MULTI-ROBOT TASK ALLOCATION THROUGH SHARE-RESTRICTED RESOURCES

Pedro Mitsuo Shiroma*, Mario F. M. Campos*

*VeRLab: Laboratório de Visão Computacional e Robótica
Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos n. 6627, Prédio do ICEx – Belo Horizonte, MG, Brasil

Emails: pshiroma@dcc.ufmg.br, mario@dcc.ufmg.br

Abstract— Multi-robot systems can be found nowadays executing several tasks such as environment cleaning, area surveillance, searching for survivors, and planetary exploration. Among the several problems that arises in such scenarios, multi-robot task allocation (MRTA) is an important problem that has captured the interest of researchers for many years. In this work we propose a new methodology to handle the MRTA problem, based on share-restricted resources. Informally, a share-restricted resource is any resource in a robot or in the environment that cannot be freely shared among the robots. A coalition of actions is allocated to each task, and each action defines a set of constraint functions applied to the share-restricted resources, which is used to determine if a robot can execute a new task or not in parallel with existing tasks. These concepts will grant a robot the ability to execute multiple tasks simultaneously. Preliminary results in scenarios like obstacle avoidance, multi-robot routing and object transportation demonstrates the soundness of our solution.

Keywords— Multi-robot task allocation, robot coordination, mobile robots.

Resumo— Sistemas multi-robóticos podem ser encontrados nos mais diversos cenários como limpeza de ambientes, patrulhamento de áreas, busca por sobreviventes, exploração planetária, dentre outros. Dentre os diversos problemas que surgem nestes cenários, a alocação de tarefas para múltiplos robôs é um importante problema que tem capturado o interesse dos pesquisadores por muitos anos. Este trabalho propõe uma nova abordagem baseada no conceito de recursos com compartilhamento limitado. Informalmente, um recurso com compartilhamento limitado é qualquer recurso pertencente a um robô ou ao ambiente que não pode ser livremente compartilhado entre os robôs. Uma coalisão de ações é alocada para cada tarefa e, cada ação define uma função de restrição sobre o conjunto de recursos com compartilhamento limitado, o qual é usado para determinar se a nova tarefa pode ser executada em paralelo com as tarefas atuais ou não. Estes conceitos irão permitir a um robô executar múltiplas tarefas simultaneamente. Resultados preliminares em cenários como desvio de obstáculos, roteamento e transporte de objetos demonstram a viabilidade da solução proposta.

Palavras-chave— Alocação de tarefas para múltiplos robôs, coordenação entre robôs, robôs móveis.

1 INTRODUCTION

Multi-robot systems is a challenging field that has been receiving increasing attention in robotics over the past years. Among the advantages that such systems presents over single robots, we can highlight the increased robustness, efficiency, and higher range of tasks that can be accomplished. Cooperation has naturally arisen as one of the main approaches to fully utilize the potential of each robot in the group. Several issues related to coordination and control have been considered in the recent literature, which important results being obtained. As the number and complexity of robots increases, distributing tasks among them has become increasingly important, and is known nowadays as the multi-robot task allocation problem. If more than one team is capable to carry out a task, then an automatic decision process should be used to select the best team. In a more realistic scenario, new robots can be dynamically inserted while other robots can fail. Also, their capabilities are not know *a priori*, and therefore the solver must be able to handle the dynamical insertion and removal of robots when forming the teams.

Since modern robots are usually equipped with several sensors and actuators, they potentially have the resources to execute multiple tasks simultaneously, and an efficient task allocation mechanism should be able to contemplate these features. Gerkey and Matarić (2004) proposed a taxonomy to characterize the MRTA architectures based on the relationship between robots, tasks and assignment. According to them, MRTA architectures can be classified as: single-task robots (ST) – robots able to execute at most one task at a time, and **multi-task robots (MT)** – robots capable of executing more than one task concurrently; single-robot tasks (SR) and **multi-robot tasks (MR)**; and **instantaneous assignment (IA)** or time-extended assignment (TA). This work can be classified as a MT-MR-IA approach.

