Self-organized resource allocation for reconfigurable robot ensembles

Julian Hanke, Oliver Kosak, Alexander Schiendorfer, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Hanke, Julian, Oliver Kosak, Alexander Schiendorfer, and Wolfgang Reif. 2018. "Self-organized resource allocation for reconfigurable robot ensembles." In *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 3-7 Sept. 2018, Trento, Italy*, edited by Antonio Bucchiarone and Alberto Montresor, 110–19. Piscataway, NJ: IEEE. https://doi.org/10.1109/saso.2018.00022.

Self-organized Resource Allocation for Reconfigurable Robot Ensembles

Julian Hanke, Oliver Kosak, Alexander Schiendorfer, and Wolfgang Reif
Institute for Software & Systems Engineering, Augsburg University, Germany
E-Mail: {hanke, kosak, schiendorfer, reif}@isse.de

***Abstract*—Mobile robot systems usually are designed, built, and programmed for dedicated use cases. Consequently, especially for unmanned aerial vehicles diverse applications result in very heterogeneously designed robots. To overcome this need for specialization, we propose to dynamically adapt the robots' capabilities at run-time. This is done by connecting and disconnecting hardware modules providing those capabilities, i.e., re-allocating resources within the robot ensemble. Thereby, no longer individualized robots have to be designed for different tasks. Instead, the system is enabled to adapt its hardware configuration to changing requirements. For calculating necessary adaptations, i.e., solving the resource allocation problem, we propose a heuristic, market-based approach that exploits the possibility to decompose the resource allocation problem and distributively finds a solution. We show that our approach outperforms a centralized one especially when increasing the problem size in terms of agents, tasks, and relevant capabilities while providing the same quality.**

***Index Terms*—multi-agent, multi-robot, resource allocation, self-awareness, self-configuration**

I. Introduction

Mobile robots and according multi-robot applications have finally made their way from research to industry and public. Current developments indicate their usefulness in versatile forms of applications which offer the potential to improve human life in a multitude of different areas. From applications simply dedicated to public amusement during mass events (e.g., at the Olympic Winter Games 2018 in Pyeongchang, or CES 2018 in Las Vegas) over intensified use for supporting different research areas (e.g., environmental research [1]–[4]) up to mission critical tasks like supporting rescue forces in dangerous environments [5]–[8]. Especially systems including unmanned aerial vehicles (UAVs) experience an upswing. While offering a huge set of heterogeneous robot capabilities for this wide range of use cases, each application for itself relies on specifically constructed robots tailored to their dedicated tasks. Robots used for meteorologic research, e.g., rely on accurate sensing capabilities with specialized sensor combinations [1], whereas search and rescue robots require accurate visual imaging [9] and manipulating capabilities of different kinds [10]. Due to that need for specialization, every time an existing application is modified or a new use case is found where mobile robots may help with, new robots offering the particular capabilities have to be designed, constructed, programmed, and deployed. The approach proposed in our previous work [8], [11] aims at overcoming this limitation. By *separating capabilities from robots*, we introduce a new degree of flexibility for adapting to changing task requirements or even application requirements. We already support this approach with a modular reference software architecture and according algorithms that are able to exploit the resulting freedom in capability re-allocation [8]. Our study in this paper focuses on enabling the key feature for this approach, i.e., a mechanism for re-allocating capabilities to robots. Therefore, we propose a solution to solve the problem of resource allocation, where *resources* are hardware modules that deliver certain capabilities and *resource holders* are robots that thereby are equipped with these capabilities, making the robots applicable in versatile tasks and applications. We develop a non-specialized approach that can be deployed in various case studies. To evaluate our work we embed it in the case study of chemical accidents, where an ensemble of robots is used to autonomously support fire fighters in handling such a situation. Unfortunately, such accidents are not rare as current incidents at Arkema Texas (2017) or at BASF Germany (2016) show. Such scenarios are characterized by a high variety in required capabilities needed to execute different tasks of relevance in a certain order that can be classified as *ScORe* missions [12]. Fig. 1 depicts how an ensemble of robots executes such a ScORe mission after an accident. First, the robots employ a universal gas sensor (ALL) to *Search* (*S*) for the initially unknown relevant parameter, i.e., the gas with the highest risk potential. Then, the ensemble has to *continuously Observe (cO)* how the identified gas (e.g., gas X in Fig. 1) disseminates in order to assess its potential harm. To track X

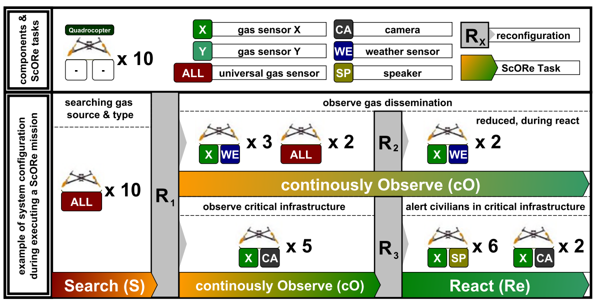

Fig. 1. Simplified example of how an ensemble consisting of reconfigurable robots can execute a ScORe mission, here, a gas accident. The legend (top part) shows available components (robots, sensors, and actuators). The bottom part shows changes in the system's configuration (i.e., component compositions) in the course of the mission. R_1, R_2, R_3 indicate situations where a resource re-allocation may happen within the ensemble.

a capability for high precision measuring and a capability to measure weather data (WE) for dissemination predictions is needed. Besides, the ensemble should still be on the lookout for other potentially harmful gases which again requires the universal gas sensor capabilities. Additionally, the robots need to observe the concentration of X along critical infrastructure and identify endangered people with cameras (CA). Finally, we expect appropriate *Reactions (Re)*, e.g., the ensemble supports the evacuation of threatened inhabited areas. In Re, one task still needs the capabilities from *cO* and another task requires the capability to inform endangered persons with loudspeakers (SP). For an ensemble of size 10, a possible task assignment is shown in Fig. 1, where resource re-allocations happen at R_1, R_2, and R_3. A conventional, non-reconfigurable system (with fixed hardware configurations) would lead to a much larger ensemble to achieve the same task coverage $(10 \cdot |\{\mathsf{ALL}\}| + 3 \cdot |\{\mathsf{X}, \mathsf{WE}\}| + 5 \cdot |\{\mathsf{X}, \mathsf{CA}\}| + 6 \cdot |\{\mathsf{X}, \mathsf{SP}\}| = 24$ devices instead of 10). To identify the hardware needed for providing certain capabilities, agents are able to query a *common knowledge data base* defining the relation between hardware types and capabilities. Given, e.g., the hardware modules X and WE, the agent gains the capability of *measuring gas X*, *estimating wind* (from WE), and the combined capability *estimate gas cloud distribution* (that uses both X and WE to predict the future gas dissemination). In this study, we neglect the quality of hardware modules (e.g., measuring frequency of a sensor) and only focus on its type. Nevertheless, in all following definitions and algorithms hardware module quality can be easily considered. We also assume hardware modules that can be plugged in each other infinitely (e.g., as daisy chain [13]). To achieve the ability to actually reconfigure our robots online, i.e., to enable online adaptivity, modularity, and other necessary self-organization abilities, we designed a single robot of our ensemble as a *Jadex Active Components Platform* [14]. By representing capabilities as such active components, we are able to design them as modular and dynamically loadable software artifacts, each encapsulating all knowledge necessary for their execution. At the same time, this design provides the possibility to dynamically equip robots with new capabilities at run-time and makes our robots self-aware concerning their current capabilities. To provide a solution to solve the problem of resource allocation, this paper is organized as follows: In Section II, we describe the problem to be solved. Subsequently, we propose our algorithmic approach to solve the problem in Section III. We then present our results in Section IV, investigate related work in Section V and wrap up our work in Section VI, giving also some suggestions on further research.

