/ma#>

SELECT ?entry
WHERE {
  ?entry moz:hasEntryPoint ?entryValue .
  [] moz:hasContextEntryPoint ?contextEntry .
  ?entryValue a ma:TaskEnhanced .
  OPTIONAL {
    select ?entryValue (count(?ts) as ?overlappingSkills) {
      ?contextEntry ma:hasSkill ?ts .
    } group by ?entryValue
  }
  FILTER (?overlappingSkills > 0)
  OPTIONAL {
    select ?entryValue (count(?res) as ?neededResources) {
    } group by ?entryValue
  }
  OPTIONAL {
    select ?entryValue (count(?resource) as ?overlappingResources) {
      ?entryValue ma:needsResource
      [ ma:isResource ?resource ;
        ma:hasAmount ?neededAmount
      ],
      ?contextEntry ma:hasResource
      [ ma:isResource ?resource ;
        ma:hasAmount ?agentAmount
      ],
      FILTER (lbound(?resource) || ?neededAmount <= agentAmount)
    } group by ?entryValue
  }
  FILTER (lbound(?neededResources) || ?overlappingResources > 0)
  ?entryValue ma:centralPoint
  [ ma:isPoint ?p ;
    ma:posX ?x ;
    ma:posY ?y
  ];
  ?contextEntry ma:hasPosition
  [ ma:isPoint ?a ;
    ma:posX ?b ;
    ma:posY ?c
  ];
  BIND ((abs(?x - ?b) + abs(?y - ?c)) as ?distance)
}
GROUP BY ?entry ?distance
ORDER BY ASC (?distance)
```

Fig. 7 – SPARQL for selecting tasks from Semantic MozartSpaces.
Voronoi cell in order to reduce spatial interference between robots, the user writes tasks to Semantic MozartSpace (2), referred to as Semantic MS. Each generated task is translated in a semantic entry and written to the underlying triple store (3). After the tasks are produced, the execution phase of mission starts. Since robots are aware of their local context, they fetch local information, i.e., type of an attached implement, amount of available resources, position, to build a query (4) for selecting a matching task from the local repository (5). Semantic MozartSpaces executes the query received from a robot (6) and returns a matching task (7). Upon receiving the task, the robot executes it (8) and writes it to the container (9) which stores it into the triple store (10). After the robots successfully execute all generated tasks, the mission is finished. While the task allocation in RHEA scenario is static and performed before the mission starts, the distributed task allocation in extended scenario is based on dynamic mapping between tasks and available robots.

6. Results and discussion

The described implementation has been tested to acquire initial efficiency measurements, i.e., to determine the expected performances and scalability of the Semantic MozartSpaces and to identify potential optimization areas. We evaluated performance of the area decomposition algorithm and spaces in five scenarios: (1) evaluation of spaces performances when first all tasks are produced and then executed, (2) comparison of three different configurations for the area decomposition algorithm, (3) scenario where a new task is created during a mission execution phase, (4) testing the subsume matching degree of task selection algorithm (introduced in Section 3), and (5) testing the plugin matching degree of task selection algorithm (also introduced in Section 3).

Performance tests of the spaces were focused on observing the system behavior, i.e., time required to execute a mission when the number of tasks and executors vary. Additionally, we tested the system scalability by adding more executors, from 3 to 14, for the same number of produced tasks. One simplification of the evaluation scenario as compared to the real world is related to the task duration which is 2500 ms (correspond to approximately 4 meters trajectory of a robot moving at 6 km/h). In this specific scenario where the tasks are first produced and then executed, specific amount of time is needed to generate desired number of tasks. Fig. 9 shows how the production time behaves when a number of produced tasks increases. The production time slowly increases, stays around 1 s, when the number of tasks is less than 100 and increases faster when the number of produced tasks is higher than 100, e.g., it is around 1 s for 5 tasks, and around 5 s for 600 tasks.

Fig. 10 shows how the number of executors and tasks influences execution time in the scenario where tasks are first produced and then executed. Execution times, when multiple executors are deployed, converge when there are more than 300 tasks. E.g., 8 executors perform 50 tasks in 18 s and 600 in 410 s where 14 perform 50 tasks in 10 s and 600 in 380 s. We detected that 14 executors don’t outperform 8 executors, especially when there are more than 300 tasks. This is due to complex scheduling mechanisms in the MozartSpaces.

In the RHEA scenario, tasks that build a mission are defined in advance and their number is less than 100. In terms of estimating performance, we can use presented results to estimate the mission execution time. Due to the fact that all tasks are produced in advance, we choose simulation results from Figs. 9 and 10 to predict the execution time. First, we estimate a time to produce 100 tasks to 1,5 s (Fig. 9) and then use Fig. 10 to select the optimal number of executors. Since there are 3 robots in RHEA, the time to complete the mission is around 90 s (corresponds to 150 m trajectory), in total 91,5 s (1,5 s for production and the rest for execution). It is worth to notice that the execution time depends on the time that executor needs to perform a task; 2500 ms. RHEA setup with three executors is used as a configuration for the other four test cases.

Area decomposition algorithm prefers either vertical or horizontal decomposition of a working area. To estimate which decomposition yields better results, we decomposed the same working area (300 m long and 200 m wide) using three different setups for the area decomposition algorithm:
(1) preferred horizontal decomposition, similar to Fig. 3b, (2) preferred vertical decomposition, Fig. 3c, and (3) balanced decomposition. Working area comprised of 15 tasks, 5 tasks required flaming skills, 5 spraying skills, and 5 tilling skills. Each task had a size of three or four building blocks in a different configuration. One building block represents 10 m × 10 m cell (600 cells in total). There were three different executors each matching the one type of requested skills. It is worth to notice that all executors had enough resources to fulfill all matching tasks.