ALLIANCE (Parker, 1998) uses a distributed behavior-based architecture where tasks are performed by selecting a behavior set. Impatience and acquiescence attributes are used to trigger the process of taking over tasks from other robots or giving up one's own current task. BLE (Werger and Mataric, 2000) uses the component and connector software architecture, with behaviours being modelled as components and task allocation being the connectors.

In Multi-Agent Systems (MAS) a common

concept present amongst most works is *coalition*, a temporary organization of agents that are brought together in order to solve a specific task. Theoretical results were conducted by Shehory and Kraus (1998) to study coalition formation of software agents, and were the inspiration of several successful works, like ASyMTRe (Parker and Tang, 2006), where robot capabilities are modelled as schemas, and a task is defined as a set of motor schemas, for which solutions are automatically generated by forming a coalition of schemas; and RACHNA (Vig and Adams, 2006), which defines a vector of robot capabilities that is matched against a vector of task requirements. According to Vig and Adams (2006), a great number of works developed in MAS cannot be directly transferred to multi-robot scenarios, because they do not consider restrictions that arise with real robots, like losses communication, device failure, situated agents, dynamical environments, and non-transferability of resources.

Traderbots (Dias, 2004) is a distributed architecture which can form local centralized coalitions based on market theory. It assumes that, if each robots tries to maximize its own profit, then global cost will also be maximized. Coordination process is modelled using both cooperative and competitive robots. Also, tasks are accomplished even in a scenario with no communication; although communication will improve efficiency in the generated solutions. M+ (Botelho and Alami, 1999) is a decentralized protocol aimed in reallocation and replanning, which is divided into three layers: A task allocator, based on the Contract Net Protocol (Smith, 1980); a fault-tolerance module; and a task executor module, responsible for the coordination. The CNP was also the starting point of several successful works, like MURDOCK (Gerkey and Matarić, 2002), which uses a greedy algorithm and a time-limited contract to provide fault-tolerance. In Chaimowicz et al. (2002) the task allocation problem is modelled as a hybrid automaton. Task assignment is treated as discrete events and the controllers are represented as continuous states. Therefore, it can potentially share the benefits of formal analytical machinery developed for hybrid automata.

Our approach is based on the CNP to form coalitions of actions and is similar to the ASyMTRe approach in many aspects. However, unlike (Parker and Tang, 2006), where the output of motor schemas are summed to generate the overall behavior, our approach tries to find out a motor output that will always be admissible for all allocated tasks. It distinguishes from other related work because it enables the design of multiple concurrent coalition solutions, each one assigned to a task, which can include actions belonging to overlapping robots. This means that a robot can be assisting the execution of multiple tasks concurrently through the actions, and therefore be classified as a multi-task (MT) robot.

The multi-robot task allocation problem can be defined as: Given $\mathbf{R} = \{r_1, r_2, ..., r_m\}$, a set of m heterogeneous mobile robots, $\mathbf{T} = \{t^1, t^2, ..., t^p\}$, a set of p tasks to be executed, which can be randomly inserted. Let $\tau_i \subset \mathbf{R}$ be a team of robots, and $\mathfrak{R} = 2^{\mathbf{R}}$ be the set of all teams of robots that can be formed. The problem addressed in this work can be stated as to find a function $\mathcal{A} : \mathbf{T} \mapsto \mathfrak{R}$ such that $\mathcal{A}(t^k)$ is a team of robots capable of performing the task t^k.

2 METHODOLOGY

Given a team of heterogeneous mobile robots, our objetive is to assign tasks like `box pushing` or `transportation` to robots as they are inserted by an external agent, like a human operator.

2.1 Action

We model robot capabilities or skills, such as *read laser*, *detect obstacles*, *avoid obstacles*, *push box*, as a set of *actions* which are, to some extent, similar to the *schema* concept (Arkin, 1987).

Definition 1 An action is any computational module that can either produce data, consume data, or accomplish a task.