II. Problem Definition

Regarding the application environment motivated in Section I, the paramount problem that needs to be solved thus is an instance of the task allocation problem [15]. Usually, tasks in our multi-robot system setting have to be solved by multiple agents. In the scope of this paper, we consider ScORe missions where robots can be assigned to at most one task at a time. By using appropriate decomposition techniques (e.g., hierarchical task networks [16]), complex tasks that require multiple agents to be solved can be decomposed into multiple primitive tasks, which can then be handled by a single agent. With regard to the taxonomy introduced by [17] for classifying *multi-robot task allocation problems (MRTA)*, can be mapped to a *multi-robot single-task* allocation problem where tasks are instantly assigned (*ST-SR-IA*). For being able to solve the aforementioned complex tasks, each of the decomposed primitive tasks has to be allocated to an agent capable of handling it, i.e., an agent providing all capabilities that are required for solving the primitive task. The group of these agents forms the coalition that in a coordinated manner is then able to solve the complex task, as supposed by [15]. Obviously, this mechanism relies on having enough agents available providing the required capabilities. As a consequence, the task allocation fails if such a coalition can not be found resulting in having the task unsolvable at least for the current state of the system. Instead of adding more appropriately configured agents to the system (a typical solution how recent other approaches try to handle such situations, cf. [18]–[20]), our approach allows for reconfiguring the agents that are already participating in the system in terms of their provided capabilities. Thereby, the task allocation problem gets augmented by a *multi-agent resource allocation problem (MARA)* [21]. As it can be seen in Fig. 2, if no coalition can be found for a complex task, the system is enabled to reallocate resources (i.e., hardware modules) to new resource holders (i.e., the agents). By their very nature, resources in our MARA are discrete, indivisible, not sharable, static, and multi-unit (classified according to [21]). By considering the decomposed tasks' required capabilities as requirement for the resource allocation problem, a consistent system configuration has to be found to successfully finish a re-started task allocation and subsequently execute the complex task. Separating task and resource allocation in two problems reduces complexity. It eases finding task allocations when the system is already configured appropriately (this is the common case). Further this approach reduces the problem size in critical situations where time spend on reconfiguration is needed to be minimal. With this combined approach, a valid task assignment can be found in every case where the user defined task for the ensemble is feasible in any configuration of the system, i.e., there is at least any coalition with any agent configuration that is able to solve the task. If the reconfiguration fails, e.g., when hardware modules break down, an error handling can be initiated (not in the scope of this paper). We previously stated, that as side effect, this approach further facilitates the act of

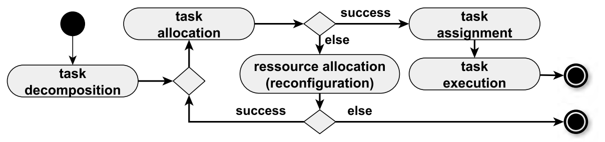

Fig. 2. Abstract definition of our process for solving tasks in the multi-robot ensemble, considering the separation of capabilities from robots. If decomposed tasks can not be allocated, a new resource allocation is calculated and the task allocation is restarted.

planing for the robot ensemble as planning no longer has to take into account heterogeneous agents but is able to plan for homogeneous agents [11]. Additionally, the overall need for hardware (robots and sensors/actuators) is reduced if actually implemented in real-world, which we aim to.

A. The Resource Allocation Problem

The resulting *resource allocation problem (RAP)* is formulated as a *constraint satisfaction and optimization problem (CSOP)* [22, Ch. 10] that minimizes the required amount of reconfigurations r_a concerning hardware modules, aggregated over all participating agents $a \in A$. This minimization problem is subject to multiple constraints regarding participating agents as well as relevant tasks t, that define the requirements of the resource allocation problem. More precisely, these requirements are defined by the set of capabilities C_t a task $t \in T$ requires from an agent to make it capable of handling the task. The solution of the CSOP then shall guarantee that for each task there is at least one agent that is capable of handling it as the agent provides all capabilities C_a that are required by the task ($C_t \subseteq C_a$). Because this allocation is still subject to the ST-SR, we define it by an injective function $f : T \rightarrowtail A$ (*task-covering-contraint*, cf. (3)) that maps a unique agent $a \in A$ to every task $t \in T$, ensuring that $C_t \subseteq C_{f(t)}$ is true for at least one agent. While f can also be used by the task allocation afterward (if enriched with other relevant constraints, e.g., for allocating tasks to the fastest or the nearest agent), we want to abstract from defining a concrete function during the resource allocation for allowing more possible solutions and thus more flexibility. The set of capabilities C_a an agent provides results from the set of hardware allocated to the agent $\mathbf{H_a^{post}}$ during the resource allocation process. $\mathbf{H_a^{post}}$ is the set of hardware allocated to a by the solution of the resource re-allocation, being the set of decision variables for the CSOP. The relation between allocated hardware and provided capabilities is defined by a *common knowledge data base* that, queried with a certain set of hardware types, delivers the resulting capabilities as described in Section I. As every hardware has a defined weight $w(h)$ and each agent has a maximum payload $p(a)$, i.e., the maximum aggregated weight an agent is able to move with (which we claim to be an inherent capability of each agent), the allocation also has to take care of not overloading an agent (*payload-constraint*, cf. (1)). Of course, each hardware module can only be allocated to one agent at a time (*hardware-constraint*, cf. (2)). We define the CSOP as follows:

$$\begin{aligned}
\text{min.} \quad & \textstyle\sum_{a\in A} r_a \\
\text{s. t.} \quad & \forall a \in A : p(a) \geq \textstyle\sum_{h_a \in \mathbf{H_a^{post}}} w(h_a), && (1)\\
& \forall a \neq b \in A : \mathbf{H_a^{post}} \cap \mathbf{H_b^{post}} = \emptyset, && (2)\\
& \exists f : T \rightarrowtail A, \forall t \in T, C_t \subseteq C_{f(t)} && (3)\\
\text{with} \quad & r_a := \left| H_a^{pre} \, \Delta \, \mathbf{H_a^{post}} \right|, && (4)\\
& C_a \subseteq C := query\_database(\mathbf{H_a^{post}}), \\
& C_t \subseteq C, H_a^{pre}, \mathbf{H_a^{post}} \subseteq H
\end{aligned}$$

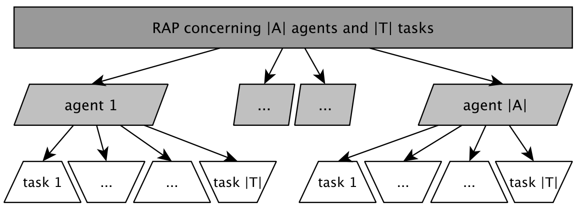

Fig. 3. Two step decomposition of the RAP. While the centralized problem definition from Section II-A has to handle all $|A|$ agents and $|T|$ a decomposed partial problem only has to regard one agent and one task at a time.

where A is the set of all agents, T the set of all relevant tasks, H the set of all hardware modules, C the set of all capabilities, H_a^{pre} is the set of hardware allocated to agent $a \in A$ previous to an resource re-allocation. In (4) we use the symmetric difference $A \Delta B := (A \setminus B) \cup (B \setminus A)$ to identify this set of changed hardware.

B. Decomposing the Resource Allocation Problem

When analyzing the complexity of the CSOP in Section II-A, it is obvious that its computability is only given for small sizes of H, A, and T. As for every agent there has to be decided which of the available hardware modules should be allocated to allow a valid task allocation, the number of solutions of the RAP is in $O(|A|^{|H|} \cdot |T|^{|A|})$. In other words, for every possible resource allocation ($|A|^{|H|}$ combinations [21]) we need to validate that a task allocation is possible ($|T|^{|A|}$ combinations [17]).[1] To come by this complexity, we propose to decompose the RAP, generate partial solutions, and aggregate them to achieve a solution approximately similar to the central solution concerning the optimization criteria, i.e., minimizing hardware reconfigurations (this heuristic already proved to be very efficient for resource allocation in the domain of virtual power plants we studied earlier [12]). Due to the global component of the RAP (cf. (2)) we prefer this contract net inspired solution [23] over specifying our problem as distributed constraint optimization problem (DCOP) [24]. Thereby we exploit a problem-specific decomposition and avoid messaging overhead. The decomposition can be achieved by dividing the calculation of $\mathbf{H_a^{post}}$ for all $a \in A$ into partial problems concerning only a single agent $\alpha \in A$, each. The resulting $|A|$ partial problems are individualized for each agent, thus, instead of regarding all other agents when minimizing the amount of hardware reconfigurations like it is done in the centralized CSOP (cf. Section II-A), each agent α only regards its own configuration, e.g., minimizes r_α and not $\sum_{a\in A} r_a$. Consequently, (2) is neglected here but respected in the following solution aggregation. Resulting from the definition of the task allocation problem as ST-SR (cf. Section II-A), the problem size of the RAP is further reduced in terms of the tasks that are regarded while calculating a partial solution. More precisely, for solving one partial problem only one task $\tau \in T$ has to be regarded at a time. To cover all tasks of the original problem, the partial problem has to be solved for