To quantify the results of three different area decomposition configurations, we measured the size of area each executor covered during a mission execution. Three different decomposition setups yielded three coverage distributions. Table 1 compares three distributions against uniform coverage distribution which expects each robot will cover 200 cells since there are 600 cells in total. First column in Table 1 denotes the area decomposition configuration, next three columns represent three different executors and the number of covered cells, and the last one is a deviation from uniform coverage distribution. Due to the lowest deviation, vertical configuration is chosen as a setup in following test cases.

Fig. 11a illustrates a field divided into 15 cells each containing one task. The number and types of tasks and executors is same as in the previous test case. Observing Fig. 11a, it can be noticed that the flaming (orange) task marked with a black circle spans over two cells. Therefore, an executor assigned to the main cell, the one where the center of mass is, can just partially execute the task. However, since executors inspect whole cells, not just tasks, an executor in an adjacent cell, where the task spans, generated a new flaming task which has the size of a one building block. Hence, the whole mission ended up with having 16 instead of initial 15 tasks and the flaming executor had one task more to perform than peers. This use case demonstrated two important features provided by the developed framework: (1) ability to dynamically generate new tasks, and (2) supporting exact matching semantic degree between requested and advertised resources.

Last two test cases pertain to the validation of subsume and plugin matching degrees in the task selection algorithm. Both tests were run on the field configurations from Fig. 11. To test subsume matching degree, we added one more skill and accompanying resources to an existing tilling executor which is now able to handle two types of tasks, tilling and flaming.

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Spraying</th>
<th>Flaming</th>
<th>Tilling</th>
<th>Δ Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>220</td>
<td>160</td>
<td>220</td>
<td>80</td>
</tr>
<tr>
<td>Balanced</td>
<td>242</td>
<td>142</td>
<td>216</td>
<td>116</td>
</tr>
<tr>
<td>Vertical</td>
<td>195</td>
<td>176</td>
<td>229</td>
<td>58</td>
</tr>
</tbody>
</table>
Additionally, we reduced the amount of the resources for the flaming agent which is now able to execute three instead of five flaming tasks. Under the given setup, we run a mission which yielded following results. All tasks were performed and the mission was accomplished. However, task distribution was not uniform because the flaming executor only performed three out of five flaming tasks. The other two were performed by the tilling executor enhanced with a flaming skill. We noticed that the enhanced executor always performed the closest tasks, no matter whether they were tilling or flaming tasks. Our experiment confirmed the subsume property of the task selection algorithm, i.e., an executor with complex skill composed out of two or more atomic skills is also able to execute atomic tasks.

Plugin matching degree is validated in the last experiment where three complex tasks are added. Complex tasks are composed of atomic tasks, tilling and flaming in this case, and are painted in green on Fig. 11b. On the other hand, there are three executors, each with only one atomic skill, on disposal. In order to conform to plugin matching degree, the task selection algorithm should be able to deal with composed tasks where at least two robots have to collaborate. After executing a mission under the given circumstances, the system yielded with a total of 18 tasks. It means that each time an executor selected a composed task, it executed its atomic task and created a new atomic tasks. This behavior attributed to accomplishing the mission. The experiment demonstrated two important features provided by the developed framework: (1) executors collaboration on complex tasks, and (2) supporting plugin matching semantic degree between requested and advertised resources.

7. Conclusion and future work

The integration of the area decomposition algorithm and the space-based paradigm with underlying semantics provided a robust and scalable middleware for a task allocation in multi-robot systems. Two important new features are: (1) the area decomposition which ensures that each robot operates in its own cell and therefore decreases spatial interference between robots, which leaves the robots more time to focus on a domain work, and (2) Semantic extension for MozartSpaces that empowers robots to automatically infer the task-robot mapping based on the data in the dynamically updated triple store, and to dynamically select a task for immediate execution. Although the Semantic MozartSpaces framework integrated two different technologies, MozartSpaces and semantics, it retained complete set of functionalities from both, i.e., coordination mechanisms and transactions support from MozartSpaces and query and reasoning capabilities endowed by semantics technologies. Furthermore, we have demonstrated how the framework behaves in the RHEA scenario where a set of robots execute specific tasks and how the execution time depends on an insertion of new resources.

The benefit of using semantics for a task allocation is two-fold: (1) developed ontology provides uniform description of heterogeneous and distributed resources, and (2) semantically annotating tasks and services yields more accurate matching and thus results in more efficient utilization of resources. Former enables all heterogeneous robots to execute tasks produced by the third party user because all entities conform to the introduced ontology. Moreover, heterogeneous robots can even generate ad-hoc tasks which are executable by other robots. Later showed that the matching algorithm satisfies exact, subsume, and plugin matching degrees which are well established guidelines in reviewed literature.

Future work will be steered in the direction of modelling robot behavior within a cell and during a transition from the one to the other cell. Focus will be on developing coordination patterns for robots while they are roaming from one cell to the other because it is the most critical part of a mission. Additionally, we will study how OWL reasoning together with a geographical distribution of a central task repository influences mission duration, communication and coordination patterns between robots.

Acknowledgments

This paper is partially supported within the European Research Project RHEA. The Telecommunications Research Center Vienna (FTW) is supported by the Austrian government and the City of Vienna within the competence center program COMET.
REFERENCES