Let $a_{i,j}$ be the j-th action in robot r_i. Also, let n_i be the total number of running actions in robot r_i. We adopt, similar to Parker and Tang (2006), a set of information types $\mathbf{F} = \{f_1, f_2, ..., f_p\}$ and restrict all actions inputs and outputs to be a subset of $\mathbf{F}$. Notice that an action can produce data by directy reading from a sensor or transform one information into another, like a range-data to a list of obstacles.

By defining robot capabilities as a set of independent actions or schemas (like in (Parker and Tang, 2006; Vig and Adams, 2006)) it is possible to transparently handle failures. When a sensor fails, only the capabilities (actions) that depend on that sensor output are affected. Thus, if another robot is able to provide the same information of the fault module, then consumer actions will still be able to execute after a reconfiguration.

However, schema-based approaches only checks for the inputs to determine if they can be activated. An important aspect in our work is that, even if all actions' inputs are available, this does not necessary means that it is allowed to run and produce its output data to be sent to another action. The constraint functions defined later is the other pre-requisite to activate an action.

Resources like communication link, processor, battery power and the robot pose have physical restrictions that limit the amount of actions that can be concurrently running. For example, a communication link cannot exceed the device's maximum bandwidth, and hence, higher demands would not be acceptable. Therefore, our model should refuse new connections while the link is not capable to adequately handle new requests. We capture this concept by the following definition:

Definition 2 A share-restricted resource is any property in the environment that cannot be freely shared among the actions.

Examples of share-restricted resources are robot position, robot energy, communication bandwidth, and free configuration space. A share-restricted resource can either belong to a robot (e.g. energy, position) or be intrinsic to the environment (e.g. the free configuration space). Let ${}^{i}\chi = \{{}^{1,i}\chi, {}^{2,i}\chi, \ldots, {}^{s_i,i}\chi\}$ be the set of share-restricted resources in the environment ($i = 0$) or in robot r_i ($i > 0$). For each share-restricted resource ${}^{k,i}\chi$, we associate a codomain ${}^{k,i}C$ which "measures" the availability of a shared-resource ${}^{k,i}\chi$. Each action $a_{i,j}$ defines a constraint function ${}^{k,l}\varphi_{i,j}(t) : \Re \mapsto {}^{k,l}C$ which measures the amount of the share-restricted resource ${}^{k,l}\chi$ in robot r_l (or in the environment, if $l = 0$) required by $a_{i,j}$ at time t. The space ${}^{k,l}C$ is defined such that it accepts two operators, a compound operator:

$$\oplus : {}^{k,l}C \times {}^{k,l}C \mapsto {}^{k,l}C, \tag{1}$$

which is used to "sum" the constraints imposed by two constraint functions, and a comparison operator:

$$\prec : {}^{k,l}C \times {}^{k,l}C \mapsto \{true, false\}, \tag{2}$$

used to check if the sum of constraint functions exceed the maximum capacity, ${}^{k,l}\varphi_{max}$, of the share-restricted resource. Therefore, if we have two actions, $a_{i,1}$ and $a_{i,2}$, and a share-restricted resource ${}^{k,l}\chi$, the sum of the two constraint functions can be denoted as ${}^{k,l}\varphi_{i,1} \oplus {}^{k,l}\varphi_{i,2}$. Additionally, if an action $a_{i,j}$ is not running, we define ${}^{k,l}\varphi_{i,j} \oplus {}^{k,l}\varphi_{r,s} = {}^{k,l}\varphi_{r,s}$. Let's define: $\sum_{j=1}^{n_i} {}^{k,l}\varphi_{i,j} \triangleq {}^{k,l}\varphi_{i,1} \oplus {}^{k,l}\varphi_{i,2} \oplus \ldots \oplus {}^{k,l}\varphi_{i,n_i}$, as the constraint imposed by all running actions in robot r_i over share-restricted resource ${}^{k,l}\chi$. Similarly, define: $\sum_{i=1}^{m} {}^{k,l}\varphi_{i,j} \triangleq {}^{k,l}\varphi_{1,j} \oplus {}^{k,l}\varphi_{2,j} \oplus \ldots \oplus {}^{k,l}\varphi_{m,j}$, as the composition of the constraint functions imposed by all robots over share-restricted resource ${}^{k,l}\chi$. Therefore,

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} {}^{k,l}\varphi_{i,j} \prec {}^{k,l}\varphi_{max} \tag{3}$$

can be interpreted as "Does the sum of the constraints imposed by all active actions exceed the maximum capacity of share-restricted resource ${}^{k,l}\chi$?" Next we define the codomain, φ_{max}, and operators $\oplus$ and $\prec$ for the most common share-restricted resources in robotics.