[1]Considering the worst case $|T| = |A|$, the problem is in $O(|A|^{|H|+|A|})$.

every task (i.e., $|T|$ times) by each agent α consequently (cf. Fig. 3). In the tradition of auction mechanisms, which are the most common market-based approaches [19], we call a solution to one partial problem *proposal*, i.e., the solution for the partial problem concerning agent α and task τ is named $H_\alpha^{pro}(\tau)$. This partial RAP again is encapsulated in a CSOP. For generating a valid proposal, the calculated partial solution $H_\alpha^{pro}(\tau)$ has to assert that the proposed resource allocation results in all relevant capabilities needed for the defined task τ (*capability-constraint*, cf. (5)), when queried within the data base while respecting α's *payload-constraint* (6). With that, the complexity of solving a decomposed partial problem is in $O(2^{|H|})$, respectively $O(|T| \cdot 2^{|H|})$ for all tasks. The definition of this partial CSOP is as follows (bottom of Fig. 3):

$$\begin{array}{lll} \text{min.} & r_\alpha & \\ \text{s. t.} & C_\tau \subseteq \mathbf{C}_\alpha, & (5) \\ & p(\alpha) \geq \sum_{h_a \in \mathbf{H}_\alpha^{\mathbf{pro}}(\tau)} w(h_\alpha) & (6) \\ \text{with} & r_\alpha = \left| H_\alpha^{pre} \Delta\ \mathbf{H}_\alpha^{\mathbf{pro}}(\tau) \right|, & \\ & \mathbf{C}_\alpha \subseteq C := query\_database(\mathbf{H}_\alpha^{\mathbf{pro}}(\tau)), & \\ & C_\tau \subseteq C, H_\alpha^{pre}, \mathbf{H}_\alpha^{\mathbf{pro}}(\tau) \subseteq H & \end{array}$$

To aggregate all partial solutions back to a solution for the original CSOP, another optimization problem has to be solved for finding the best combination of all partial solutions. Thereby, similar to the centralized solution defined in Section II-A, the amount of changed hardware modules shall be minimized while allocating resources to agents. The allocation itself again is subject to a set of constraints. Similar to the centralized CSOP, the *hardware-constraint* (8) as well as the *task-covering-constraint* (9) has to hold for a valid solution. In comparison to the centralized RAP, the complexity of the aggregation problem is heavily reduced. The *payload-constraints* of individual agents no longer have to be reviewed, as they are already taken into account in the proposal generation, c.f., (6). Moreover, the possible resource allocations for every agent are restricted to those covered by a proposal (*proposal-constraint*, cf. (7)). This reduces the amount of combinations from $|A|^{|H|} \cdot |T|^{|A|}$ to $|T|^{|A|}$ for the aggregation problem. This problem can be defined as follows:

$$\begin{array}{lll} \text{min.} & \sum_{a \in A} r_a & \\ \text{s. t.} & \forall t \in T, \exists a \in A : H_a^{pro}(t) = \mathbf{H_a^{post}}, & (7) \\ & \forall a \neq b \in A : \mathbf{H_a^{post}} \cap \mathbf{H_b^{post}} = \emptyset, & (8) \\ & \exists f : T \rightarrowtail A, \forall t \in T, C_t \subseteq C_{f(t)} & (9) \\ \text{with} & r_a = \left| H_a^{pre}\ \Delta\ \mathbf{H_a^{post}} \right| & \end{array}$$

The complexity of the decomposed approach for solving the RAP consequently is in $O(|T| \cdot 2^{|H|} + |T|^{|A|})$, adding up the complexity for generating proposals (done in parallel for all agents) with complexity of aggregating the proposals. Due to the decomposition and distributed calculation of partial problems it is not guaranteed that there is an optimal or even any solution for the aggregation problem. Some dependencies of the centralized CSOP are hidden for individual agents and considered the first time when aggregating the proposals, e.g., the *hardware-constraint* (resources might be allocated more often than once in different proposals). In Section III we propose an appropriate heuristic to reduce the occurrence of this dead-lock situations, among others.

III. Solving the RAP with TRANSFORMRS

In order to solve the RAP, we introduce two different algorithmic approaches. The first approach implements the methodical procedure to solve the RAP centrally while the second does so distributively. Since distributed approaches can lead to dead-lock situations, we further present an integrated solution combining the aforementioned two approaches in order to mitigate this, which we call **TRANSFORMRS** (**T**ask and **R**esource **A**llocatio**n** **S**trategy **for** **M**ulti-**R**obot **S**ystems). We assume that there are no communication issues, e.g., ensured by a reliable communication infrastructure and therefore each robot can always reach all other robots.

A. Solving the RAP centrally

The central algorithm for solving the RAP reconfiguration sequence for an ensemble with the size of n robots (i.e., $|A|$ agents) is illustrated by the activity in Fig. 4. The left side of the illustration shows the role distribution of the entire ensemble within the resource allocation process. Here, any robot can adopt the role of the coordinator (upper part) while all other robots are participants (lower part). The right side points out the different activities for the coordinator and the participants in two separated, but interacting activity diagrams. When the need arises to re-allocate resources in the ensemble as a task allocation failed in advance (cf. Fig. 2), the ensemble autonomously initiates the adaption process. At first, the two roles must be determined. Since all robots are equal from a software perspective, every robot is able to adopt the role of the coordinator. The actual assignment is decided by an appropriate leader election algorithm [25] we do not focus on in this paper. After the coordinator has been elected, it collects the necessary data for solving the RAP. Since it is a centralized approach, we made the assumption that the current system configuration is known globally by every robot. The first component of this data collection is the information pertaining to individual robots, such as their current configuration and maximum load capacity, whereas the second component consists of the task requirements in terms of their required capabilities. After the required data collection

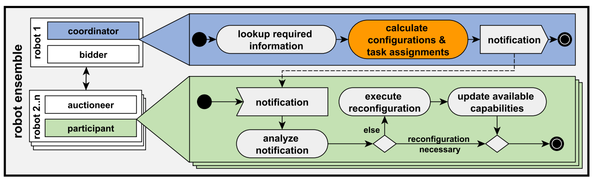

Fig. 4. Activity for solving the resource allocation problem centrally. One robot adopts the role of the coordinator, calculates a new resource allocation for all n participating robots, and informs them about the new system configuration. All other robots adopt the role of participants that realize the new system configuration. Task allocation and execution are not represented.

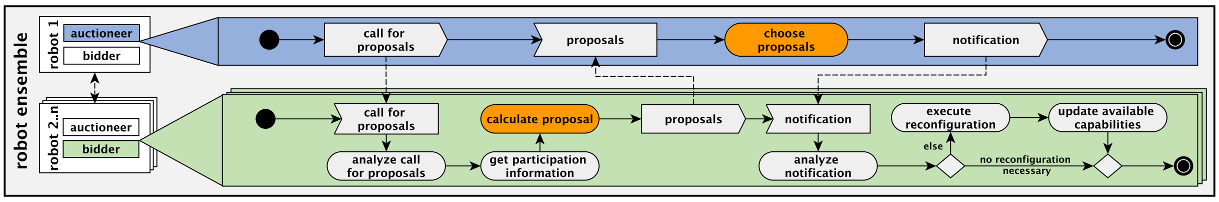

Fig. 5. Activity for solving the resource allocation problem distributively. One robot adopts the role of the auctioneer initiating a market-based resource allocation with a call for proposals (CfP). When receiving proposals from participating robots (bidders) it aggregates them to a new system configuration solving the RAP, if possible. Each bidder calculates only partial solutions to the RAP (cf. Fig. 3). Similar to the centralized solution, bidders realize new system configurations when they get informed by the auctioneer. Again, nether task allocation nor execution are represented.

is completed the coordinator centrally solves the Resource Allocation Problem (RAP) as defined in Section II-A. The overall solution contains the new configurations comprised of hardware modules, which the coordinator distributes to all available participants. As soon as the participant receives the message containing the new resource allocation, it analyzes it by comparing its current hardware configuration with the new configuration. Additionally, it saves the new (current) configuration of all other robots to keep the global knowledge consistent. Then it decides if a reconfiguration is necessary or not. If so, an actual reconfiguration is executed leading to change of hardware modules. Each robot is now equipped with new capabilities. If a robot already possesses the required hardware modules the reconfiguration is complete without any hardware modifications for this participant. The same process is performed on all participants involved in parallel. In a final step, all participants notify the coordinator that they have finished their own reconfiguration and the overall ensemble reconfiguration sequence is finished by the coordinator (not included in Fig. 4).

B. Solving the RAP Distributively

Depending on the application scenario, the number of hardware modules, tasks, and agents can each be very large. To counteract this complexity, we propose to decompose the RAP, by calculating partial solutions and aggregating them in order to find the solution that most closely resembles the central solution as formalized in Section II-B. Analogously to Fig. 4 for the central approach, Fig. 5 is separated into the same two parts for the distributed approach. As the problem is solved using a market-based approach, we introduce additional roles for the auction. In addition to its usual role, the coordinator adopts the role of the auctioneer while the participants additionally become bidders. Just like in the central case, every robot can take on any role in principle, so first of all a leader election is performed to determine the roles. As mentioned in Section II-B, the objects up for auction are the tasks for which proposals containing the hardware modules that the bidder would like to allocate for each task are generated. To start the auction, the auctioneer sends a call for proposals which consists of the required tasks including their required capabilities. After receiving the call for proposals the bidder analyzes it and collects the pertinent information. This entails information on all existing available hardware in the system (but not its current allocation), the individual robot's maximum payload as well as other local dependencies, e.g., between hardware types and capabilities. As soon as all required information has been collected, the bidder calculates a bid based on its local knowledge by solving the CSOP for proposals in Section II-B for every task of the RAP and sums up as a single proposal that is sent to the auctioneer. Each bid contains the hardware modules that the bidder would like to employ for the associated task i.e., a partial resource allocation as a partial solution of the RAP. These steps are carried similarly out by all other bidders. Once the auctioneer has received all proposals it aggregates all these partial solutions of the RAP back to a complete solution by solving the aggregating CSOP of Section II-B. All bidders are notified by sending a message with the new configuration, this time only containing the accepted bid including the participants' new hardware configuration instead of the whole new system configuration as needed in the central approach. From this step on, the activity follows the same procedure as in the centralized approach.

C. Integrated Solution: TRANSFORMRS

As described in Section II-B, each robot only considers its own configuration and available hardware modules when solving the partial RAP distributively, instead of considering all changes by other robots when minimizing the amount of hardware reconfigurations. Due to this decomposition, some dependencies of the centralized CSOP are not apparent to the individual robot, e.g., specific hardware modules might be allocated more often than once in different proposals (*hardware-constraint*). As soon as the partial solutions are aggregated, it is possible that no solution can be found and a dead-lock situation occurs. We circumvent this problem by requesting more than one proposal from every bidder, each of them unique concerning the requested hardware for a specific task to increase the number of possible allocations. In order to enable submission of multiple bids for the same task, bids of declining cost-effectiveness concerning r_α (k-best) must be calculated. The first proposal thus contains the best local solution, in which as little as possible plugging and unplugging processes take place. The second proposal represents the second best, locally calculated solution and so on. To allow all bidders to submit more than one proposal, the auctioneer additionally sends the desired number of proposals (k). Depending on k, the bidder calculates up to k proposals which are then sent to the auctioneer in the next step. The sequence follows the same procedure as described in the distributed approach. As

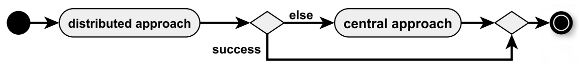

Fig. 6. Activity for our TRANSFORMRS approach. If no solution for the RAP can be determined distributively, a solution is calculated centrally.

this is still a heuristic approach, which does not guarantee a solution even if one exists, we overcome this dead-lock situation by integrating the distributed with the centralized approach (shown in Fig. 6), we call TRANSFORMRS. After the distributed approach fails to solve the RAP, additional communication takes place to provide the auctioneer with local information about all bidders in order to solve the RAP centrally. After the communication is completed the RAP is solved with the central approach analogously to Section III-A.

IV. Evaluation

In our evaluation we want to examine our approaches from Section III and compare them under different conditions. Therefore, we are interested in analyzing relevant properties of the central approach (**CA**, cf. Section III-A) and the distributed approach (**DA,** cf. Section III-B) concerning *1) the total time* consumed for computing a new resource allocation, *2) the quality* of a solution concerning needed hardware changes (cf. $\sum_{a \in A} r_a$ in Section II-A), and *3) the success rate*, i.e., how often is a solution to the RAP found. TRANSFORMRS introduced in Section III-C is not subject to the comparison as it combines the aforementioned CA and DA in a promising way. Instead, we investigate on our k-best heuristic for improving success rate and quality. We state the following hypotheses we want to verify in the following:

$\mathcal{H}_1$) DA outperforms CA in terms of computation time needed, especially with increasing problem size.

$\mathcal{H}_2$) The quality of solutions calculated by DA is equal to that calculated by CA.

$\mathcal{H}_3$) The success rate of DA is higher than that of the CA within a defined time limit.

$\mathcal{H}_4$) Increasing k in our k-best heuristic increases success rate and solution quality of TRANSFORMRS.

A. Testbed

For both CA and DA we define a maximum timeout for calculating valid solutions of 300 seconds, justified by the fact that we are deploying our implementation to real robots where immense timeouts are undesirable. This timeout is of high relevance for measuring the success rate as we only accept those solutions as valid ones that where calculated before it. The success rate also is influenced by the way we create our evaluation scenarios. As we do not evaluate our algorithms integrated with the rest of our system but as a modularized component we decided to determine initial conditions (i.e., agent and hardware properties as well as task requirements) randomly but equal for comparative evaluations. This of course does not guarantee that in general there is a valid solution for every initial configuration (cf. Section III) which is undesirable for our comparative evaluations. Therefore, we are not able to evaluate the results from our approaches to a "ground-truth" but only against each other. That means that for a given initial condition we say a RAP is solvable if and only if one of our approaches (including all heuristics) is able to find a solution within the allowed time frame.[2] While according to this, the number of total runs performed per configuration differs (cf. Table I), we ensured to have at least 100 valid runs per problem size (except for one) to achieve significant results. The problem size is determined by the number of participating agents $|A|$, the number of tasks $|T|$ defining the resource requirements, the number of capabilities $|C|$ a task can require, and the number of hardware modules $|H|$, randomly instantiated of different types (i.e., providing different capabilities). For identifying the individual effect on the properties we investigate in hypothesis $\mathcal{H}_1$ – $\mathcal{H}_4$, we systematically increase each of these parameters and create an evaluation run for every reasonable parameter combination. To reduce complexity in our evaluations, we restrict our MRTA to SR-ST. Consequently, e.g., combinations containing $|T| > |A|$ or $|T| > |H|$ are excluded. We iteratively increase the problem size in terms of $|A|$, $|T|$, and $|C|$ in steps of 2 in $[2, 10]$ (the limit of 10 assures feasibility also for mobile hardware while still being adequate for our problem domain) while keeping $|H| = 10$ (to assure there are enough modules for the requirements). To support our hypothesis, we perform paired t-tests (one- or two-sided, where indicated) for normally distributed populations, respectively Mann-Whitney-U tests for non-normal distributed ones. To verify the (non-) normal-distribution we perform Kolmogorov-Smirnov tests. The evaluations for Section IV-B were executed on high-performance server hardware (16 core @3GHz CPU, 32 GB RAM), evaluations for Section IV-C were executed on the same hardware as well as on single-board computers (ODROID-XU4) we usually use to control our mobile robots. To find solutions for the RAP (cf. Section II), we use Gecode as a solver for the CSOPs which we modeled with MiniZinc.[3] We do not take into account messaging overhead in our run time measurements (CA needs $|A|$ messages, DA $3 \cdot |A|$).