2.1.1 Communication link

Communication, and specially wireless communication, has been fundamental for the operation of mobile robots. Packet loss, data corruption and network disconnection are usual events in real scenarios. Bandwidth is the main limitation imposed by communication links, so we define ${}^{k,l}\varphi_{max}$ as the maximum bandwidth. Thus, the codomain for the communication bandwidth $C \triangleq \Re$, and the constraint function $\varphi_{i,j}$ is the required bandwidth by action $a_{i,j}$ to properly execute its operations. In this case, the $\oplus : \Re \times \Re \mapsto \Re$ operator is defined simply as the algebraic sum, and $\prec : \Re \times \Re \mapsto \{true, false\}$ operator is the "less or equal" operator.

For example, if an action $a_{i,1}$ requires data size of 100 Kbits at a rate of 100 samples/sec (totalizing 10Mb/s), action $a_{i,2}$ requires 40 Mb/s, and action $a_{i,3}$ demands 30Mb/s, and the maximum bandwidth, φ_{max}, is $100Mb/s$, then actions $a_{i,1}$, $a_{i,2}$ and $a_{i,3}$ can share the communication resource since $10Mb/s + 40Mb/s + 30Mb/s \leq 100Mb/s$.

2.1.2 Processor

The dynamical aspect of the real world coupled with the complexity of data analysis of some data sources, like cameras, turn the robot into a voracious processor consumer. However, in order to increase autonomy, a robot should rely only in low power components, which consequently restricted processing power. Therefore an efficient use of the processor is essential for any architecture that will handle task allocation in a complex scenario. It is specially critical in actions that impose real-time constraints like low-level controllers.

Thus, we define φ_{max} as the maximum processing power allocated to the MRTA (disconting the time allocated to other processes, e.g. operating system). The constraint function $\varphi_{i,j}$ for an action $a_{i,j}$ is the required processing power for that action. The operator $\oplus : \Re \times \Re \mapsto \Re$ for the processor is defined as the algebraic sum, and the operator $\prec : \Re \times \Re \mapsto \{true, false\}$ is defined as the "less or equal" operator.

2.1.3 Position

One of the most important share-restricted resources that we must deal with in robotics is the robot position. The position of a robot is intimately related to the motor and actuator resource and, for a given action, the commands sent to the actuators may cause one of three effects: (i) make the robot advance toward its goal; (ii) be irrelevant to the execution of that action; or (ii) be harmful and provide a negative impact to the completion of the action.

The codomain for the position resource is defined as the special Euclidean space ($C \triangleq \mathcal{SE}(3)$). The constraints of all actions can then be joined into the configuration space (Fig. 1) and, if their intersections is not the empty set, it means that there exist a set of motor commands that satisfies both tasks. Therefore, we have that $\oplus \triangleq \cap$, $\prec \triangleq \neq$ and $\varphi_{max} \triangleq \emptyset$. Remember that the constraint function is a timed-function, so the region can change over the parameter t.