B. Results

To support our hypotheses $\mathcal{H}_1$ – $\mathcal{H}_3$ we compare results achieved from CA with that of the DA with $k = 1$ (cf. Table I, columns labeled with $k = 1$), while for supporting $\mathcal{H}_4$ we investigate in columns with $k = 2$ and $k = 3$ also. In Table I, problem sizes are encoded as, e.g., *10:8:6:4* for $|A| = 10$, $|H| = 8$, $|T| = 6$, $|C| = 4$. We further classify the problem sizes by selecting one of these parameters (*A,H,T,C*) to be either set to a *great* or *low* number and whether the problem size defined by the other parameters is *small* or *big*[4], e.g., analyzing the parameter agents (**A**) set to a low (**L** = *4*) size

[2]The ground-truth can not be determined otherwise as this would need the central solver to either prove a CSOP to be solvable or insolvable which often can not be computed in adequate time as the problem itself is NP-hard [21].

[3]ODROID on http://www.hardkernel.com/, MiniZinc on http://www.minizinc.org/, and Gecode on http://www.gecode.org/

[4]*Small / big*: We set other parameters to the lowest / highest possible values.

TABLE I

COMPARATIVE EVALUATION OF THE CENTRAL APPROACH (CA) AND DISTRIBUTED APPROACH (DA) AND EVALUATION OF THE K-BEST-ALGORITHM OF TRANSFORMRS. RUN TIME COMPARES NEEDED TIME FOR CA AND DA (FOR EACH $k \in \{1, 2, 3\}$), QUALITY THE AMOUNT OF NEEDED PLUG-IN / PLUG-OFF PROCESSES, AND SUCCESS THE AMOUNT OF VALID SOLUTIONS FOUND WITHIN OUR TIME LIMIT. FOR EACH COMPARISON. CA AND DA WORK ON EQUAL PROBLEMS. THIS IS ALSO ENSURED FOR THE K-BEST EVALUATION (EQUAL PROBLEMS FOR EACH K COMPARED).

problem size A:H:T:C	**# total runs**	**runtime in milliseconds: mean (std)**						**quality in needed reconfigurations: mean (std)**									**success: absolute (rate)** and **improvement absolute (relative)**						
		k=1		*k=2*		*k=3*		*k=1*			*k=2*			*k=3*			*CA*	*k=1*		*k=2*		*k=3*	
		CA	DA	CA	DA	CA	DA	CA	k=1	delta	k=1	k=2	delta	k=2	k=3	delta	# (rate)	# (rate)	+/- (rel)	# (rate)	+/- (rel)	# (rate)	+/- (rel)
ASL *2:10:2:2*	183	172 (8)	426 (26)	173 (8)	952 (58)	173 (8)	1272 (246)	0.77 (0.71)	0.77 (0.71)	0.00 (0.0)	0.77 (0.71)	0.77 (0.71)	0.00 (0.0)	0.77 (0.71)	0.77 (0.71)	0.00 (0.0)	183 (1.0)	129 (0.7)	0/54 (0.7)	158 (0.86)	29/0 (1.22)	158 (0.86)	0/0 (1.0)
ASG *10:10:2:2*	199	1419 (2958)	612 (114)	1710 (3573)	2873 (4544)	1751 (3620)	3952 (5136)	0.68 (0.65)	0.68 (0.65)	0.00 (0.0)	0.64 (0.64)	0.64 (0.64)	0.00 (0.0)	0.65 (0.64)	0.65 (0.64)	-0.04 (0.31)	199 (1.0)	187 (0.94)	0/12 (0.94)	183 (0.92)	11/15 (0.98)	186 (0.93)	3/0 (1.02)
ABL *4:10:4:4*	1138	2202 (8707)	893 (140)	2219 (6727)	2143 (240)	2219 (6727)	3247 (532)	2.73 (0.99)	2.80 (1.07)	-0.07 (0.35)	2.73 (0.99)	2.75 (1.03)	0.04 (0.3)	2.73 (0.99)	2.75 (1.03)	0.00 (0.0)	1138 (1.0)	391 (0.34)	0/747 (0.34)	733 (0.64)	342/0 (1.87)	733 (0.64)	0/0 (1.0)
ABG *10:10:4:4*	274	258744 (99323)	1136 (200)	248413 (107739)	70768 (92141)	240998 (114806)	70065 (80131)	1.32 (0.69)	1.32 (0.69)	0.00 (0.0)	0.28 (0.62)	2.24 (0.85)	0.12 (0.53)	0.31 (0.65)	2.21 (0.86)	-0.12 (0.62)	57 (0.21)	199 (0.73)	168/26 (3.49)	159 (0.58)	57/97 (0.8)	115 (0.42)	4/48 (0.72)
2:10:2:2		cf. **ASL**																					
TSG *6:10:6:2*	618	123731 (109208)	1167 (165)	141809 (120748)	20962 (10279)	141809 (120748)	44227 (16080)	3.94 (0.8)	3.94 (0.8)	0.00 (0.0)	3.19 (1.71)	3.86 (0.89)	0.00 (0.0)	3.19 (1.71)	3.86 (0.89)	0.00 (0.0)	563 (0.91)	21 (0.03)	4/546 (0.04)	181 (0.29)	160/0 (8.62)	181 (0.29)	0/0 (1.0)
TBL *10:10:2:4*	196	12939 (41173)	699 (118)	17199 (51855)	5710 (10388)	18812 (55935)	6206 (13932)	1.03 (0.73)	1.03 (0.73)	0.00 (0.0)	1.01 (0.74)	1.02 (0.73)	0.00 (0.0)	1.01 (0.73)	1.02 (0.73)	-0.03 (0.23)	190 (0.97)	180 (0.92)	3/13 (0.95)	174 (0.89)	13/19 (0.97)	175 (0.89)	4/3 (1.01)
TBG *10:10:6:4*	71	300375 (43)	1715 (291)	300375 (29)	75024 (88875)	300381 (28)	118401 (100551)	- -	- -	- -	0.00 (0.0)	7.00 (0.0)	0.00 (0.0)	- -	- -	0.00 (0.0)	4 (0.06)	42 (0.59)	42/4 (10.5)	26 (0.37)	25/41 (0.62)	6 (0.08)	0/20 (0.23)
HSL *2:2:2:2*	15465	125 (7)	384 (52)	125 (6)	809 (15)	125 (6)	1015 (197)	1.26 (0.67)	1.26 (0.67)	0.00 (0.0)	1.26 (0.67)	1.26 (0.67)	0.00 (0.0)	1.26 (0.67)	1.26 (0.67)	0.00 (0.0)	15465 (1.0)	9617 (0.62)	0/5848 (0.62)	12648 (0.82)	3031/0 (1.32)	12648 (0.82)	0/0 (1.0)
2:10:2:2		cf. **ASL**																					
HBL *10:4:4:4*	378	45130 (67567)	976 (167)	45245 (70612)	13233 (18664)	45245 (70612)	20657 (27895)	2.79 (0.59)	2.81 (0.65)	-0.02 (0.18)	2.64 (0.85)	2.82 (0.64)	0.00 (0.0)	2.64 (0.85)	2.82 (0.64)	0.00 (0.0)	361 (0.96)	128 (0.34)	7/240 (0.35)	294 (0.78)	166/0 (2.3)	294 (0.78)	0/0 (1.0)
10:10:4:4		cf. **ABH**																					
2:10:2:2		cf. **ASL**																					
CSG *2:10:2:10*	1124	18478 (64623)	590 (165)	18381 (64527)	1284 (328)	18381 (64527)	1512 (555)	1.99 (1.26)	1.99 (1.27)	0.00 (0.08)	1.92 (1.3)	2.00 (1.26)	0.00 (0.0)	1.92 (1.3)	2.00 (1.26)	0.00 (0.0)	1089 (0.97)	909 (0.81)	33/213 (0.83)	976 (0.87)	67/0 (1.07)	976 (0.87)	0/0 (1.0)
CBL *10:10:4:2*	316	210576 (130356)	967 (185)	158513 (145970)	110510 (108538)	87800 (133046)	71972 (77058)	1.40 (0.81)	1.40 (0.81)	0.00 (0.0)	0.66 (0.56)	1.48 (0.68)	0.06 (0.38)	0.82 (0.45)	1.24 (0.63)	0.00 (0.0)	125 (0.4)	230 (0.73)	153/48 (1.84)	123 (0.39)	56/163 (0.53)	57 (0.18)	0/66 (0.46)
CBG *10:10:4:10*	281	291803 (44311)	1613 (408)	290400 (46168)	38199 (63476)	291129 (45389)	42791 (69047)	1.80 (0.98)	1.80 (0.98)	0.00 (0.0)	0.08 (0.42)	3.00 (0.92)	0.04 (0.37)	0.07 (0.4)	3.02 (0.93)	-0.09 (0.57)	15 (0.05)	234 (0.83)	224/5 (15.6)	229 (0.81)	46/51 (0.98)	213 (0.76)	4/20 (0.93)

while other parameters define a big (**B** = $|A|$*:10:4:4*) problem is abbreviated as **ABL**.

Investigating $\mathcal{H}_1$: To support our hypothesis we compare run times for different problem sizes picked from Table I where we increase the parameter under observation from *low* to *great* and the problem size from *small* to *big* respectively. For achieving a fair comparison, we neglect results originating from initial conditions where DA determines that there is no valid solution.[5] In *small* configurations CA outperforms DA in every configuration significantly when the parameter of relevance is set to low (e.g., ASL: CA needs 172ms, DA needs 426ms, $p = 7.1 \cdot 10^{-215}$, or HSL: c = 125ms, d = 384ms, p = 0.0). This is due to the natural overhead (proposal generation and aggregation) DA suffers from and CA does not. When increasing to the respective *great* problem, this relation is reversed completely. While run time for CA increases from ASL to ASG with factor 8.25 DA increases only with factor 1.47. The same relation holds for CSL to CSG (for CA with factor 107.43, for DA with factor 1.38) and also for TSL to TSG (CA with factor 720.34, DA with factor 2.74). Thus, in these *great* configurations DA outperforms CA significantly. Only when increasing $|H|$ from HSL to HSG has less influence on CA's runtime (factor 1.38, factor for DA is 1.11, cf. ASL). In big problems CA's performance is even worse and already close to the calculation limit within our time out (we consider timed out runs with 300s in the average calculation which is an underestimation). While *low* numbers for $|A|$ (ABL needs $2,20$s), $|T|$ (TBL needs $12,94$s) and $|H|$ (HBL needs $45,13$s) are still feasible within our time limit (cf. CA in Fig. 7 left), increasing the parameter of interest to *great* (cf. CA in Fig. 7 right) leads to impractical run times of $258,74$s in ABG (factor 117.5), $300,385$s in TBG (factor 23.2), and $258,74$s in HBG (factor 5.73). For $|C|$ increasing from CBL to CBG seems to not have that huge influence on run time (factor 1.38) but this is only due to the time out restriction and the already high run time for CBL ($212,58$s) and thus run times for CBL and CBG are impractical (cf. Fig. 7). Like in *small* problems DA again is very robust against increasing from *low* to *great* in *big* problems for almost all parameters. The biggest increase in run time is caused by $|T|$ (from TBL with 699ms to TBG with 1715ms, factor 2.45), but for all problem sizes the distributed approach's run time is below 2000ms thus significantly lower than CA for all *big* problems in Table I. We see that especially with increasing the problem size, the DA outperforms CA by orders of magnitude significantly (Cohens's d test results in values from $258k - 300k$ in *big* problems in Table I) and thus $\mathcal{H}_1$ **holds**. This is also true for all problems bigger than $|A| \geq 6, |H| \geq 6, |T| \geq 4, |C| \geq 4$ where we see a mean run time for CA vs. DA of $1016s$ $(3192s)$ to $575s$ $(166s)$ which is significant ($p = 2.30 \cdot 10^{-5}$).

Investigating $\mathcal{H}_2$: To support $\mathcal{H}_2$, we compare the results from different problem sizes. Again, we want to perform fair comparisons and thus only consider results originating from initial conditions where both approaches achieved a valid solution. By analyzing our results in Table I, we can not find any significant difference between CA and DA (all data sets are normally distributed, so we use t-tests). Differences occurring in ABL ($p = 0.35$), HBL ($p = 0.84$), and CSG ($p = 0.94$) are only by chance. Thus, we find that if DA achieves a solution its quality is equal to that CA would have found and $\mathcal{H}_2$ **holds**.

[5]As those situations are uncovered very quickly, results would be falsified in terms of reducing the mean run time of DA while increasing that of the CA. This also explains varying average run times for CA in Table I for $k = 1, 2, 3$

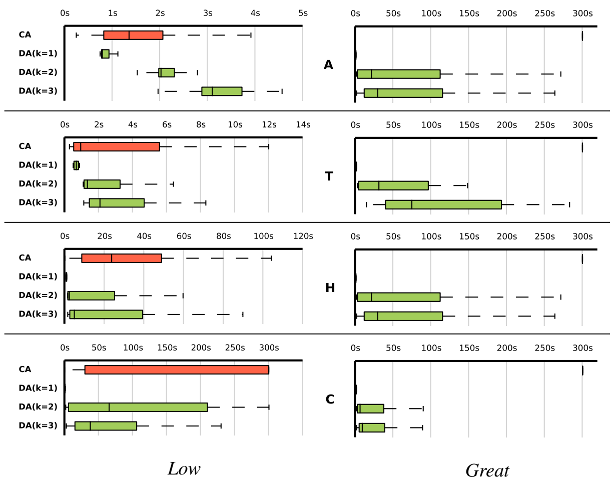

Fig. 7. Run time comparison of CA and DA (k=1,2,3) in *big* problems (box plots for time in seconds). We scale x-axis dynamically for each problem size.

For generalizing this statement, we also evaluated further problem sizes but never identified a significant difference (lowest confidence but not rejected for *4:8:2:2* with $c = 0.73$, $d = 0.86$, $p = 0.11$, as we test for equality here).

Investigating $\mathcal{H}_3$: To measure the success rate we only consider those runs performed on initial conditions where we definitely know a valid solution exists (cf. Section IV-A). In Table I within columns *success* the rate is given as ratio calculated by the total amount of solutions found by an approach divided by the number of total runs in the first column. In the second column, the improvement to the approach to the left is given in absolute numbers ($+$ indicates that, e.g., DA found solutions where CA did not and $-$ vice verca) and in relative numbers. For *small* problem sizes, we see that CA is capable of finding valid solutions in almost every run, indicated by a success rate of 1 (we only see minor drop outs with 0.91 in TSG and 0.97 in CSG and CBL). Compared to this, DA achieves a much more diverse spread success rate, ranging from 0.03 in TSG (where the central approach achieves 0.91) to 0.92 in TBL (central achieves 0.97 here). With a look on improvement from CA to DA in *small* problems, on the one hand the biggest improvement can be seen in CSG, where DA solves 33 problems CA could not, while there are 213 problems that on the other hand can not be solved. For all *small* problems the improvement concerning DA over CA is below 1.0. This indicates, that for *small* problem sizes $\mathcal{H}_3$ must be rejected. In *big* problem sizes this is not longer the case, as the CA can not hold its success rate. It especially suffers from increasing $|A|$, respectively $|H|$. While the success rate is still 1.0 for ABL and 0.96 in HBL, it drops to 0.21 in ABG / HBG). The same effect can be seen concerning $|C|$. From 0.4 for CBL the success rate drops to 0.05 in CBG. In contrast, DA is not influenced that hard by switching from *low* to *great* in *big* problems. The biggest drop can be seen from TBL with 0.92 to TBG with 0.59, which obviously is a very hard problem when taking into account that CA only successes with a rate of 0.06. For ABL the success rate of 0.34 (and for HBL 0.34 respectively) does not drop but increases instead in ABG / HBG to 0.73 which compared to the CA is an improvement of $+168/-26$ in absolute numbers (i.e., 349%). The same can be seen for CBL with an improvement from 0.73 to CBG with 0.83 where especially the improvement from CA to DA of $+224/-5$ (i.e., 1560%) is remarkable. This effect can be lead back to the increased flexibility through an increased amount of proposals in ABL / ABG and to the reduced chance for resource conflicts in HBL / HBG. To support our hypothesis we made some further evaluations including all other combinations in *big* problems to get a general statement where we compare the success rate of the CA to DA. The results is that the overall average success rate for CA is 0.04, for DA 0.6, which is a significant difference ($p = 2.5 \cdot 10^{-6}$) and thus $\mathcal{H}_3$ **holds** for *big* problems in general (like in Table I we define big problems as $|A| = 10, |H| = 10, |T| \geq 4, |C| \geq 4$).[6]