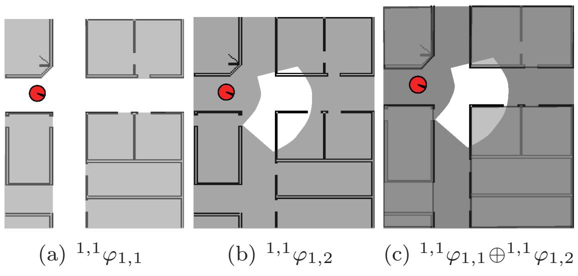
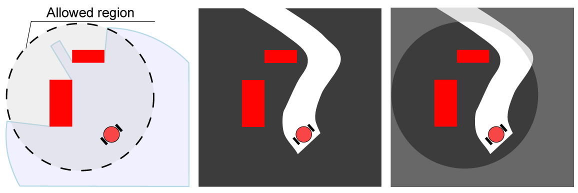

(a) ${}^{1,1}\varphi_{1,1}$ (b) ${}^{1,1}\varphi_{1,2}$ (c) ${}^{1,1}\varphi_{1,1} \oplus {}^{1,1}\varphi_{1,2}$

Figure 1: In light gray the constraints imposed by a survey action. In dark gray the constraints of a controller. The intersection of the areas (in white) indicates the admissible velocities for both actions.

For example, in order to check if actions $a_{i,1}$, $a_{i,2}$ and $a_{i,3}$ can be executed concurrently, we test if $\varphi_{i,1} \cap \varphi_{i,2} \cap \varphi_{i,3} \neq \emptyset$ is true.

Note that not only actions that produce velocity commands can impose constraints over the position of a robot, but actions that read sensors can also restrict the allowed configuration space.

Suppose, now, that an action is querying the obstacles in a given region (Fig. 2). In order to properly answer the query, and consequently keep the established contract, the robot must stay at a maximum distance from the region (assuming an omnidirectional sensor). Therefore, constraint functions allow us to evaluate if a set of actions can be concurrently executed by a robot.



(a) Omnidirectional sensor. (b) Path following action. (c) Sum of both constraints.

Figure 2: A sensor can also constrain the allowed position of a robot. When the robot reaches the upper boundary, it must decide which action it will continue to execute and which one it will stall.

The coalition formation protocol is presented in (Shiroma and Campos, 2009).

3 EXPERIMENTAL RESULTS

3.1 Simulation setup

The experiments were conducted using the player/stage/gazebo framework (Collett et al., 2005). To validate our architecture we designed a set of three tasks executed by two robots, namely, transportation, obstacle avoidance, and multi-robot routing. Each robot is controlled by an independent client, which is responsible for listening for new tasks, bid for new auctions, query for new data, and execute the assigned tasks. All communication among clients (i.e. robots) is implemented using the TCP protocol. The available actions to the robots are shown in Table 1.

Action	input	output	goal
route	robot-pose	–	routing
transport	robot-pose	–	transportation
avoid obstacle	obstacles	–	obst. avoidance
detect obstacle	range-data	obstacles	–
read laser	–	range-data	–
read localize	–	robot-pose	–

Table 1: List of available actions.

3.2 Task validation

Although obstacle avoidance is normally defined as a sub-part of a task's solution, we choose to consider it as another task that must be accomplished concurrently with the remaining tasks. An *avoid obstacle* action, which is responsible to constraint the pose share-restricted resource, was created to execute this task. The *avoid obstacle* action requires a list of *obstacles* as input, that can be produced by a *detect obstacle* action, which, in turn, requires *range-data* produced by a *read laser* action. Fig. 3 shows the coalition formed to perform the obstacle avoidance task.

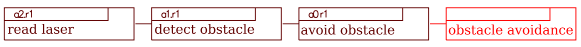


Figure 3: Coalition formed to execute the obstacle avoidance task.

In the context of this work, a transportation task is defined as *waypoints* (producer and consumer) that a robot must continually and sequentially visit. Poses that drives the robot further from current distance to the waypoint are marked as not allowed, and the height of the remaining poses are computed based on its distance to the waypoint. The constrained C-space can be represented by an image, with black pixels corresponding to poses not allowed by an action, while gray pixels corresponding to poses allowed by that action. The brighter the pixel the more attractable is the corresponding pose. The pose with maximum height is choose to estimate the velocity that will be taken.

In this experiment we set the producer to be at $(-5, 0)$ and the consumer at $(5, 0)$ (Fig. 6(a)). Four obstacles were spread in the environment, with two of them in the path that directly connects producer and consumer. In this experiment, there are two actions constraining the allowed C-space. The constrained C-space for the *avoid obstacle* action (fig. 4(e) to 4(h)) and for the *transport* action (fig. 4(i) to 4(l)) are composed using Equation 3 (fig. 4(m) to 4(p)).