Investigating $\mathcal{H}_4$: We see, that the success rate is very diversely influenced from increasing k in our k-best algorithm. On the one hand, Table I shows that increasing k from 1 to 2 has a positive effect especially in *small* problems. For ASL, e.g., we see an improvement from 0.7 to 0.86 with 29 additional (0 less) problems solved or even more in TSG (from 0.03 to 0.29) or HSL (0.62 to 0.82). But also in *big* problems $k = 2$ improves success rate in certain configurations, e.g., in ABL from 0.34 to 0.64 and in HBL from 0.34 to 0.78. On the other hand, we also see heavily decreasing success rates, e.g., in ABG (from 0.73 to 0.58) or CBL (from 0.73 to 0.39). This drop effect is due to the increased combinatorial problem of aggregating partial problems which often is not possible within the time-out (we see, e.g., in CBL the mean run time of $k = 2$ compared to $k = 1$ increases from less than $1s$ to $110s$ and a standard deviation of $108s$). Increasing $k = 2$ to $k = 3$ does not seem to bring any benefit, again due to heavily increased run times preventing a solution to be determined before the time out. Some problems are even worse handled by $k = 3$ compared to $k = 2$, e.g., for CBL we see a drop in success rate from 0.39 to 0.18 with a total of 66 less solved problems. For all problem sizes included in Table I we can not find any significant improvement from $k = 1$ to $k = 2$ nor from $k = 2$ to $k = 3$ concerning solution quality, but in contrast see an increased amount of needed reconfigurations (this still holds for all other evaluated problem sizes). Thus $\mathcal{H}_4$ only **partially holds** concerning success rate and has to be rejected for quality improvements. Consequently, for TRANSFORMRS a portfolio approach for selecting the appropriate size of k for specific problem sizes.

C. Further Results

To validate real world applicability, we further evaluate *1) the scalability of DA* in problem sizes similar to that described in Section I and *2) the deployment on robot hardware*.

[6]There is still a significant difference for $|A| \geq 8, |H| \geq 6, |T| \geq 4, |C| \geq 4$ with CA (0.24) vs. DA (0.48) with ($p = 5.2 \cdot 10^{-6}$). Reducing $|A|$ to 6 seems to be the cutting point (CA 0.43 vs. DA 0.42) with ($p = 0.84$).

For *1)* we investigate in problems of size *10:h:10:6* with $h \in \{48, 96, 192\}$. The run time of DA in such problems still stays in a tolerable frame of $12.16s$ $(9.72s)$ for $h = 48$, $34.37s$ $(27.41s)$ for $h = 96$, and $87.89s$ $(64.26s)$ for $h = 192$. Remarkable is that the success rate of DA in those problems is always 1.0. This can be explained by the correlation of the increased amount of available hardware with the variance in allocated hardware within proposals. This induces that for great scale problems (concerning $|H|$) DA is very well suited. Evaluation of other problems increasing $|A|$ show that this is not longer true for $|A| > 10$. Those sizes need further decomposition strategies we want to investigate in our future research. For *2)* we deploy TRANSFORMRS on our robot hardware and evaluate ABG, TBG, HBG, and CBG from Section IV-B. We find that those problems are also feasible within appropriate time (ABG / HBG: $3.47s$ $(0.60s)$, TBG: $4.88s$ $(0.93s)$, CBG: $4.93s$ $(0.70s)$) with success rates between 0.83 and 0.92. Compared to the results in Section IV-B we see a hardware related increase ranging from 285% to 306%. Thus, our approach also passes the reality check.

V. Related Work

Several approaches already deal with the commonly known problem of resource allocation. Also the idea of a distributed, i.e., market-based problem solving is widely known, however it is rather more common for solving the task allocation problem [17], [26] than the resource allocation problem [21], especially in multi-robot systems. To the best of our knowledge, we are the first to solve the resource allocation problem in a multi-robot scenario for re-allocating robot capabilities, i.e., solving the RAP with limited, indivisible, not sharable, static, and multi-unit resources and actually execute the re-allocation online in a distributed, market-based fashion. The resource allocation problem first of all is classified by the type and characteristics of resources that are subject to the allocation. Following the classification of [21], resources can be continuous or discrete, divisible or indivisible, sharable or not sharable, static or dynamic, and single-unit or multi-unit. Approaches for distributed resource allocation in multi-agent systems typically deal with divisible and not sharable resources like energy [12], [27]. In that domain, the demanded amount of energy is the resource that shall be allocated to energy producers with the goal to optimally allocate the whole resource available. While this is a completely different optimization goal like the one we examine in this paper, [12] and [27] also use market-based, distributed mechanisms to allocate resources, which is possible in general when dealing with divisible resources [28]. Other approaches define the allocation of computational load needed to handle a computing task as resource allocation problem [29]. The goal of these approaches is to solve the resource allocation problem by minimizing the time needed to calculate a task and thereby is an instance of the resource scheduling problem [30], differentiating it from our resource allocation problem. Another research field that tackles the problem of resource allocation is the one of cloud networked robotics [31]. The resource to be allocated is either the bandwidth, i.e., the network the robots work in [32] or the robots themselves that can be allocated by other agents or humans [33]. When resource allocation and multi robot systems come up together in literature, often a task allocation problem is expressed as an resource allocation problem. In [34], e.g., a routing problem is defined as resource allocation problem on goal destinations that should only be visited once (tasks) while keeping the needed resources minimal. The term of self-reconfiguring robots up to now basically was used for a field of research investigating in algorithms for optimal, efficient re-shaping (e.g., snake-bot [35]), and self-assembly [36], most often focusing on mechanical connections of the robot modules [37]. In comparison to the robots empowered with self-reconfiguration abilities of our understanding, the robots in these approaches are limited in their actual usage in real world applications. The way we think of self-reconfiguration in terms of hardware exchanges for robots to achieve new capabilities was investigated very sparse up to now. In Preece et al. [38] robots can be equipped with different hardware for different tasks. In contrast to our approach, the allocation of resources is done centrally and offline by querying a static ontology that defines which capabilities are needed for certain tasks. Our system in comparison is designed to react to changed task requirements at run-time. Moreover, Preece et al. [38] only use unary relationships between hardware modules and capabilities, while our approach can exploit any relationship between these two.

VI. Conclusion

We model the problem of resource allocation in multi-agent systems as a constraint optimization problem to cope with typical issues that come up with robot ensembles. Resources we consider are hardware modules that can be easily connected and disconnected to each robot and that deliver certain capabilities for the corresponding robot. A new allocation of resources is needed when the robot ensemble is not able to fulfill tasks committed to it, i.e., the capabilities needed to handle those tasks are not provided by the robots. Within our model, we can easily respect the internal properties of the robots involved which are relevant for re-allocating resources, e.g., a robot's maximum payload or individual geometric designs.