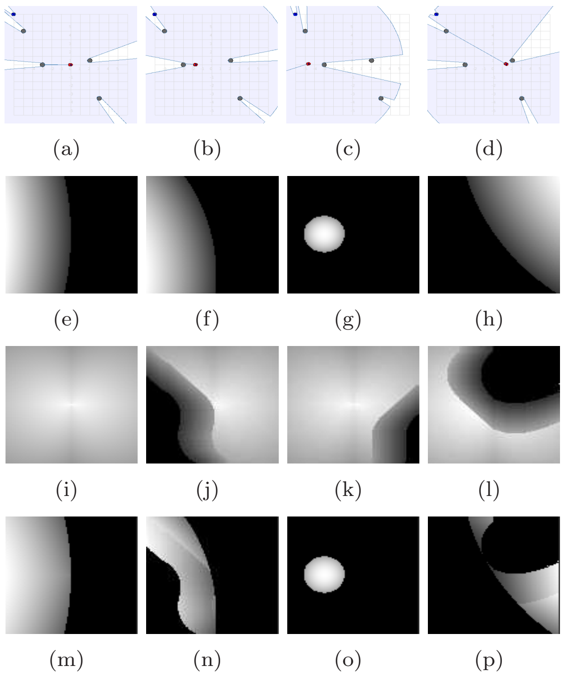


Figure 4: The constrained C-space for the *transport* and *avoid obstacle* actions, and their composition.

Fig. 5 shows the progress of the availability in the pose share-restricted resource in the previous experiment. The two depressions for the transport action corresponds to the robot approximating the producer and the consumer waypoints. The three valleys for the avoid obstacle corresponds to the two obstacles.

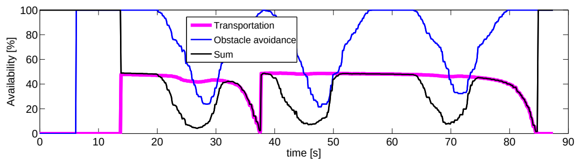
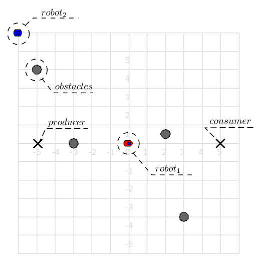


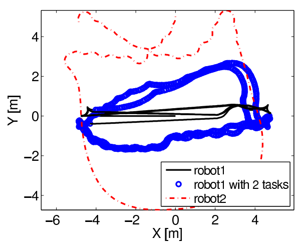
Figure 5: The availability of the pose share-restricted resource over time.

To illustrate our system's capacity of handling multiple tasks simultaneously, a second task is inserted after awhile. The second task consists of visiting all waypoints defined by an 13m × 13m grid around the origin spaced by 2 meters. In order to accomplish this task, the robot must pass, at least once, closer to all points within the region. The *route* action used to accomplished this task uses a greedy algorithm, which tries to visit the nearest waypoint at each instant. It is defined such that it allows the robot to be at any position in the environment, and thus its constraint function does not affect in the available positions. However, it does affect the determination of the velocity that will be used. The height of each pose is equal to $k(1 - d/d_0)$, where d_0 is the distance from the current origin to the nearest waypoint.

Three scenarios were elaborated to test this setup. In the first scenario, the robot is not allowed to execute multiple tasks simultaneously, and thus we use these results as the baseline for the remaining experiments. The second scenario consists in one robot trying to execute both tasks simultaneously, which illustrates the main contribution of this work. The robot starts to move off the line connecting the waypoints defined by the transportation task in order to accomplish both tasks, until it reaches an equilibrium and the robot is no longer capable of executing the routing task adequately, and eventually it goes on to watch previously visited areas. At last, in the third scenario, a second robot is inserted in the environment and it takes control of one of the tasks when it is detected that it is stalled. The transportation task was defined such that the robot must move from the producer waypoint to the consumer waypoint five times. The trajectories performed by the robots in the third experiment can be seen in Fig. 6(b).