For solving the resource allocation problem, we propose TRANSFORMRS an algorithmic approach that exploits the possibility of decomposing and solving the problem in a distributed fashion. Our evaluation shows that this approach significantly reduces the time used for calculating new resource allocations compared to a centralized one, especially with increased problems size in terms of agents, tasks, available hardware, or capabilities needed by tasks. Furthermore and to the best of our knowledge, we provide the first approach to adapt the capabilities provided by robots at run-time which overcomes the need of specialization typically restricting the applicability of robot systems in other current approaches.

In future experiments we plan to extend the models used for individual robots to respect increased robot heterogeneity, e.g., individual hardware-capability relationships resulting from heterogeneous geometric designs, that can easily be integrated in our current model. We expect that these will even emphasize more the advantages of our distributed approach which we detected in Section IV when increasing the problem size. Further, we already achieved some first results concerning the real world implementation of our robots equipped with the ability to autonomously reconfigure their capabilities that we plan to investigate in more in the near future. Also, we are already developing algorithms for robust execution of tasks.

REFERENCES

[1] B. Wolf, C. Chwala, B. Fersch, J. Garvelmann, W. Junkermann, M. J. Zeeman *et al.*, “The scalex campaign: Scale-crossing land surface and boundary layer processes in the tereno-prealpine observatory,” *Bulletin of the American Meteorological Society*, vol. 98, no. 6, pp. 1217–1234, 2017. [Online]. Available: https://doi.org/10.1175/BAMS-D-15-00277.1

[2] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira *et al.*, “Evolution of collective behaviors for a real swarm of aquatic surface robots,” *PLoS ONE*, vol. 11, no. 3, pp. 1–25, 03 2016. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0151834

[3] R. Thenius, D. Moser, S. Kernbach, I. Kuksin, O. Kernbach, E. Elena Kuksina *et al.*, “subclutron: a learning, self-regulating, self-sustaining underwater society/culture of robots,” *Art. Life and Intell. Agents Symposium, 2016*, 2016.

[4] O. Kosak, C. Wanninger, A. Angerer, A. Hoffmann, A. Schierl, and H. Seebach, “Decentralized coordination of heterogeneous ensembles using jadex,” in *2016 IEEE 1st Int. Workshops on Found. and Applications of Self* Systems (FAS*W)*, 2016, pp. 271–272.

[5] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset *et al.*, “Search and rescue robotics,” in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008.

[6] K. Daniel, B. Dusza, A. Lewandowski, and C. Wietfelds, “Airshield: A system-of-systems muav remote sensing architecture for disaster response,” in *Proc. 3rd Annual IEEE Systems Conf. (SysCon)*, 2009.

[7] O. Kosak, “A decentralised swarm approach for mobile robot-systems,” in *Organic Computing: Doctoral Dissertation Colloquium 2015*, vol. 7. kassel university press GmbH, 2015, p. 53.

[8] O. Kosak, C. Wanninger, A. Angerer, A. Hoffmann, A. Schiendorfer, and H. Seebach, “Towards self-organizing swarms of reconfigurable self-aware robots,” in *Found. and Applications of Self* Systems, IEEE Int. Workshops on*. IEEE, 2016, pp. 204–209.

[9] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan *et al.*, “An autonomous multi-uav system for search and rescue,” in *Proc. of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, ser. DroNet ’15. Florence, Italy: ACM, 2015, pp. 33–38. [Online]. Available: http://doi.acm.org/10.1145/2750675.2750683

[10] G. De Cubber, D. Serrano, K. Berns, K. Chintamani, R. Sabino, S. Ourevitch *et al.*, “Search and rescue robots developed by the european icarus project,” in *7th Int. Workshop on Robotics for Risky Environments*. Citeseer, 2013.

[11] O. Kosak, “Facilitating planning by using self-organization,” in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sept 2017, pp. 371–374.

[12] O. Kosak, G. Anders, F. Siefert, and W. Reif, “An Approach to Robust Resource Allocation in Large-Scale Systems of Systems,” in *Self-Adaptive and Self-Organizing Systems (SASO), 2015 IEEE 9th Int. Conf. on*, 2015, pp. 1–10.

[13] D. C. Andreas, C. D. Eckhoff, and R. D. Loveman, “Serial device daisy chaining method and apparatus,” Aug. 9 2005, US Patent 6,928,501.

[14] L. Braubach and A. Pokahr, “Developing distributed systems with active components and jadex,” *Scalable Computing: Practice and Experience*, vol. 13, no. 2, pp. 100–120, 2012.

[15] O. Shehory and S. Kraus, “Methods for task allocation via agent coalition formation,” *Art. Intelligence*, vol. 101, no. 1, pp. 165 – 200, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370298000459

[16] I. Georgievski and M. Aiello, “An overview of hierarchical task network planning,” *CoRR*, vol. abs/1403.7426, 2014. [Online]. Available: http://arxiv.org/abs/1403.7426

[17] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The Int. Journ. of Robotics Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[18] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger *et al.*, “The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments,” in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov 2012, pp. 1–4.

[19] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-based multirobot coordination: A survey and analysis,” *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.

[20] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *The Int. Journ. of Robotics Res.*, vol. 32, no. 12, pp. 1495–1512, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/12/1495.abstract

[21] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet *et al.*, “Issues in multiagent resource allocation,” *Informatica*, vol. 30, no. 1, 2006.

[22] E. Tsang, “Foundations of constraint satisfaction,” 1995.

[23] R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver,” *IEEE Trans. on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[24] F. Fioretto, E. Pontelli, and W. Yeoh, “Distributed constraint optimization problems and applications: A survey,” *CoRR*, 2016.

[25] F. S. Gharehchopogh and H. Arjang, “A survey and taxonomy of leader election algorithms in distributed systems,” *Indian Journ. of Science and Technology*, vol. 7, no. 6, p. 815, 2014.

[26] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” in *Cooperative Robots and Sensor Networks 2015*. Springer, 2015, pp. 31–51.

[27] H. F. Wedde, “Dezent – a cyber-physical approach for providing affordable regenerative electric energy in the near future,” in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, Sept 2012, pp. 241–249.

[28] K. Lai, B. A. Huberman, and L. R. Fine, “Tycoon: A distributed market-based resource allocation system,” *CoRR*, vol. cs.DC/0404013, 2004. [Online]. Available: http://arxiv.org/abs/cs.DC/0404013

[29] S. Banerjee and J. P. Hecker, “A multi-agent system approach to load-balancing and resource allocation for distributed computing,” in *First Complex Systems Digital Campus World E-Conference 2015*, P. Bourgine, P. Collet, and P. Parrend, Eds. Cham: Springer International Publishing, 2017, pp. 41–54.

[30] R. Kolisch and S. Hartmann, *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*. Boston, MA: Springer US, 1999, pp. 147–178. [Online]. Available: https://doi.org/10.1007/978-1-4615-5533-9_7

[31] K. Kamei, S. Nishio, N. Hagita, and M. Sato, “Cloud networked robotics,” *IEEE Network*, vol. 26, no. 3, pp. 28–34, May 2012.

[32] L. Wang, M. Liu, and M. Q. H. Meng, “A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems,” *IEEE Trans. on Cybernetics*, vol. 47, no. 2, pp. 473–484, Feb 2017.

[33] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, “Cloud robotics: Current status and open issues,” *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[34] S. Rathinam, R. Sengupta, and S. Darbha, “A resource allocation algorithm for multivehicle systems with nonholonomic constraints,” *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 98–104, Jan 2007.

[35] R. Thakker, A. Kamat, S. Bharambe, S. Chiddarwar, and K. M. Bhurchandi, “Rebis - reconfigurable bipedal snake robot,” in *2014 IEEE/RSJ Intern. Conf. on Intel. Robots and Systems*, Sept 2014, pp. 309–314.

[36] M. Yim, W. m. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson *et al.*, “Modular self-reconfigurable robot systems,” *IEEE RAM*, vol. 14, no. 1, pp. 43–52, 2007.

[37] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, “Design of the atron lattice-based self-reconfigurable robot,” *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, Sep 2006. [Online]. Available: https://doi.org/10.1007/s10514-006-8546-1

[38] A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley *et al.*, “Matching sensors to missions using a knowledge-based approach,” *Proc. of SPIE: Defense Transformation and Net-Centric Systems*, vol. 6981, pp. 698 109–1, 2008.