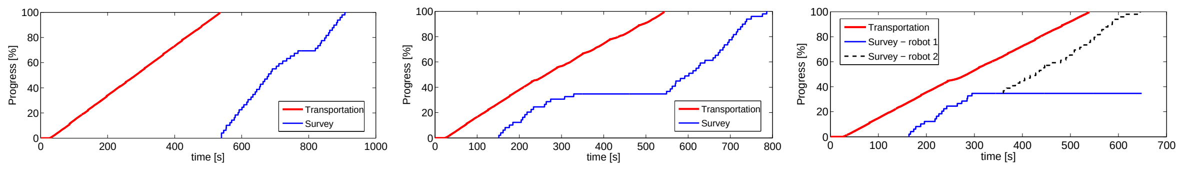
(a) Experimental setup.



(b) Trajectory of two robots executing two tasks.

Figure 6: Experimental setup: two robots in an environment with obstacles.

Fig. 7(a) shows the progress of the two tasks being accomplished separately, which is how we would expect previous works would accomplished them. In Fig. 7(b) we can see that, although the completion time for the first task has increased (from 535 to 545 sec), the overall completion time decreased from 908 to 786 sec. The insertion of a second robot (at time 160) decreased the completion time even more, to 650 sec.



(a) A single robot executing two tasks sequentially. (b) A single robot executing two tasks concurrently. (c) Two robots executing two tasks.

Figure 7: The progress of the tasks.

4 CONCLUSION

Although similar to schema-based approaches (Parker and Tang, 2006; Vig and Adams, 2006), in our approach, the sum of motor outputs is replaced by the position share-restricted resource. The introduction of share-restricted resources and constraint function concepts enabled us to clearly define when two tasks can be concurrently executed or when a robot can form a new coalition. It also allows us to design a system robust to failures and which presents better performance when compared to similar approaches in the literature. Future works will present more complex scenarios with long-term experiments and more challenge tasks that will explore the potential presented in the proposed methodology.

Acknowledgments

The authors would like to thank CAPES, CNPq, and FAPEMIG for their support.

References

Arkin, R. C. (1987). Motor schema based navigation for a mobile robot: An approach to programming by behaviour, *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Vol. 4, pp. 264 – 271.

Botelho, S. S. C. and Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement, *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Vol. 2, Detroit - Michigan, pp. 1234–1239.

Chaimowicz, L., Campos, M. F. M. and Kumar, V. (2002). Dynamic role assignment for cooperative robots, *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Washington - DC, pp. 292–298.

Collett, T. H., MacDonald, B. A. and Gerkey, B. P. (2005). Player 2.0: Toward a practical robot programming framework, *Australasian Conference on Robotics and Automation*, Sydney, Australia.

Dias, M. B. (2004). *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*, PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Gerkey, B. P. and Matarić, M. J. (2002). Sold!: Auction methods for multi-robot coordination, *IEEE Trans. on Robotics and Automation* **18**(5): 758–768.

Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems, *The Intl. Journal of Robotics Research* **23**(9): 939–954.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation, *IEEE Trans. on Robotics and Automation* **14**(2): 220–240.

Parker, L. E. and Tang, F. (2006). Building multirobot coalitions through automated task solution synthesis, *Proc. of the IEEE* **94**(7): 1289–1305.

Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation, *Artificial Intelligence* **101**(1-2): 165–200.

Shiroma, P. M. and Campos, M. F. M. (2009). CoMutaR: A framework for multi-robot coordination and task allocation, *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*. accepted.

Smith, R. G. (1980). The contract net protocol: high-level communication and control in a distributed problem solver, *IEEE Transactions on Computers* **C-29**(12): 1104–1113.

Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation, *IEEE Trans. on Robotics* **22**(4): 637–649.

Werger, B. B. and Mataric, M. J. (2000). Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams, *Proceedings, 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pp. 347–356.